Visualisierung konzeptioneller Beschreibungen von Programmieraktivitäten

Diplomarbeit zur Erlangung des akademischen Grades
Diplom-Informatiker

FREIE UNIVERSITÄT BERLIN

INSTITUT FÜR INFORMATIK

Betreuender Hochschullehrer:
Prof. Dr. Lutz Prechelt
Zweitprüfer:
Prof. Dr. Elfriede Fehr
Betreuer:
Stephan Salinger

Vorgelegt von Nicolas José Ngandeu
Geboren am 10. November 1972 in Sofia
Matrikelnummer: 3260940

Abgabedatum: 23. Februar 2007

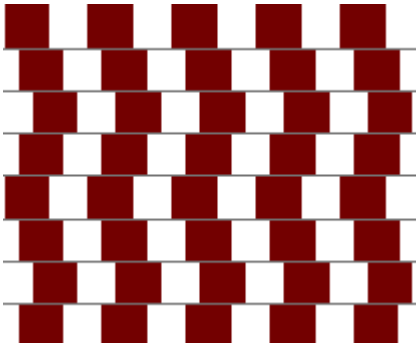
Eidesstattliche Erklärung:

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit zum Thema „Visualisierung konzeptioneller Beschreibungen von Programmieraktivitäten“ bis auf die offizielle Betreuung selbst und ohne fremde Hilfe angefertigt habe. Die benutzten Quellen und Hilfsmittel vollständig angegeben sind.

Berlin, den 23.02.2007 ...……..

(Unterschrift)

Zusammenfassung



Der Volksmund lehrt, dass ein Bild mehr als tausend Worte sagt. Aber kann ein Bild nicht auch mehr als tausend Worte lügen? In der obigen Abbildung ist schwer zu erkennen, ob die horizontalen Striche parallel sind oder nicht. Die Visualisierung wird in sehr vielen Bereichen (Wissenschaft, Wirtschaft etc.) gebraucht. Sie kann aber bei schlechtem Design oder schlechter Realisierung mehr schaden als helfen.

Im Rahmen dieser Diplomarbeit wird ein Werkzeug für die Informationsvisualisierung entworfen und implementiert. Es sollen Programmiertätigkeiten von Entwicklern grafisch darstellt werden. Mit der Generic TV ist es möglich, zeitabhängige abstrakte Daten abzubilden. Dadurch kann man sie besser analysieren.

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

Die Software Generic TV bietet zahlreiche Funktionalitäten. Sie bürgen für die Möglichkeit verschiedener Sichten (Multiple Views) auf die Inputdaten. Die generische Schnittstelle und Erweiterungen der Software erlauben eine Visualisierung unterschiedlich formatierter Inputdateien. Generic TV soll den Mitarbeitern der Arbeitsgruppe „Software Engineering“ der Freien Universität Berlin bei der explorativen Analyse von Programmiertätigkeiten dienlich sein.

Für Alexandra, Malik und Nandi

Danksagungen

Ich möchte mich bei Dipl.-Math. Stephan Salinger für die gute Betreuung, die konstruktive Kritik und das stets vorhandene Verständnis bedanken.

Für die Unterstützung und die Zusammenarbeit bei der Einarbeitung in den Forschungsgegenstand möchte ich mich bei Dipl.-Inf. Sebastian Jekutsch bedanken.

Ein besonderer Dank für das Korrekturlesen geht an Manuel Frey.

Ohne die Unterstützung meiner liebevollen Frau Alexandra und meiner Kinder, Malik und Nandi, hätte ich diese Arbeit nicht bewerkstelligen können.

Papa, Maman merci de m'avoir toujours soutenu.
Sans vous je n'aurais jamais réussi !!!

Inhaltsverzeichnis

I. Einleitung 7

I.1. Einführung und Motivation 7
I.2. Aufbau der Arbeit 14

II. Problemstellung 15

III. Die Akteure 29

III.1. Mikroprozesse des Programmierens 29
III.2. Die Inputdaten 30
III.3. Das Framework 36
III.3.1. Motivation der Auswahl und Vorstellung von JfreeChart 36
III.3.2. Einblick in die Architektur von JfreeChart 40

IV. Informationsvisualisierung: Stand der Forschung und der Technik 42

IV.1. Begriffsklärung und Einführung 42
IV.2. Einflussfaktoren auf eine Visualisierung 45
IV.2.1. Bearbeitungsziele einer Visualisierung 45
IV.2.2. Wahrnehmungskapazitäten des Benutzer 46
IV.3. Der Visualisierungsprozess 48
IV.4. Darstellungstechniken 51
IV.4.1. Übersicht & Detail 51
IV.4.2. Scrolling & Paging 52
IV.4.3. Information Hiding 52

IV.5. Visualisierungstechniken für zeitabhängige Daten 53
IV.5.1. Visualisierung von Daten mit Zeitbezug 53
IV.5.2. Visueller Attribute zur Darstellung von Datenwerten. Quantitative Darstellung 56

V. Analyse und Konzeption der Generic TV 62

V.1. Analyse 62
V.1.1. Analyse und Umsetzung der Raumdarstellungstechniken 65
V.1.2. Analyse und Umsetzung der Zeitdarstellungstechniken 68
V.1.3. Analyse und Umsetzung der Realisierung einer quantitative Darstellung 68

V.2. Arbeitsablauf mit Generic TV (Produktsübersicht) 72

V.3. Konzeption 78
V.3.1. Einflussfaktoren 78
V.3.2. Abstraktionsschichten 78
V.3.2.1. Abstraktionsschicht 1: DataReader 79
V.3.2.2. Abstraktionsschicht 2: DataLoader 82
V.3.2.3. Abstraktionsschicht 3: DataViewer 87
V.3.2.4. Abstraktionsschicht 4: UserTools 88

VI. Fazit 94

VI.1. Kritik: Probleme, benötigte Verbesserungen 94

VII. Literatur 98

Anhang I – Aufgabenstellung 102
Anhang II – Pflichtenheft 103
Anhang III – Analyse der grafischen Elemente 114
Anhang IV – Testdokumentation 121

I. Einleitung

I.1. Einführung und Motivation

"Visualization is more than a method of computing. Visualization is the process of transforming information into a visual form, enabling users to observe the information. The resulting visual display enables the scientist or engineer to perceive visually features which are hidden in the data but nevertheless are needed for data exploration and analysis" [NM97]

Durch die Visualisierung werden abstrakte Daten oder Zusammenhänge grafisch repräsentiert. Relevante Eigenschaften der Daten sollen intuitiv erfassbar, verständlich und bewertbar sein[SM04]. Eine Visualisierung kann in den verschiedensten Formen realisiert werden. Beispiele für Visualisierungen wären: Eine Karte des Liniennetzes der Berliner U-Bahn, die Verfilmung eines Drehbuchs oder das Periodensystem der Elemente. Diese grafischen Repräsentationen von Daten sind analysierbar.

Der Einfluss der visuellen Datenanalyse wächst stetig. Ihre Bedeutung macht sich insbesondere in der Wissenschaft bemerkbar, vor allem bei Daten, die groß und- oder komplex sind.

Die Visualisierung ist eine wichtige Methode, um Menschen komplexe Informationen schnell und leicht zu übermitteln, Zusammenhänge erkennbar darzustellen sowie die Mensch-Maschine-Interaktion zu vereinfachen. Die Fähigkeiten und Eigenschaften des menschlichen visuellen Systems werden besonders ausgenutzt. Das Auge ermöglicht die schnelle Aufnahme einer großen Informationsmenge, die im Gehirn verarbeitet werden kann. Die Abbildung 1 zeigt ein kleines Beispiel der Zusammenarbeit zwischen Auge und Gehirn. Beim Versuch die Ziffer Null in den ersten 100 Nachkommastellen der Zahl Pi zu finden, kann man im unteren Teil der Abbildung 1 eine farbliche Visualisierung sehen. Sie bietet Vorteile beim schnellen Erkennen der Null-Nachkommaziffern.

Abbildung 1 – Visualisierung der zahlen null bei ersten 100 Nachkommastellen der Zahl PI

Visualisierungen werden in allen Bereichen der Forschung und der Wirtschaft eingesetzt. Das Spektrum der Benutzung von Visualisierungen weitet sich vor allem im Ingenieurwesen, der Medizin und der Wirtschaft aus. Die Online-Enzyklopädie Wikipedia liefert sehr interessante Beispiele für den unterschiedlichen Einsatz von Visualisierungen:

- *Ein Chef zeigt die Umsatzzahlen des vorigen Quartals in einem Balkendiagramm.*

- *Der Turbinen-Ingenieur lässt sich den Luftstrom anhand der Dichten bildlich darstellen und betrachtet den zeitlichen Verlauf in einem Film.*

- *Der Leiter der Revenue-Management-Abteilung einer Fluggesellschaft erstellt für einen bestimmten Zeitraum ein Balkendiagramm der Entwicklung von stornierten Buchungen.*

In der computerisierten Welt von heute wird Visualisierung zunehmend zu einem Kommunikationsmittel. Also zu einer Schnittstelle zwischen Mensch und Maschine. Durch Visualisierungsprogramme, wie zum Beispiel Microsoft Excel, ist die grafische Darstellung von abstrakten Daten mittels Computern möglich. Sie wird Computer-generierte Visualisierung genannt.

Dies heißt aber nicht, dass Visualisierung eine Erfindung des Computerzeitalters ist. Der Bedeutsamkeit des Informationstauschs zwischen der grafischen Darstellung und dem Menschen war man sich schon vor mehr als einhundert Jahren bewusst. Der französischer Ingenieur Charles Joseph Minard (1781-1870) zeigte in einer Darstellung (siehe Abbildung 2), wie man von mehreren Variablen abhängige Daten darstellen kann.

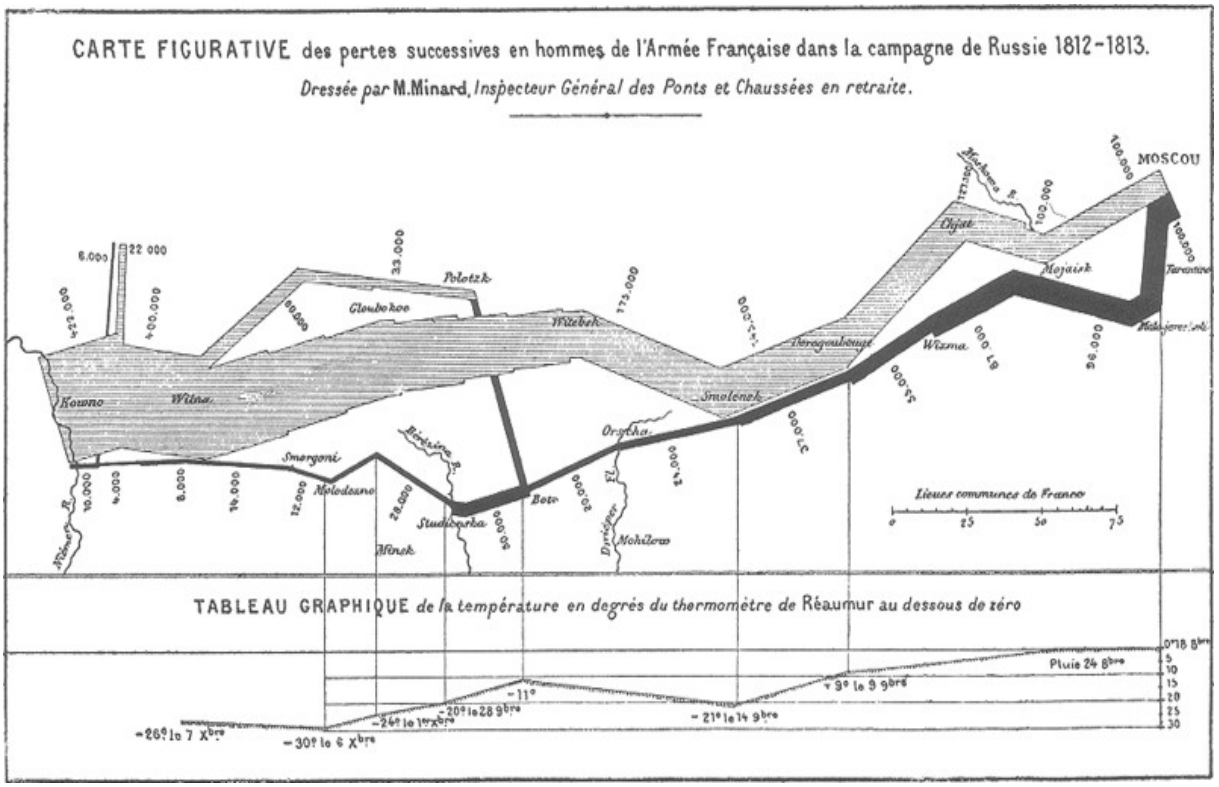


Abbildung 2 - Joseph Minard: Karte des Russlandfeldzuges von Napoleon von 1812-1813 [DR99]

In dieser einzigen Grafik sind vier unterschiedliche Parameter visuell miteinander kombiniert: Die Größe der Armee (durch die Breite), der Aufenthaltsort der Truppen zu einem bestimmten Zeitpunkt (durch die Namen der Städte), die Bewegungsrichtung der Armee (von Kowno nach Moskau und zurück) und die Umgebungstemperatur beim Rückzug der Armee nach der Belagerung von Moskau.

Durch die visuelle Darstellung lassen sich Aspekte und Beziehungen übersichtlicher als mit anderen Darstellungen veranschaulichen. Die Computer-generierten Darstellungen ermöglichen dem Benutzer durch Interaktionen mit dem Computer die Darstellung dieser Daten zu beeinflussen. Er nimmt nur nicht die Daten visuell wahr, sondern kann sie nach persönlichen Interessen auch gestalten.

Bei einer Fluggesellschaft könnten zum Beispiel die abstrakten Daten „Buchungen“ im Multiple View-Verfahren dargestellt werden. Die unterschiedlichen Sichten wären eventuell als:

- normale Buchungen,
- Gruppenbuchungen,
- bezahlte Buchungen,
- stornierte Buchungen

klassiert.

Diese unterschiedlichen Sichten auf den identischen Datenbestand[1] sind Stärke der Computer-genierten Visualisierung. Die Manipulation der Daten-Darstellung durch den Benutzer ist eine wichtige Komponente der Visualisierung. Die Visualisierung wurde als eigenständige Disziplin nach der Veröffentlichung des NSF Landmark Reports *Visualization in Scientific Computing* im Jahre 1987 [CB87] etabliert. In dieser Disziplin werden zwei Visualisierungs-Arten unterschieden [CB87]:

- *Die wissenschaftliche Visualisierung ist die Darstellung von gemessenen Daten oder Simulationsergebnissen, denen unmittelbar physikalische Prozesse zugeordnet werden können.*
- *Die Informations-Visualisierung ist die Darstellung von Daten, die nicht unmittelbar mit physikalischen Zuständen und Prozessen assoziiert werden. Dies sind zum Beispiel Dokumente, Börsenergebnisse und Demografiedaten.*

Heutzutage können die zwei Visualisierungsarten durch Computer generiert werden. Ihre Gemeinsamkeit ist, dass es sich um eine Daten-Darstellung handelt, die für das menschliche Auge erzeugt wurde. Den Unterschied machen die Eigenschaften der Inputdaten. Im Wesentlichen unterscheidet auf dem Gebiet Visualisierung folgende Datenklassen [SH04]:

- Multiparameterdaten: Hierbei handelt es sich um Datenmengen, die Werte von mehreren Parametern beinhalten, wie es beispielsweise bei großen Datenbanken der Fall ist.
- Volumendaten: Sind skalare Daten, die über einem 3-dimensionalen regelmäßigen Gitter gegeben sind. Beispiele hierfür sind Datensätze, die durch Simulationen gewonnen werden.
- Vektorielle Daten: Hierbei ist mindestens ein Vektor pro Gitterpunkt eines strukturierten Gitters gegeben. Dies können z.B. Geschwindigkeitsvektoren sein, um das Geschwindigkeitsfeld einer Strömung zu charakterisieren.

[1] Wie im obigen Fall „Buchungen“

Die Abbildung-3 veranschaulicht die unterschiedlichen Datenklassen und ihre Darstellung. Das erste Bild zeigt die Abbildung von Multiparameterdaten mit Tiefendiagrammen im Arkonabecken der westlichen Ostsee. Die zweite Darstellung ist eine Visualisierung von Vektordaten aus dem Konrad-Zuse-Institut und die dritte Grafik ist eine Visualisierung von Volumendaten am Beispiel von MRI-Daten mittels einer Sonde.

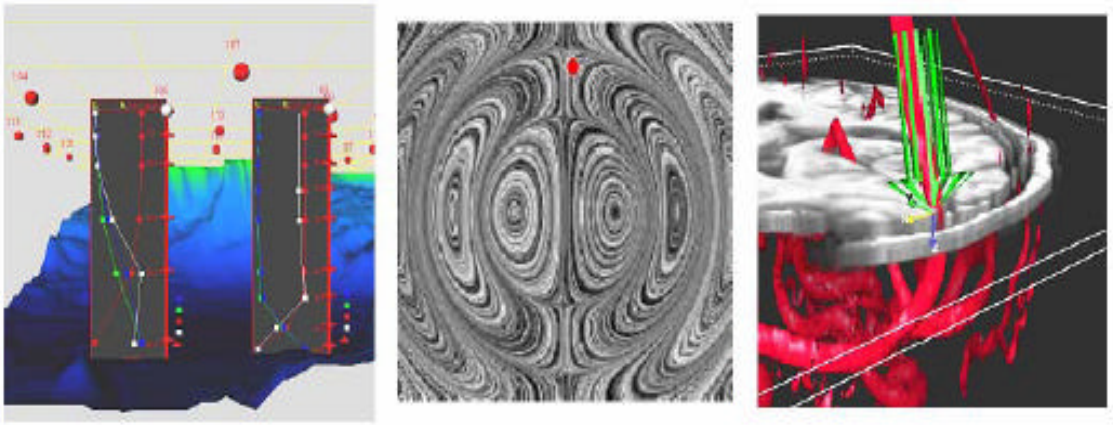


Abbildung 3 – Visualisierungsbeispiele unterschiedlichen Datenklassen [SH04]

Unabhängig von ihrer Art (wissenschaftliche oder Informationsvisualisierung), von den darzustellenden Datenklassen, vom Anwendungszweck und der Präsentationsform, lassen sich vier elementare Aufgaben der Visualisierung formulieren [DR99]:

- Symbole, Diagramme oder Animationen helfen komplexe Prozessabläufe und Objektbeziehungen zu veranschaulichen und gegebenenfalls zu vereinfachen.

- Die Visualisierung vereinfacht den Zugang zu Massendaten, z.B. durch Klassifikation und Datenstrukturierung.

- Die Visualisierung hilft bei der Analyse und Interpretation von Daten, bei der Aufdeckung verborgener Trends sowie bei der Erkennung von Mustern.

- Die Visualisierung entspricht der Neigung und Kultur der menschlichen Spezies, visuelle Informationsprozesse und Repräsentationsformen zu bevorzugen. Aus der Gehirnforschung ist bekannt, dass sich Visualisierung positiv auf die Gedächtnisleistung und auf die menschliche Informationsaufnahme auswirkt.

Im Rahmen dieser Diplomarbeit soll ein Informations-Visualisierungs-Werkzeug entworfen werden, das die Visualisierung von Programmieraktivitäten von Softwareentwicklern ermöglicht.

Informationsvisualisierung beinhaltet unter anderem die computergestützte Aufbereitung und die visuelle Repräsentation abstrakter Informationen. So wird der Zugang zu elektronisch gespeicherten Daten erleichtert. Alle Konzepte, Methoden und Werkzeuge zur visuellen Darstellung von Informationen werden als Informationsvisualisierung zusammengefasst.[DR99] Die Informationsvisualisierung soll textbasierte Informationssysteme strukturell aufbereiten und so ihre Darstellung ermöglichen. Dabei werden folgende Ziele verfolgt [DR99]:

- Das Auffinden spezifischer Informationsdomänen in großen Datenbeständen.
- Das Erkennen von Relationen, Strukturen oder Trends in unstrukturiert erscheinenden Informationsmengen.
- Multiple Views, d.h. unterschiedliche Sichten auf identische Datenbestände.

Der Zusammenhang zwischen Informationsvisualisierung und Programmieraktivitäten ist nicht unbedingt selbstverständlich. Um die Verbindung zwischen Informationsvisualisierung und Programmieraktivitäten deutlicher darzustellen, wäre die Beantwortung folgender Fragen wichtig:

- Welches bzw. was sind die Programmieraktivitäten von Entwicklern?
- Warum würde es sich lohnen, sie zu beobachten?

Welches bzw. was sind die Programmieraktivitäten von Entwicklern?

Unter Programmieraktivitäten von Entwicklern sind alle Tätigkeiten (Schritte) zusammengefasst, die durch die Entwickler ausgeübt werden. Unter solchen Tätigkeiten können zum Beispiel das Editieren einer Datei oder das Durchführen eines Testlaufs verstanden werden.

Im Rahmen unterschiedlicher Forschungen an der Freie Universität Berlin (FUB) werden diese Schritte aufgezeichnet und mit durch den Begriff ***Mikroprozess***[2][JS05] definiert. Im Kapitel III wird darauf ausführlich eingegangen. Das Ziel der Aufzeichnung ist die Vorgehensweise von Programmierern bzw. Paarprogrammierern[3] über ein bestimmtes Zeitintervall zu untersuchen. Denn so können typische Aktivitäts- und ***Verhaltensmuster*** entdecken werden. Der Mikroprozess des Programmierens ist die Abfolge einzelner individueller Aktivitäten der Entwickler bei ihrer Arbeit [JS05]. Dazu gehören nicht nur die Tätigkeiten am Computer, sondern auch sonstige Vorgänge, die einen Einfluss auf die Arbeit der Programmierer haben könnten. Beispiele dafür wären Unterbrechungen des Arbeitsprozesses (wegen eines Telefonats oder Holens einer Tasse Kaffee oder eine kurze Unterhaltung mit einem Kollegen). Diese Tätigkeiten können also sehr unterschiedlicher Art sein. Trotzdem gibt es Gemeinsamkeiten: Jede Tätigkeit hat einen Zeitbezug. Eine Tätigkeit fängt zum Zeitpunkt „t“ an und hört zum Zeitpunkt „t + n“ auf. Daraus ergibt sich, dass jede Tätigkeit eine genaue Dauer besitzt.

Warum würde es sich lohnen, die Programmieraktivitäten zu beobachten und zu analysieren?

Für die Forscher der FUB sind zwei Fragen besonders interessant:

- Welche Interaktionen sind bei der Zusammenarbeit von Paarprogrammierern besonders produktiv?

[2] Zur Hervorhebung werden neue wichtige Begriffe ***kursiv und fett*** gesetzt.
[3] Paarprogrammieren bedeutet, dass zwei Programmierer an einem Computer an einer Programmieraufgabe gemeinsam arbeiten

- An welchen Aktivitätsfolgen kann man erkennen, dass (Paar-)Programmierer gerade unsicher oder unkonzentriert sind? Woraus eventuell Programmierfehler entstehen.

Um solche Schlussfolgerungen ziehen zu können, müssen die Forscher der FUB in der Lage sein, bestimmte Mikroprozesse der Programmierer zu beobachten. Eine solche Beobachtung ist wiederum nur möglich, wenn die Mikroprozesse aufgezeichnet bzw. beobachtet werden. Zurzeit werden diese Aufzeichnungen auf unterschiedliche Art und Weise durchgeführt:

- Videosequenzen: Die Programmierer werden in Bild und Ton aufgezeichnet. Synchron wird auch der Desktop (ihre aktuelle Arbeit) aufgenommen. Alle diese Daten werden zu einem Video zusammengefasst. Das Verhalten der Programmierer wird beobachtet. Mit der Hilfe von bestimmten Werkzeugen (wie Atlas.ti[4]) werden die Aktivitäten der Entwickler auf diesen Videosequenzen beschrieben und durch Label ***annotiert*** bzw. ***kodiert***. Das gesamte Verfahren sowie die dazu gehörenden Begriffe werden im nächsten Kapitel ausführlich vorgestellt.

- Bestimmte Mikroprozesse der Softwareentwicklung werden auf automatische Weise aus den Softwareentwicklungswerkzeugen (wie Eclipse) aufgezeichnet. Dieses automatisierte Verfahren wird durch ein Werkzeug namens **ElectroCodeoGram (ECG)** realisiert[EC05]. Das ECG wurde zur Aufzeichnung von Programmierschritten in der FUB entworfen und implementiert.

Nach mehreren Verarbeitungsprozessen mit unterschiedlichen Verfahren[5] entsteht eine Proktokollierung der Mikroprozesse der Programmier. Diese textbasierte Daten werden als Dateien gespeichert. Die Informationsvisualisierung wurde so definiert, dass die Möglichkeit textbasierte Informationssysteme strukturell aufzubereiten und darzustellen, gegeben sein muss. Eine Visualisierung der aufgezeichneten Programmiertätigkeiten kann man mit der Visualisierung von abstrakten zeitabhängigen Daten jetzt gleichstellen.

<u>Zeitabhängige Daten</u>

„Eine Zeitabhängigkeit in den Daten ist gegeben, wenn eine unabhängige Variable, das heißt eine Dimension des Beobachtungsraumes, mit der physikalischen Größe „Zeit“ assoziiert ist.“ [SH06]

Die Visualisierung zeitabhängiger Daten hat eine sehr lange Geschichte. In der Wissenschaft wurde die erste Visualisierung einer Zeitreihe[6] im 18. Jahrhundert durch Lambert eingeführt[SM03]. Es gibt sehr viele Anwendungsbereiche der Analyse von zeitabhängigen Daten. Autretende Fragen solcher Analysen könnten z.B. sein:

- Verringert die Einführung der Gurtpflicht die Anzahl tödlicher Verkehrsunfälle?

- Führt Arbeitslosigkeit zur Depressionen beim Menschen?

- Führt die Einnahme des Medikamentes „X“ zu einer Besserung des Patienten „Y“?

---

[4] Atlas.ti ist eine QDA-Software und wird im nächsten Kapitel vorgestellt
[5] Im Kapitel II wird eine ausführliche Beschreibung von den zurzeit angewendeten Methoden und Werkzeugen zur Aufzeichnung von Mikroprozessen gegeben.
[6] Englisch: time series

- Was ist die durschnittliche Studiendauer an der Universität „X“.

Mit Hilfe der Visualisierung von zeitabhängigen Daten können folgende wichtige Fragen für deren Analyse beantwortet werden [ME95]:

- Ist einen Datensatz an einem bestimmten Zeitpunkt vorhanden?
- Wann existiert ein bestimmter Datensatz?
- Wie oft und wie lange tritt ein Datensatz über ein bestimmtes Zeitintervall auf?
- In welcher Reihenfolge treten bestimmte Datensätze auf?
- Kann die Präsenz von Mustern in der Existenz von Datensätzen festgestellt werden?

Alle zeitabhängigen Datensätze sind durch Anfangs- und Endzeitpunkte charakterisiert. Sie haben also eine bestimmte Dauer (wie bei Programmiertätigkeiten in dieser Diplomarbeit).

Lambert Wiesing beschreibt die Visualisierung, als *„etwas sichtbar machen, was nicht sichtbar ist“* [LW97]. In der Analyse von Programmiertätigkeiten wird das „Unsichtbare“ durch Verhaltensmuster von Programmierern repräsentiert. Diese Muster sind eventuell in den Daten versteckt.
Die Software, die im Rahmen dieser Diplomarbeit erstellt wird, ist ein sehr nützliches Werkzeug für die Erkennung der Verhaltensmuster durch Visualisierung

Motivation

Das Forschungsgebiet, in dem ein solches Werkzeug eingesetzt werden soll finde ich sehr interessant. Die Herausforderung einer implementierungsorientierten Arbeit, in der Kundenanforderungen grafisch umgesetzt werden müssen, empfand ich als besonders reizvoll. Ein „richtiges“ Softwareprojekt zum Abschluss meines Studiums zu führen, war für mich eine Chance, die ich auf keinen Fall verpassen wollte. Das Projekt ist sehr praxisorientiert. Es bietet mir die Gelegenheit einer kundenorierten Softwareentwicklung. Mein Betreuer (Herr Stephan Salinger) sowie ein zweiter Forscher der FUB (Herr Sebastian Jekutsch) fungierten in diesem Fall als Kunden. Die unterschiedlichen Datenerhebungsmethoden[7], generieren genauso viele unterschiedliche Inputdatenformate. Die hohe Generizität durch eine leichte Weiterentwicklung der Software im Hinblick auf die Inputdaten muss garantiert werden. Dieser Fakt stellte einen weitereren verlockenden Aspekt des Projekts dar.

Aus dieser letzten Anforderung leitete ich den Namen der Software ab:

Generic Time-dependent data **V**isualization (**Generic TV**).

[7] Die Datenerhebungsmethoden werden im Kapitel II vorgestellt.

I.2. Aufbau der Arbeit

Diese Diplomarbeit ist folgendermaßen gegliedert.

In Kapitel II wird auf die Problemstellung in einer zusammengefassten Form eingegangen. Zuerst wird die Bedeutung von Fehlern in der Softwareentwicklung vorgestellt. Danach erfolgt eine Einführung in das Forschungsgebiet, welches Generic TV unterstützen soll. Desweiteren wird in Kapitel II die Datenerhebung und deren resultierende Probleme bei der Analyse vorgestellt. Aus der Problemstellung ergeben sich die Anforderungen für Generic TV, welche im letzten Abschnitt des Kapitels ausgeführt werden.

In Kapitel III werden neue Begriffe wie z.B. ***Mikroprozess***, ***Annotation***, ***Spuren***. Vorgestellt. Durch die Begriffserklärung soll ein besseres Verständnis für das Einsatzsgebiet von Generic TV geschaffen werden. Ein Einblick in die Inputdaten der Software wird in diesem Kapitel auch stattfinden. Er beginnt mit ihrer Generierung und endet mi ihrer Darstellung durch Generic TV. Ein entscheidender Moment für die Realisierung eines Softwareprojekts ist die Suche und der Einsatz von wiederverwendbaren Komponenten. Mir war schon am Anfang der Arbeit bewusst, dass ich ohne den Einsatz eines gut geeigneten Frameworks keine zufriedenstellenden Ergebnisse bekommen würde. Die Vorstellung und die Motivation meiner Wahl für die Java Bibliothek JFreeChart werden in Kapitel III durchgeführt.

Die Generic TV ist ein Visualisierungswerkzeug. Bevor ich mit dieser Diplomarbeit anfing, konnte ich nicht ahnen, wie umfangreich und wichtig die Disziplin Visualisierung ist. Ich werde im Kapitel IV dieses Gebiet vorstellen. Da die Daten in einem Visualisierungswerkzeug sehr wichtig sind[8], werde ich im vierten Kapitel die Visualisierungsmöglichkeiten von Informations- und Multiparameterdaten darlegen.

In Kapitel V stelle ich den Softwareentwurf von Generic TV vor. Die Umsetzung von unterschiedlichen Darstellungstechniken wird erläutet und begründet. Der Arbeitsablauf der Software wird präsentiert. Die in Generic TV angewendetenAbstraktionsschichten werden in diesem Kapitel vorgestellt begründet.

Meine Ausarbeitung wird im Kapitel VI mit einer Zusammenfassung der geleisteten Arbeit abgeschlossen. In diesem Kapitel erfolgt ein persönliches Fazit. Danach werden angeratene Verbesserungen für das System vorgestellt.

[8] Sie entscheiden, welche Visualisierungsart man realisieren muss.

II. Problemstellung

In diesem Teil der Ausarbeitung wird die Aufgabenstellung der Diplomarbeit erläutet. Außerdem wird eine Zusammenfassung der Softwareanforderungen gegeben.
Bevor dies gemacht wird, ist eine kurze Einführung in das Forschungsgebiet notwendig. Diese Einführung beschreibt in einer zusammengefassten Form die Forschungsgegenstände und die Forschungsmethodik der Softwaretechnik-Forschungsgruppe (AG SE) der FUB.
Die bei der derzeitigen Datenauswertung entstanden Probleme haben zur Aufgabenstellung dieser Diplomarbeit beigetragen. Die Notwendigkeit des Einsatzes eines Visualisierungswerkzeugs für die Analyse der erhobenen Daten wird danach erläutert.

Mit größter Sorgfalt muss ein Softwareentwickler auf die Qualität seiner Software (SW) achten. Das Ziel jeder Softwareentwicklung sollte es sein, dass das Softwareprodukt hohe Qualitätseigenschaften besitzt. Die erreichten Qualitätseigenschaften hängen aber von vielen unterschiedlichen Faktoren ab. Als Beispiele wären die zur Verfügung stehende Zeit oder die vorhandenen Mittel (Geld, Personal, Fachwissen) bei der Erstellung von SW anzuführen.

Was sind aber Qualitätseigenschaften einer SW? Lutz Prechelt, Leiter der AG SE der FUB, unterscheidet bei Qualitätseigenschaften einer SW[PL06]:

- Externe Qualitätseigenschaften (aus Benutzersicht)

    a. **Benutzbarkeit**: Bedienbarkeit, Erlernbarkeit, Robustheit
    b. **Verlässlichkeit**: Zuverlässigkeit, Verfügbarkeit, Sicherheit, Schutz
    c. **Brauchbarkeit**: Angemessenheit, Geschwindigkeit, Skalierbarkeit, Pflege

- Interne Qualitätseigenschaften (aus Entwicklersicht)

    a. **Zuverlässigkeit**: Korrektheit, Robustheit, Verfügbarkeit
    b. **Wartbarkeit**: Verstehbarkeit, Änderbarkeit, Testbarkeit, Korrektheit, Robustheit
    c. **Effizienz**: Speichereffizienz, Laufzeiteffizienz, Skalierbarkeit

Für den Benutzer ist die Funktionalität enorm wichtig. Aus Entwicklersicht sind Zuverlässigkeit und Korrektheit die zentralen Qualitätsaspekte einer SW.

Die Frage, ob eine Software die an sie gestellten Aufgaben erfüllt, spricht das Problem der Softwaredefekte an. Was ist ein Softwaredefekt? Helmut Balzert definiert einen Softwaredefekt als falsches Verhalten eines Programms [BH98]. Der kritische Punkt bei dieser Definition ist, dass sie viel Raum für Spekulationen bzw. Diskussionen offen lässt. Harry Sneed liefert eine interessante Folge von möglichen Definitionen für Defekte in der Softwareentwicklung [HS_I]:

- Von einem Defekt kann nur gesprochen werden, wenn ein gegebener Sachverhalt von einer vereinbarten Norm abweicht.

- Ein Softwaredefekt kommt vor, wenn die eigentliche Leistung der Software von der Leistungsbeschreibung abweicht.

- Ein Softwaredefekt ist gegeben, wenn die Software sich anders verhält als in Dokumentationen bzw. Benutzerhandbüchern beschrieben.

- Ein Softwaredefekt ist vorhanden, wenn die Software nicht so funktioniert, wie der Benutzer es erwartet.

- Ein Softwaredefekt ist alles, was dem Benutzer nicht passt.

Ein Vergleich zwischen der ersten und der letzten Sneed-Definition zeigt, wie unterschiedlich die Meinungen über Softwaredefekte sein können. Unabhängig von der Interpretation bleibt aber festzuhalten, dass Defekte durch mindestens einen Fehler des Programmierers entstehen. Wichtig ist jedoch immer dabei, dass Defekte in der SW behoben werden müssen. Spätestens dann, wenn sie zum Versagen der SW führen. Das amerikanische IT-Magazin „Wired" [NZ_I] hat die zehn schlimmsten Softwareversagen aufgelistet. Manche haben Menschenleben gekostet (Nationales Krebs-Institut Panama City im Jahr 2000[9]) und andere haben Milliarden-Schäden verursacht.

Die Konfrontation mit solchen Folgen ist für die Softwareentwicklung in vielerlei Hinsicht herausfordernd. Daher lautet die erste Frage: Wie entstehen überhaupt Softwarefehler, die eventuell zu Versagen führen können?

[9] Krebspatienten wurden auf Grund eines Softwareberechnungsfehlers mit der doppelten Dosis bestrahlt. Mindestens 8 Personen starben. Die Ärzte wurden wegen Mordes angeklagt

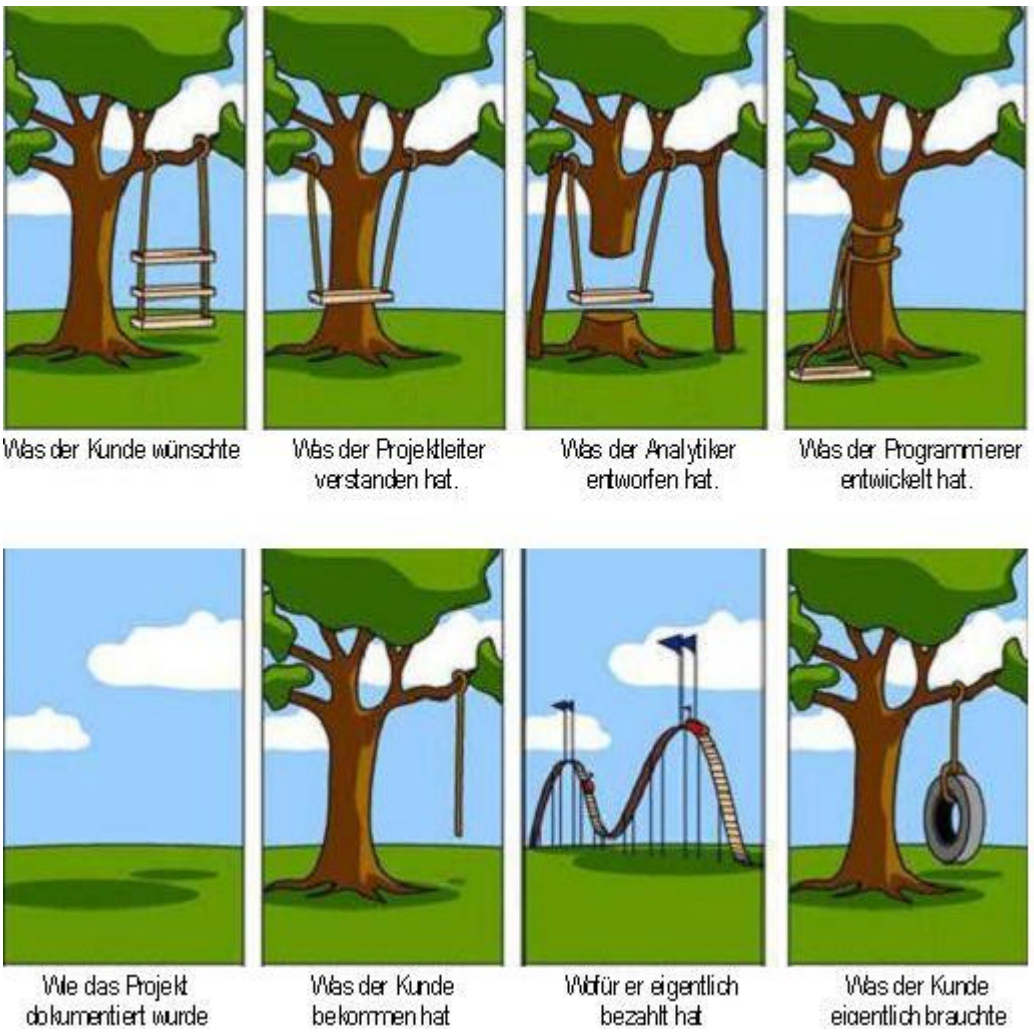


Abbildung 4 - Softwaretechnik

Durch die Abbildung 4 kann man erkennen, dass Softwaredefekte in allen Phasen eines Softwareprojekts erzeugt werden können. Der Auftraggeber, der vielleicht nicht genau weiß, was er braucht bzw. der seine Wünsche nicht äußern kann; der Informatiker, der die Wünsche des Auftraggebers falsch versteht oder der Entwickler, der den Entwurf fehlerhaft implementiert, könnten dafür Beispiele sein. Softwaredefekte habengemein, einmal abgesehen von ihrem Ursprung, dass sie enorme Kosten für ihre Behebung verursachen. Das deutsche Ministerium für Forschung und Bildung gab im Jahre 2001 eine Studie heraus [HS_I]. Diese Studie zeigt die enormen Kosten, die durch Softwarefehler verursachten werden:

- Wert Software Anwendungssysteme = 25 Milliarden Euro
  '
- Größe der Software Anwendungssysteme = 1,25 Milliarden Anweisungen
- Bei einer mittleren Fehlerrate von 3 Fehlern pro 1000 Anweisungen = 3,5 Millionen Fehler
- Eine Fehlerbehebung kostet im Durchschnitt 13 Stunden = Euro 650,-
- **Die Behebung 3,5 Millionen Fehler kostet Euro 2,4 Milliarden bzw. 9% des Wertes**

Helmut Balzert [BH98] klassifiziert die Kosten der Fehlerbehebung. Er setzt in Beziehung zur Softwareentwicklungsphasen. H. Balzert behauptet, dass die Behebung nach dem Einsatz der Software 200-mal teurer ist, als eine Behebung in der Analyse- oder Designphase.

Würden alle Softwaredefekte in den frühen Phasen des Entwicklungsprozesses korrigiert, gäbe es keine hohen Folgekosten. Daraus folgt, dass es keinen Forschungsbedarf in dieser Hinsicht gäbe.

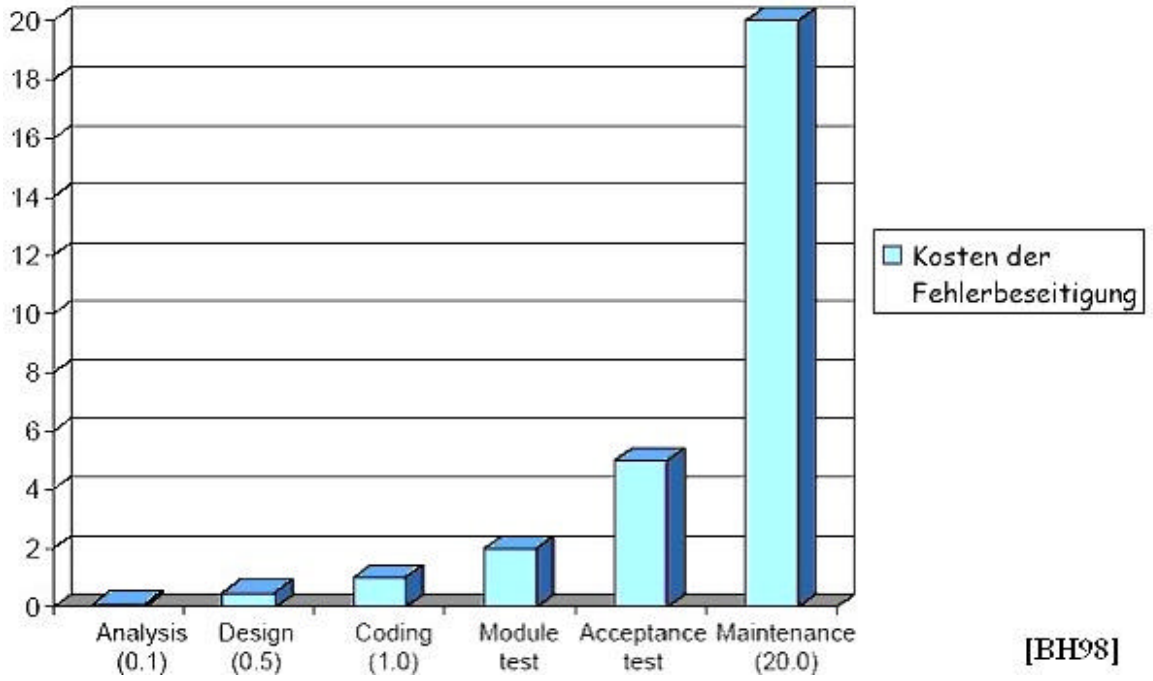


Abbildung 5 - Kosten der Fehlerbeseitigung in der SW-Entwicklungsphase

Die Forschungsgebiete

Das Ziel dieser Diplomarbeit ist keine Analyse der Entstehung oder Behebung von Defekten in der Softwareentwicklung. Generic TV ist aber eine Software, die unter anderem[10] das Forschungsgebiet „Vermeidung von Programmierfehlern“ unterstützen soll. Aus diesem Grund war eine kurze Einleitung über Softwarefehler, -defekte und –versagen nötig.

Eine der ersten Fragen, die sich mir bei der Erarbeitung des Themas stellten, war nach dem Sinn des Erfassens der Programmiertätigkeiten von Entwicklern.

Der Grund der Erfassung der Programmiertätigkeiten ist unter anderem festzustellen, unter welchen Voraussetzungen Programmierfehler entsehen. Eines dieser Forschungsprojekte beschäftigt sich mit dem Versuch, Fehler bei der Softwareentwicklung zu vermeiden. Je früher man Programmierfehler erkennt, desto besser. So könnte man kurz und prägnant den Ansatz dieses Forschungsprojekts charakterisieren.

Wie am Anfang dieses Kapitels erwähnt wurde, führen Fehler zu Defekten. Durch Defekte kann es zum Versagen einer SW kommen. Die Qualitätssicherung (QS) bei der Softwareentwicklung fasst alle Maßnahmen zur Gewährleistung einer guten Qualität des Softwareprodukts zusammen.

Die analytische QS ist als wertvoll in der Softwareentwicklung einzustufen. Sie ist für die Vermeidung von Programmierfehlern ist nicht geeignet, weil sie fertige Softwaremodule auf Qualität testet. Die Fehler, welche beim Programmieren entstehen, werden erst im Nachhinein. Das Ergebnis einer analytischen QS ist eine Besserung des Produkts durch das nachträgliche Aufinden von eventuellen Mängeln. Im Gegensatz dazu, ergreift eine konstruktive QS Vorkehrungen, damit Qualitätsmängel erst gar nicht entstehen. Sie beseitigt bei entdeckten Mängeln nicht nur den Mangel selbst, sondern auch seine Ursache(n) und ggf. deren Ursache(n)[PL06].

Das Forschungsgebiet „Vermeidung von Programmierfehlern“ versucht ein Konzept für eine konstruktive QS zu erstellen. Das Ziel ist es, Softwarefehler dort abzufangen und dort zu beheben, wo sie entstehen[11]. Was ist aber ein Programmierfehler? Sebastian Jekutsch, der sich mit dem Forschungsgebiet beschäftigt, definiert einen Programmierfehler als ein „Falschtun“ des Programmierers[JS06]. So ein „Falschtun“ kann, genauso wie die Definition des Defekts, auf unterschiedliche Art und Weise interpretiert werden:

1. Ein Programmier kann zum Beispiel einen Entwurf schlecht interpretieren und dabei eine Methode der Klassendiagramme falsch implementieren ➔ **Er macht einen Fehler**.

2. Ein Programmierer kann durch Müdigkeit abgelenkt werden. Die Aufmerksamkeit ist nicht mehr ausreichend, um Fehler im Code wahrzunehmen bzw. sofort zu erkennen ➔ **Er macht einen Fehler**.

[10] Generic TV unterstützt weitere Gebiete, wie die „Untersuchung der Arbeitsweise von Paarprogrammierern“. Ich konnte aber in dieser Ausarbeitung leider nicht alle unterstützen Gebiete vorstellen.
[11] d.h. in der Programmierungsphase

In den beiden Beispielen hat ein Programmierer einen Fehler gemacht. Die Ursachen, die zu den Fehlern führten, sind aber unterschiedlicher Natur. Aus diesem Grund hat Sebastian Jekutsch die Fehler in der Softwareentwicklung nach Ursachen klassifiziert[JS06]:

- **Irrtum**: *Der Programmierer meint, das richtige Wissen zu haben, aber es ist falsch. In voller Überzeugung macht er einen Programmierfehler. Diese Fehler werden Irrtumsfehler genannt. Beispiel: Die Bedeutung eines Methodenparameters wird missverstanden.*

- **Beanspruchung**: *Der Programmierer ist während der Arbeit überfordert, welches ihn zu Fehlern aller Art provoziert. Bessere Umstände und reichlich Zeit hätten durchaus Fehler verhindern können. Das Wissen ist also prinzipiell vorhanden, nur die Anwendung des Wissens oder die Durchführung dieser Anwendung war falsch. Beispiel: Überforderung oder Ablenkung.*

Verhaltensmuster und Programmierfehlerverhaltensmuster

Kann der Programmier seinen Fehler **selbst** im Nachhinein erkennen? Diese Frage entscheidet, ob eine konstruktive QS in der Lage wäre, den Fehler zu lokalisieren. Der Irrtumsfehler ist nur durch jemanden erkennbar, der das Richtige[12] kennt. Dies bedeutet, dass der Programmierer hat etwas Falsches gedacht und dadurch etwas Falsches getan. Keine QS wäre in der Lage den entstehenden Fehler zu erkennen. Es sei denn, sie kenne das Richtige im Voraus. Aus diesem Grund werden Irrtumsfehler bei der Forschung, die Generic TV unterstützen soll, nicht berücksichtigt.

Beanspruchungsfehler sind viel interessanter als Irrtumsfehler. Sebastian Jekutsch zentrale Hypothese über ihr Entstehen lautet wie folgt [JS06]:

- **Beim Programmieren machen sich Beanspruchungsfehlersituationen durch auffällige (evtl. persönliche) Verhaltensweisen bemerkbar.**

Aus dieser Hypothese ergeben sich zum Beispiel folgende Fragen:

- Gibt es erkennbare Umstände, die den Entwickler Defekte erzeugen lassen?

- Kann man die Entwickler während der Implementierung unterstützen (durch z.B. automatisierte Prozesse), um Fehler zu vermeiden, indem solche Umstände rechtzeitig erkannt werden?

In seiner Arbeit vertritt Sebastian Jekutsch eine zweite Hypothese, die auf das unterschiedliche Verhalten eines Programmierers zielt. Es gibt kein „universelles“ Verhalten beim Programmieren. Die Verhaltensweisen unterscheiden sich individuell (Programmierer A hört mit Kopfhörern Musik bei der Arbeit; Programmierer B nicht könnte unter solchen Bedingungen arbeiten). Interessant sind auffällige Verhaltensänderungen bei Programmierern, die laut Jekutsch mindestens einen Grund haben. Dieser Grund könnte eine veränderte Beanspruchung sein. (Programmierer B programmiert jetzt mit Musik ➔ Er ist abgelenkt)

Diese Verhaltensänderungen eines Programmierers hat Jekutsch folgendermaßen beschrieben:

[12] Welcher an dieser Stelle etwas anderes gedacht bzw. gemacht hätte.

- **Verhaltensweisen beim Programmieren existieren und sind auch so genug stereotypisch, dass man von wieder erkennbaren Programmierfehlerverhaltens-mustern reden kann.**

Das Festlegen bzw. die Suche von Programmierfehlerverhaltensmustern ist eine der Hauptanforderungen des Forschungsgebiets „Vermeidung von Programmierfehlern“ der FUB. Sobald solche Muster identifizierbar, bestimmbar und spezifizierbar sind, wäre die Warnung des Programmiers möglich. Dies würde bei der Erkennung von festgelegten Programmierfehlerverhaltensmustern erfolgen.

Ein Beispiel wäre, dass sich stets nach mehr als sechs Stunden Programmierens die Fehlerquote von Herrn X stark erhöht (Programmierfehlerverhaltensmuster). Deshalb erfolgt nach fünf Stunden ein Pausesignal auf dem Bildschrim

Der Mikroprozess[13]

Bevor man in der Lage ist, Programmierfehlerverhaltensmuster zu identifizieren, muss man zuerst ein Verhaltensmuster beschreiben können. Hierfür hat Jekutsch den Begriff des „Mikroprozesses des Programmierens“ (µPP) eingeführt. Ein µPP beschreibt die einzelnen Tätigkeiten (z.B. Codeänderungen, Testen, Nachschlagen, Fragen usw.) des Programmierers über ein bestimmtes Zeitintervall. Der µPP kann von außen beobachtet werden. Es kann als Ereignisfolge automatisch aufgezeichnet werden. Dies kann in Form einzelner Arbeitsschritte, z.B. bei einer Entwicklungsumgebung, geschehen[JS06].

µPP ist ein wesentlicher Teil der durch Generic TV darzustellenden Datensätze. Welche Möglichkeiten sind aber vorhanden, um aus µPP einen Dateninput für Generic TV zu generieren?

Erhobene Daten und dafür notwendige Werkzeuge

In der Einleitung wurden die darzustellenden Daten als zeitabhängige Daten präsentiert. Die Programmiertätigkeiten können derzeit auf unterschiedliche Weise erfasst werden. Die Konsistenz der Daten wird ausführlich im nächsten Kapitel vorgestellt. In diesem Kapitel wird deshalb ein (von derzeit zwei) Datenerhebungsverfahren vorgestellt.

Die detaillierte Beobachtung der Programmiertätigkeiten liefert die Rohdaten. Diese Beobachtungen wurden unter anderem durch eine synchrone Aufzeichnung der Aktivitäten der Entwickler möglich. Dabei wurden die Programmierer in Bild und Ton aufgezeichnet. Synchron erfolgte die Aufnahme ihrer aktuellen Arbeit[14]. Aus diesen Aufzeichnungen entstehen Beobachtungsvideodateien, bei der das Verhalten der Programmierer und ihre Arbeit zu sehen sind (Abbildung 6).

[13] Auf dem Begriff wird in einer Ausführlicherweise im nächsten Kapitel eingegangen
[14] D.h. der Desktop der Computer

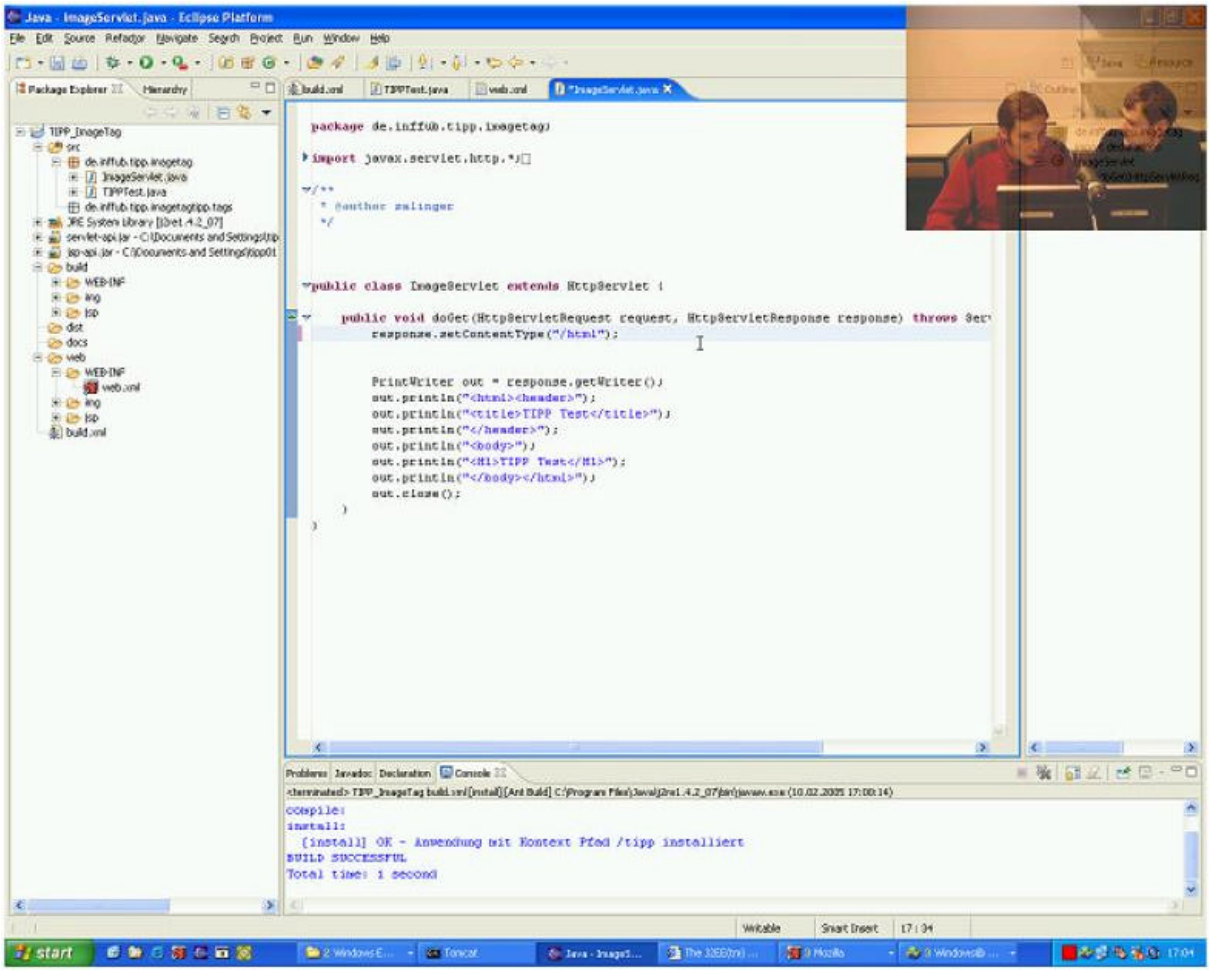


Abbildung 6 - Aufbau einer Beobachtungsvideodatei [SS06]

Die Videosequenzen[15] sind das Datenmaterial für qualitative Analysen. Die angewendete Forschungsmethodik basiert auf einem besonderen Forschungsstil, welcher Grounded Theory (GT) genannt wird. Die GT bezeichnet einen sozialwissenschaftlichen Ansatz zur systematischen Auswertung von meist qualitativen Daten (z.B. Beobachtungsprotokollen) mit dem Ziel der Theoriegenerierung [GS67]. In diesem Zusammenhang beschreibt Stephan Salinger GT als eine Methodik zur Entwicklung einer datenverankerten Theorie [SS06]. GT besteht aus einer Vielzahl von Methodiken, die im Rahmen dieser Ausarbeitung nicht weiter ausgeführt werden müssen. Der wichtigste Punkt für Datenerhebungsverfahren (durch Videosequenzen) ist **das Kodieren**.

Das Kodierverfahren innerhalb von GT beschreibt die Bildung von ***Konzepten/Kategorien*** und die Zuordnung von Daten [SS06]. Das Kodieren der Videosequenzen entspricht konzeptionell dem Beschreiben von Labels[16] an bestimmten Stellen der Videosequenzen. Das Anhängen von Labels an bestimmten Stellen der Videosequenzen ist eine konkrete Zuordnung eines Konzepts zu einem Abschnitt. Der Zweck des Kodierverfahrens ist eine Objektdefinition. Mit Hilfe des Kodierens können weitere Objekte erzeugt werden. Der Prozess der Zuordnung von Konzepten zu Abschnitten wird im Rahmen dieser Arbeit ***Annotieren*** genannt. Die Zuordnung als solche ist eine ***Annotation***. Das Verhalten der Programmierer (bzw.

[15] innerhalb dieser Arbeit auch Abschnitt genannt
[16] z.B. „Dokumentation lesen“, „Hilfe suchen“

die ausgeübten Tätigkeiten[17]) werden im Datenmaterial durch das Kodieren identifiziert. Jedes identifizierte Phänomen wird mit einem, evtl. neu zu definierenden, Konzept annotiert.

Um das Kodieren einer Videosequenz zu ermöglichen, wird das Werkzeug **Atlas.ti** verwendet. Atlas.ti ist eine so genannte Qualitative-Data-Analysis-, kurz QDA-, Software. Sie ermöglicht die Verarbeitung von Text-, Grafik-, Audio- und Videodaten. Atlas.ti ist in Lage Videosequenzen wiederzugeben. An sie kann der Benutzer beschreibende Labels (Konzepte) „kleben". Die üblichen Arbeitsschritte bei der Datenanalyse mit Atlas.ti sehen folgendermaßen aus:

- Definieren eines Abschnitts im Videomaterial;
- Annotieren des Abschnitts mit Konzepten;
- Eventuelles Verlinken des Abschnittes mit anderen Abschnitten.

Die Visualisierung der durchgeführten Arbeiten in Atlas.ti ist ausschließlich über sogenannte Networkviews möglich (Abbildung 7).

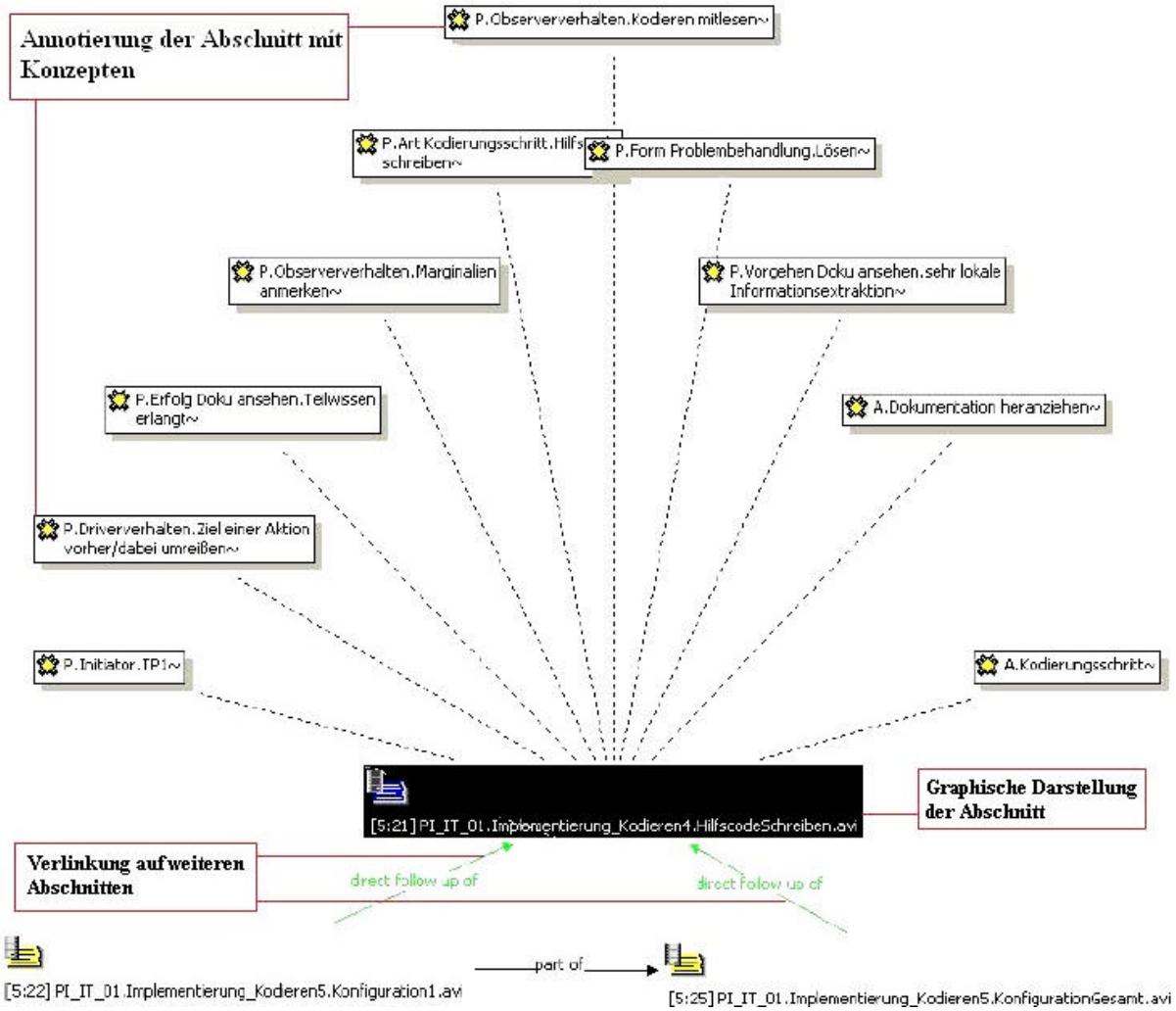


Abbildung 7 - Networkviews von annotierten Abschnitten in Atlas.ti [SS06]

[17] auch ***Phänomene*** genannt

Alle Aktivitäten der Entwickler sind wichtig. Damit sind auch Aktionen und Einflussfaktoren gemeint, die nicht unmittelbar mit Programmiertätigkeiten in Verbindung stehen. Hierfür können viele Beispiele aufgezählt werden: Jede Art von Unterbrechungen (Besuch, Telefon), Musikhören während der Arbeit oder gewisse Stresssituationen (Krankheit, persönliche Probleme). Die QDA-Software Atlas.ti ermöglicht ein komfortables Annotieren durch Konzepte der Videosequenzen. Sie hat aber für dieses Forschungsprojekt einige Minuspunkte. Die Benutzung der Networkviews ist sehr ungeeignet für die qualitative Analyse der Daten. Folgende Probleme treten auf:

- Die aktuellen Sichtweisen auf die realisierten Beobachtungen (Networkviews) geben keine Informationen über die zeitliche und inhaltliche Verteilung von Programmiertätigkeiten. Obwohl jede Programmiertätigkeit einen zeitlichen Verlauf besitzt.

- Durch die Networkviews sind keine zeitabhängigen Muster erkennbar.

- Es ist keine Visualisierung von gefilterten[18] Informationen über einen gewissen Zeitraum möglich. Zum Beispiel ist es nicht möglich, in einem Abschnitt nur die reinen Programmiertätigkeiten zu visualisieren. Daraus folgt, dass die Datenmenge sich nicht verringert.

Aus diesen Schwierigkeiten hat sich die Aufgabenstellung dieser Diplomarbeit ergeben. Die den Forschern der FUB gestellten Anforderungen an Atlas.ti zur Visualisierung der Programmiertätigkeiten werden nicht erfüllt. Atlas.ti ist aber in der Lage die Inputdaten für Generic TV bereitzustellen. Eine Funktionalität von Atlas.ti ist die Möglichkeit, eine oder mehrere Videosequenzen mit den durchgeführten Annotationen zu exportieren. Dieser Datenexport erfolgt im Dateiformat XML- oder Text. Die Datei(en) enthalten alle Informationen, die in der Videosequenz vorhanden waren, nämlich:

- Die durchgeführte Kodierung mit genauer Zeitangabe der Programmiertätigkeit;

- Die Relationen zu anderen Abschnitten;

- Kommentare etc.

Im Kapitel III wird ein weiteres Datenerhebungsverfahren vorgestellt.

Anforderungen an Generic TV[19]

In der Einleitung dieser Arbeit wurden die Aufgaben einer Visualisierung vorgestellt. Zwei von diesen Aufgaben waren:

- Eine Visualisierung vereinfacht den Zugang zu Massendaten, z.B. durch Klassifikation und Datenstrukturierung.

- Eine Visualisierung hilft bei der Analyse und Interpretation von Daten, z.B. durch Aufzeigen verborgener Trends oder bei der Erkennung von Mustern.

[18] auf bestimmte Interessenschwerpunkte
[19] Anhang A und B stellen eine detaillierte Liste der Anforderungen an Generic TV vor.

Eine Informations-Visualisierung der erhobenen Daten würde einen strukturierten Zugriff auf sie ermöglichen. Ein strukturierter Zugriff für die Forscher der FUB wäre, die Möglichkeit der Datendarstellung mit unterschiedlichen Ansichten durch Filterung des Datenbestands. Dadurch würden sich Programmierfehlerverhaltensmuster eventuell besser erkennen lassen.

Die Forscher der FUB sind derzeit in Besitz von Massendaten unterschiedlichster Art:

1. Annotierte Videosequenzen von Programmiertätigkeiten von Entwicklern,

2. Durch ECG generierte Dateien von bestimmten protokollierten Mikroprozessen.

Generic TV soll folgende Leistungen erbringen:

1. Zeitabhängige Visualisierung der Programmiertätigkeiten durch grafische Elemente,

2. Eine quantitative Darstellung[20] bestimmter Datensätze,

3. Benutzerinteraktion ermöglichen.

Zu 1. Jede beobachtete bzw. aufgezeichnete Programmiertätigkeit mit eindeutiger Anfangs- (und eventuell Endzeitpunkt) muss in der Visualisierung eindeutig erkennbar sein. Die grafische Darstellung vom Auftreten von Programmiertätigkeiten wird in dieser Arbeit ***Spur*** genannt. Generic TV muss also in der Lage sein, eine grafische Darstellung der Spuren zu realisieren.

Zu 2. Manche beobachtete Programmiertätigkeiten besitzen Eigenschaften, die wiederum Ausprägungen haben. Die Eigenschaften charakterisieren die beobachteten Tätigkeiten. Die Ausprägungen sind Werte dieser Eigenschaften. Zum Beispiel die Programmiertätigkeiten des Entwicklers Herr Muster sind beobachtet worden. Eine dieser Tätigkeit ist das „Kodieren“. Eine Eigenschaft vom „Kodieren“ ist „Geschwindigkeit“. Diese Eigenschaft besitzt den Wert „Schnell“ (Siehe Tabelle 1). Generic TV muss eine Visualisierung der Ausprägungen dieser Eigenschaften dargestellen

Beobachtete Tätigkeit	Eigenschaft	Ausprägung
Kodieren	Geschwindigkeit	Schnell
Dokumentation heranziehen	Zielsetzung	Inspiration

Tabelle 1 - Tätigkeit - Eigenschaft - Ausprägung

Zu 3. Dem Benutzer soll:

a. Die Spurbildung beeinflussen. Er muss entscheiden können, welche Programmiertätigkeiten er sehen will.

b. Die Sicht in den dargestellten Spuren durch Interaktivitäten muss veränderbar sein. Z.B. durch Filtern, die Anwendung unterschiedlicher Farben für die dar-

[20] Dabei werden die Werte der Datensätze durch visuelle Variablen dargestellt. Ausführlich wird dies in Kapitel IV vorgestellt

gestellten Programmiertätigkeiten, Ein- bzw. Ausblenden bestimmter Informationen.

Die genaue Ermittlung der Anforderungen ist ein wichtiger Teil der Ausführung eines solchen Softwareprojekts. Nach der Analyse der vorgeschriebenen[21] und der von mir erhobenen Anforderungen, hat sich ein Anforderungsdokument (Pflichtenheft)[22] ergeben. Es übertraf alle Erwartungen bei weitem.

Meine bisher erworbenen Praxiserfahrungen in der Welt der Softwareentwicklung haben mich das Folgende gelehrt:

- Informatiker müssen gedanklich flexibel sein.
- Die Kommunikation mit den Anwendern des Systems unerlässlich.

Informatiker sind in der Regel nicht mit dem Fachgebiet des Anwenders vetraut. Aus diesem Grund sollte ihre Stärke darin bestehen, sich kurzfristig in die wesentlichen Inhalte des Anwenders einarbeiten zu können. Sie müssen bei allen neuen Softwareprojekten eine neue Sprache lernen. Nämlich die Sprache des Anwenders. Sie müssen diese beherrschen, um eine effektive Anforderungsliste erstellen zu können. Nach dieser Prämisse handelte ich bei der Erstellung der Anforderungsliste von Generic TV.

Bei der Erstellung des Pflichtenheftes traten folgende Problemstellungen auf:

1. Die Generizität der Software,

2. Die Auswahl der grafischen Elemente für die Darstellung von Programmiertätigkeiten,

3. Die Manipulation der durch Generic TV erzeugten Darstellung,

4. Die Entwicklung des Benutzerinteraktionstools.

Zu 1. Dies ist ein sehr wichtiger Punkt, weil die Beobachtungsarten der Programmiertätigkeiten der Entwickler erweiterbar sein könnten. Denn neue Beobachtungsarten könnten eventuell neue Formate von Inputdaten generieren. Die Visualisierung der neuen Daten muss durch einfache Erweiterbarkeit der Software möglich sein.

Zu 2. Generic TV muss die Programmiertätigkeiten in einer Benutzeroberfläche (BOF) darstellen. Welche grafischen Elemente sind für Realisierung dieser Darstellung geeignet?

Zu 3. Diese Manipulationen werden ***Benutzerinteraktionen*** oder ***Benutzerinteraktivität*** genannt. Zu den Aufgaben zählen unter anderem:

a. Das Filtern der angezeigten Programmieraktivitäten durch eindeutige Filterparameter.

[21] Siehe Anhang A
[22] Siehe Anhang B

b. Das Anzeigen von bestimmten Eigenschaften der Programmiertätigkeiten, wie z.B. Entwicklernamen oder Dauer.

Zu 4. Sie besteht aus zwei Schritten:

a. Die Entwicklung eines Konzepts, um die Benutzerinteraktion zu ermöglichen. Folgende Fragestellungen treten z.B. auf: Wie kann der Benutzer die Farbe der dargestellten Programmiertätigkeit dynamisch ändern? Welche Voraussetzungen müssen dafür vorhanden sein? Welche Konsequenzen hat eine solche Veränderung der Farbe auf die restliche Darstellung?

b. Der Entwurf und die Implementierung eines Werkzeuges für die Durchführung der Benutzerinteraktion. Die gute Tool-Handhabbarkeit ist wichtig.

Ein entscheidender Schritt war meine Suche nach wiederverwendbaren Komponenten (Frameworks). Dies ist eine häufig auftretende Schwierigkeit bei der Softwareentwicklung. Der Einsatz von Frameworks in der Softwareentwicklung hat Vorteile. Zu diesen Vorteilen gehören unter anderem die schnellere Entwicklungszeit, die hohe Qualität und die leichte Wart- und Änderbarkeit bei neuen Systemen. Deshalb war die Benutzung eines „geeigneten“ Frameworks für die Implementierung von Generic TV wichtig. Die Gründe, die diese Aufgabe erschwerten waren sehr unterschiedlich. Für diesen Teil der Arbeit habe ich zuerst einen falschen Ansatz gewählt. Ich suchte Software bzw. Frameworks, welche die Darstellung von zeitabhängigen Ereignissen ermöglichen. Dies ist zwar, was Generic TV erreichen soll, aber die Aufgaben, die mit Generic TV gelöst werden sollen, dürfen bei der Suche nicht vergessen werden. Durch eine systematische Herangehensweise änderten sich meine Suchkriterien für ein Framework. Dabei machten sich einige Fragen bemerkbar:

- Wie möchte ich die Programmiertätigkeit darstellen?
- Welches grafische Element möchte ich dafür benutzen?
- Welche Benutzerinteraktion möchte ich nicht persönlich implementieren?

Die Beantwortung dieser Fragen steuerte die Suche besser. Nachdem ich diese Fragen geklärt hatte, fielen mir immer mehr Lösungsansätze ein. Ich entschied, z.B. dass meine Software Text und Grafiken darstellen können sollte. Dabei definierte ich Grafiken als Symbole, Icons und geometrische Figuren (Rechtecke, Ellipsen, Quadrate). Die unterschiedlichen Programmiertätigkeiten werden durch diese Grafiken visualisiert. Die Grafiken werden über eine Zeitachse dargestellt, damit man die Anfangs- und Endzeitpunkte erkennen kann. Die Abbildung 8 verdeutlicht meinen Ansatz.

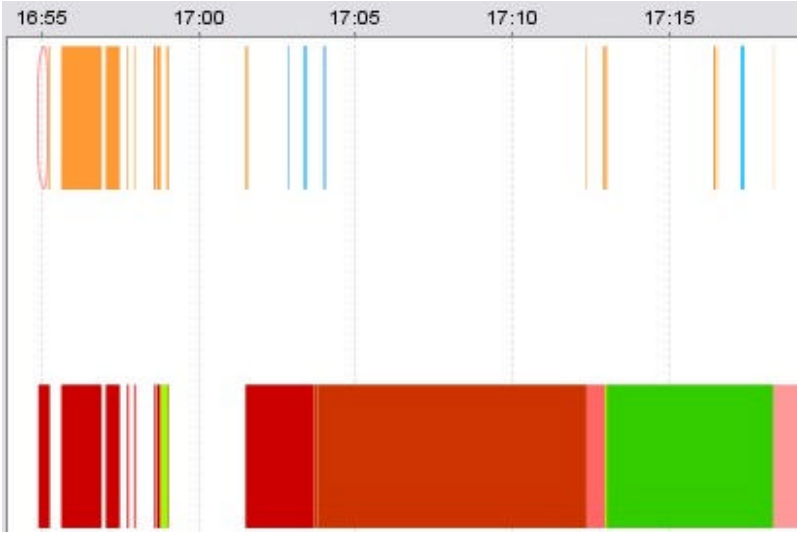


Abbildung 8 - Mögliches Aussehen von dargestellten Programmiertätigkeiten durch Generic TV

Die Kernaufgabe des gesuchten Frameworks war die zeitabhängige Darstellung von mir definierter Grafiken.

Durch die genauere Beschreibung der Anforderungen wurde die „Suche eines geeignetes Frameworks“ verfeinert. Aus diesem Prozess ergab sich eine Liste von Frameworks, die in Frage kämen. Der zweite Schritt meiner Frameworksuche, war jedes einzelne Framework bezüglich wichtiger (für Generic TV) Parameter zu „untersuchen“:

- Erfüllt es die gewünschten Benutzerinteraktionen bzw. Systemanforderungen? Wenn ja, welche?
- Wie sind die Schnittstellen mit der „Außenwelt“ umgesetzt?
- Wie ist die Dokumentation des Frameworks?
- Ist es erweiter- bzw. veränderbar?
- Darf ich es benutzen?
- Wie zukunftssicher ist der Framework?
- Wie stabil ist der Framework?

Antworten auf diese Fragen folgen in Kapitel III. Dort wird bei der Vorstellung des Frameworks ausführlich darauf eingegangen.

III. Die Akteure

In diesem Kapitel werden die wichtigen „Akteure“ dieser Arbeit vorgestellt. Es wird eine ausführliche Beschreibung von Programmiertätigkeiten gegeben. Der Begriff des Mikroprozesses des Programmierens (µPP) wird erläutert. In zweiten Teil dieses Kapitels wird der aktuelle Datenbestand, also die Inputdaten der Software, vorgestellt.

In Kapitel II erfolgte eine kleine Einführung über die Entstehung von Dateien[23] mit Atlas.ti. Hier werden diese Dateien ausführlicher beschrieben. Das heißt, ihre Struktur wird erläutert. Außerdem wird die Datenerhebung durch die ECG-Software. Den letzten Teil des Kapitels nimmt die Vorstellung eines sehr wichtigen „Akteurs“ der Entwicklung von Generic TV ein. Denn hier werden das ausgewählte Framework und vor allem die Begründung dieser Auswahl beschrieben.

III.1. Der Mikroprozess des Programmierens

Der Mikroprozess des Programmierens beschreibt die einzelnen Tätigkeiten (z.B. Codeänderungen, Testen, Nachschlagen, Fragen) des Programmierers während eines bestimmten Zeitintervalls. Die Bewegungen oder Tätigkeiten des Programmierers im Arbeitsprozess sind als µPP zu werten. Man sollte aber nach der Relevanz für die Forschung selektieren. Das Niesen des Programmierers ist zum Beispiel eine Aktivität. Diese Aktion ist für die Arbeit des Programmierers irrelevant. Es gibt sogar Aktivitäten, die für die Arbeit des Programmierers unabdingbar sind, die aber bedeutungslos für den µPP sind. Die Tatsache, dass der Programmierer atmet ist eine lebensnotwendige Aktion für seine Arbeit. Sie ist aber vollkommen irrelevant für den µPP [SF05]. Deshalb wird nur eine relevante Aktivität des Programmierers im Rahmen der Forschungen der FUB als ***Ereignis*** bezeichnet. Somit ergibt sich eine eindeutigere Charakterisierung des µPP, nämlich als eine Abfolge von „relevanten“ Ereignissen. Ein Beispiel für eine solche Abfolge von Ereignissen in einem Entwicklungsumgebungswerkzeug[24] wäre:

(1) Fenster öffnen,
(2) Datei ändern,
(3) Datei ändern,
(4) Programm ausführen,
(5) Fenster schließen.

Eine Haupteigenschaft dieser Ereignisse ist, dass einige in Beziehung zueinander stehen. Diese Beziehung könnte man wie folgt beschreiben: Eine bestimmte Folge von Ereignissen bildet ein bestimmtes Muster für eine bestimmte Tätigkeit des Programmierers. S. Jekutsch nennt solche Folgen von Ereignissen ***Episoden*** [JS06]. Im oben erwähnten Beispiel existiert eine klare Beziehung zwischen „Fenster öffnen“ und „Fenster schließen“. Alle Ereignisse zwischen diesen beiden Tätigkeiten gehören zu der Episode „Fileactive“. Es gibt weitere von Jekutsch exakt definierte Episoden. Ein Beispiel dafür wäre eine „Copy-Paste-Change“-Episode, bei der ein Codeblock kopiert und an einer anderen Stelle hinzugefügt wird. Die hier definierten Mikroprozesse werden im Rahmen dieser Arbeit mit dem in Kapitel II definierten Konzept gleichgestellt. Wie vorher erläutert, entsteht ein Konzept durch das Kodierensverfahren der Grounded Theory (GT). Stephan Salinger definiert ein Konzept als eine Bezeichnung

[23] die alle annotierten Programmiertätigkeiten enthalten
[24] z.B. Eclipse

bzw. ein Etikett, dem einzelne Ereignisse, Vorkommnisse oder andere Phänomene zugeordnet werden [SS06]. Diese Begriffsfestlegung erlaubt die Identifikation[25] des Verhaltens des Programmierers. So wie der Mikroprozess als Abfolge von „relevanten“ Ereignissen definiert worden ist, kann man durch Analogie ihn als Abfolge von „relevanten“ Konzepten definieren. Die Begriffe Mikroprozesse, Ereignis, Konzept werden alle für die Beschreibung von Verhalten der Programmierer angewendet.

III.2. Die Inputdaten

Die Inputdaten einer Visualisierung sind abstrakte Daten. Von ihnen soll eine Darstellung erstellt werden Der Zeitbezug der Daten entsteht durch die (zeitabhängigen) Tätigkeiten der Entwickler. Diese Tätigkeiten haben Anfangs- und Endzeitpunkte. Die aktuellen Datenerhebungsverfahren (durch Atlas.ti und ECG) generieren unterschiedliche Dateiarten (unterschiedliche Formate). Beide enthalten.die zu visualisierende Programmiertätigkeiten Generic TV muss auf verschiedene (aber von ihr bekannte) Dateiarten zugreifen und sie darstellen können. Das in Abbildung 9 dargestellte Metamodell fasst den Entwurf der generischen Schnittstelle von Generic TV zusammen. Dieser Entwurf wurde vom meinem Betreuer Stephan Salinger vorgeschlagenen. Das Metamodell stellt alle für die Visualisierung relevanten Objekte dar.

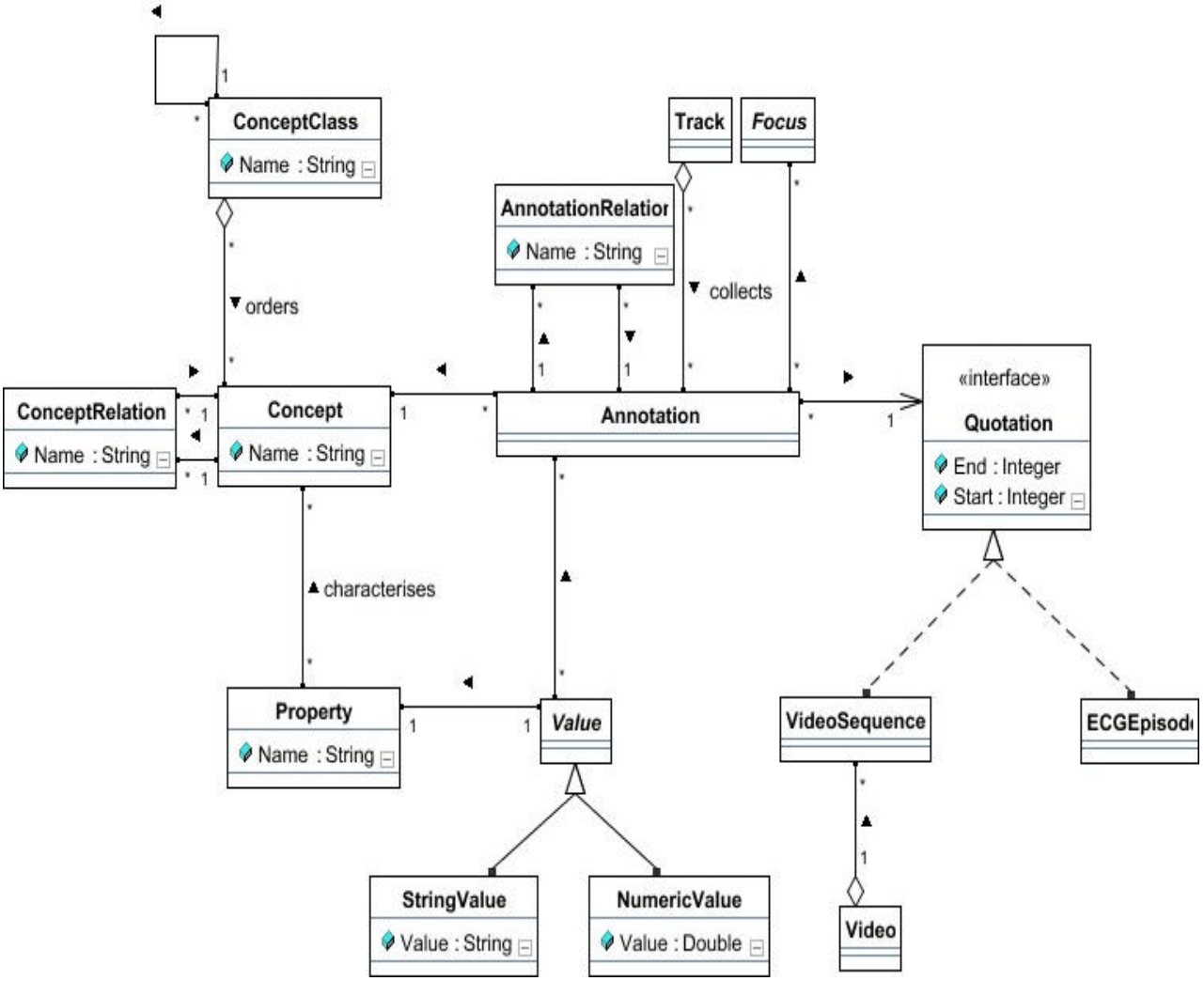


Abbildung 9 - Metamodell der generischen Schnittstelle von Generic TV

[25] durch Kodieren und Erzeugen von Konzepten

Kurze Erläutung des Metamodells

- **Quotation**: Die Begriffe Episode und Abschnitt (Videosequenz) wurden schon eingeführt. Beide enthalten Beschreibungen von Programmiertätigkeiten. Der Episode enthält eine Abfolge von Ereignissen. Der Abschnitt beinhaltet eine Abfolge bzw. Anzahl von annotierten Konzepten. Die Atlas.ti-Abschnitte und ECG-Episoden werden durch das Objekt Quotation modelliert. Eine der Eigenschaften des Objekts ist seine Anfangs- und Endzeitpunkt.

- **Annotation**: Eine Quotation besteht aus einer oder mehreren Annotation(en). Eine Annotation ist der Oberbegriff für die Modellierung der zu visualisierenden Elemente. Der Begriff Spur wurde als die grafische Darstellung der Programmiertätigkeiten definiert.

- **Track**: Sind Objekte, die aus Annotationen bestehen. Sie sind die Spuren in der Darstellung.

- **Concept**: Stellt die Konzepte bzw. Ereignisse dar.

- **ConceptClass**: Definiert die Möglichkeit, mehrere Konzepte zu gruppieren.

- **Property**: Repräsentieren die Eigenschaften von Konzepten, die wiederum eine Ausprägung besitzen

- **Value**: Ist die Oberklasse der Ausprägungen der Eigenschaften. Die wird im mehreren Klassen abgeleitet je nach dem, welche Datenyp die Ausprägung besitzt :

    a. **StringValue**
    b. **NumericValue**
    c. **BoolValue**

- **Focus**: Definiert den Ort, an dem das „Concept“ einer Annotation durchgeführt worden ist. Z.B. Concept „Datei ändern“ oder Value „Schnell Kodieren“ haben im Code stattgefunden.

- **ConceptRelation und AnnotationRelation**: Sie modellieren die möglichen Beziehungen zwischen Annotationen (z.B. Annotation B folgt Annotation A) und Konzepten (Konzepte C1 ist eine Folge von Konzept C5)

Dieses Metamodell wurde entworfen, um eine Schnittstelle zwischen den unterschiedlichen Ansätzen der Forschung der FUB und den resultierenden Elementen zu realisieren. Ohne eine gemeinsame Schnittstelle, wäre die Darstellung dieser Elemente durch ein einziges Werkzeug derzeit nicht realisierbar. Generic TV muss Annotationen grafisch darstellen. Ob es sich dabei um Ereignisse eines Episode oder annotierten Konzepte eines Abschnitts handelt, ist für das Visualisierungstool nicht relevant. Die in ECG und Atlas.ti generierten Daten des Metamodells werden nun kurz erläutet.

ECG Inputdaten

Ein wesentlicher Bauteil der Software ECG ist das sogenannte ECG- Labor (ECGLab). Durch die Entwicklungsumgebung Eclipse werden Rohdaten erzeugt. Sie bestehen aus XML-formatierten Dateien. Alle Events, welche in Eclipse aufgelöst worden sind (z.B. „Datei öffnen“, „Datei ändern“u.ä.), werden protokolliert. Diese Rohdaten stellen die Inputdaten von ECG dar. Innerhalb des ECGLabs findet eine sogenannte Episodenerkennung statt. Dabei werden aus den Rohdaten zusammengehörende Ereignisse ausgesucht. Diese bilden Episoden. Das Writertool des ECGLabs erzeugt bestimmte Dateien. Sie werden in dieser Arbeit ECG- Dateien genannt. In ihnen sind alle erkannten Episoden mit ihren Eigenschaften (z.B. Anfangs- und Endzeitpunkten) enthalten. Die Abbildung 10 zeigt eine Zusammenfassung der Generierung der Inputdaten durch ECG.

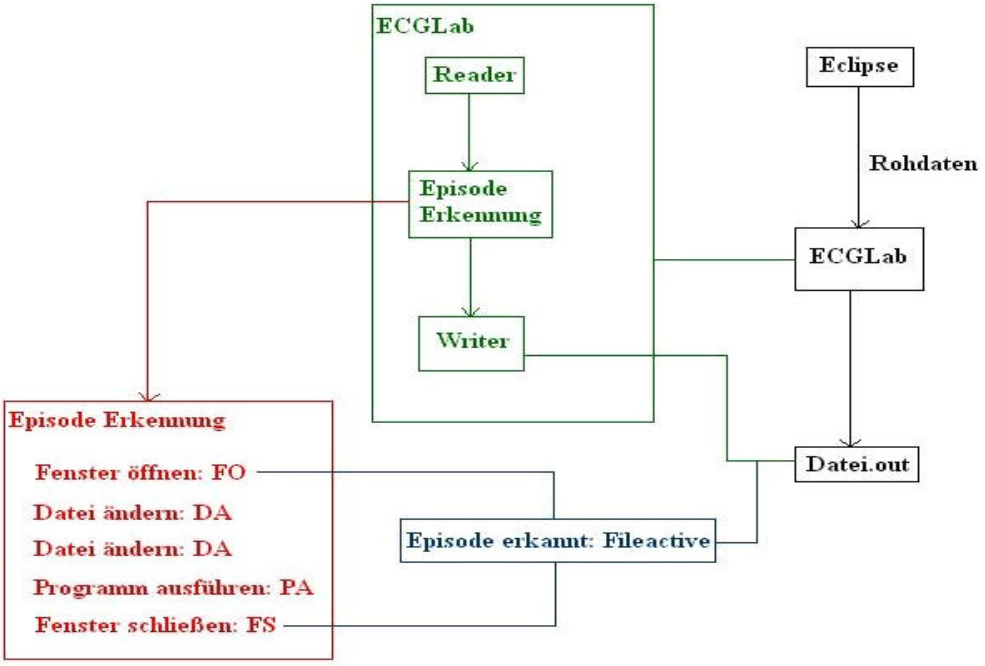


Abbildung 10 - Zusammenfassung der Generierung der Inputdaten durch ECG

Eine ECG-Datei ist in der Abbildung 10 durch das Element Datei.out gekennzeichnet. Sie enthält die Episoden. In der Abbildung 10 werden zwei Episoden erkannt:

- Fileactive ist durch die Ereignissen „Fenster öffnen“ und „Fenster Schließen“charakterisiert.
- Run ist durch den Ereignis „Programm ausführen“charakterisiert.

Eine ECG-Datei enthält also eine Abfolge von Episoden über ein bestimmtes Aufzeichnungszeitintervall. Alle Informationen[26] werden in der Episode mitprotokolliert. Der Inhalt einer ECG-Datei ist durch folgende Elemente charakterisiert:

Element	Beschreibung

[26] vorkommende Ereignisse

Type	Charakterisiert den Typ der Episode
Timestamp	Anfangszeitpunkt der Episode
Endtime	Endzeitpunkt der Episode
Ereignis	Ein in der Episode enthaltenes Ereignis
Eigenschaft 1	Erste Eigenschaft des Ereignisses
Eigenschaft 2	Zweite Eigenschaft des Ereignisses
.....	.....
Eigenschaft n	n-te Eigenschaft des Ereignisses

Tabelle 2 - Zusammenfassung der Inhalt einer durch ECG generierten Datei

Atlas.ti Inputdaten

In Kapitel II ist das QDA-Werkzeug Atlas.ti vorgestellt worden. Es ermöglicht unter anderem Videosequenzen von Entwicklersitzungen mit allen „relevanten" Tätigkeiten durch das Kodierverfahren zu annotieren. Jede Videosequenz kann sich aus anderen Videosequenzen zusammensetzen. Diese Abschnitte mit den zugehörigen Annotationen werden als XML- oder Textdateien exportiert. Der Inhalt einer XML-Datei ist durch folgende Elemente bzw. Objekte charakterisiert:

Element	Beschreibung	
Abschnitt	▪ Ist ein Datenmaterial (Videosequenz). ▪ Ist durch einen Anfangs- und Endzeitpunkt definiert. ▪ Besitzt einen Namen und evt. eine Beschreibung. ▪ Kann durch eine Anzahl von ***Konzepten*** auf ihrer gesamten Länge oder nur einem Teil annotiert werden.	
Konzept	▪ Annotiert einen Abschnitt. ▪ Ist durch seinen Namen, welcher von der ***Konzeptklasse*** abhängig, ist definiert. ▪ Er besitzt ***Eigenschaften,*** die eine ***Ausprägung*** haben	
Konzeptklasse	▪ Ist eine Menge von ***Konzepten***, die eine bzw. mehrere bestimmte Eigenschaften erfüllen	
Eigenschaft (Ereignis)	▪ Ist ein ***Konzept***. ▪ Charakterisiert die einzelnen ***Ausprägungen*** von ***Konzepten*** mit folgenden Bedingungen: • kann unabhängig von ***Konzepten*** definiert werden, • kann einem oder mehreren ***Konzepten*** zugewiesen werden, • besitzt ***Ausprägungen,*** • Bei Annotieren eines ***Abschnitts*** durch ***Konzepte*** kann nur Eigenschafts-***Ausprägung*** verwendet werden.	
Ausprägung	▪ Ist der Wert einer ***Eigenschaft***. ▪ Ist auch ein ***Konzept***, das durch eine bestimmte ***Relation*** an ***Eigenschaften*** zugewiesen werden kann.	
Relation	▪ Ist eine wiederverwendbare Beziehung, die ausgewählte ***Abschnitte*** verlinkt. (Abschnitt A folgt bzw. ist gefolgt von Abschnitt B)	In beiden Fällen besitzen sie einen Namen, ggf. eine Richtung und opt. eine

	▪ Ist eine wiederverwendbare Beziehung zwischen ausgewählten ***Konzepten***. ▪ Sie stellt die Zuordnung von Eigenschaft zum Konzept oder Ausprägung zur Eigenschaft dar. Kodierungsschritt → Geschwindigkeit → schnell	Beschreibung.

Tabelle 3 - Zusammenfassung des Inhalts einer durch Atlas.ti generierten Datei

Eine durch Atlas.ti generierte XML-Datei (die Datei werden in dieser Arbeit Atlas.ti- Datei genannt) enthält also alle relevanten Informationen, deren Generic TV die Visualisierung realisieren muss. Da das ECG- Dateiformat das erste und momentan einzige Dateiformat ist, das Generic TV einlesen und visualisieren kann hat Stephan Salinger die Implementierung eines Werkzeugs realisiert, das aus dem generierten XML- Datei, die für die Visualisierung benötigten Datensätzen „ausfiltert" und in einem ECG- Dateiformat umwandelt und dadurch die sogenannte Atlas.ti- Daten generiert. Dabei sind die Haupteigenschaften eines ECG- Formats so definiert:

- **Alle Elemente (In Tabelle 4 durch Spalte dargestellt) werden mit Tabulator getrennt.**

- **Jede Datei muss eine „Kopfzeile" (In der Tabelle 4 Fett markiert) besitzen**

-
- **Jede Datei muss folgende Elemente (In der Tabelle rot markiert) beinhalten**:
  a. Type
  b. Timestamp
  c. Endtime
  d. Ereignis bzw. Konzept

Die Tabelle 4 zeigt, wie eine durch Generic TV darstellbare Datei formatiert werden muss.

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

Tabelle 4 –Formatierung der zurzeit dargestellten Dateien

Die Tabelle 4 stellt auch die Generizität von Generic TV dar. Alle Dateien, welche gleichermaßen formatiert wurden, sind durch Generic TV dargestellbar. Unabhängig von ihrer Herkunft. Im Zusammenhang zum vorgestellten Metamodell, existieren folgende Abbildungen zwischen generierten Inputdaten (ECG- und Atlas.ti- Dateien) sowie Elementen des Metamodells.

Metamodellelement	**ECG- Daten**	**Atlas.ti- Daten**
Quotation	Episode	Abschnitt
Concept	Ereignis, Focus	Konzept
ConceptClass	Konzeptklasse	Type

Property	Eigenschaft	Eigenschaft
Value	Ausprägung	Ausprägung

Tabelle 5 - Abbildung der Inputdaten auf Metamodellelement

Die Abbildung 11 und 12 zeigen jeweils Ausschnitte aus ECG- und Atlas.ti- Daten. Aus der Abbildung 12 ist ersichtlich, dass die dargestellten Atlas.ti-Daten keine Eigenschaften besitzen. Im Gegensatz dazu haben die ECG- Daten vier Eigenschaften: duration, changed, activity, param1 und mode. In beiden Dateien sind aber die vier „Musselemente“ vorhanden: type, timestamp, endtime und resourcename.

type	timestamp	endtime	resourcename	duration	changed	activity	param1	mode
msdt.partactive.xsd	30.05.2006 16:54:53	30.05.2006 16:55:12	Welcome	19000				
msdt.partactive.xsd	30.05.2006 16:55:12	30.05.2006 16:55:15	Package Explorer					3000
msdt.windowactive.xsd	30.05.2006 16:54:53	30.05.2006 16:55:15	Java - Eclipse SDK					22000
msdt.partactive.xsd	30.05.2006 16:55:37	30.05.2006 16:56:54	Package Explorer					77000
msdt.windowactive.xsd	30.05.2006 16:55:37	30.05.2006 16:56:54	Java - Eclipse SDK					77000
msdt.partactive.xsd	30.05.2006 16:57:02	30.05.2006 16:57:29	Package Explorer					27000
msdt.windowactive.xsd	30.05.2006 16:57:02	30.05.2006 16:57:29	Java - Eclipse SDK					27000
msdt.partactive.xsd	30.05.2006 16:57:43	30.05.2006 16:57:45	Package Explorer					2000
msdt.windowactive.xsd	30.05.2006 16:57:43	30.05.2006 16:57:45	Java - Eclipse SDK					2000
msdt.partactive.xsd	30.05.2006 16:57:45	30.05.2006 16:57:45	Package Explorer					0
msdt.windowactive.xsd	30.05.2006 16:57:45	30.05.2006 16:57:45	Java - Eclipse SDK					0
msdt.partactive.xsd	30.05.2006 16:57:56	30.05.2006 16:57:57	Package Explorer					1000
msdt.windowactive.xsd	30.05.2006 16:57:56	30.05.2006 16:57:57	Java - Eclipse SDK					1000
msdt.partactive.xsd	30.05.2006 16:57:59	30.05.2006 16:57:59	Package Explorer					0
msdt.windowactive.xsd	30.05.2006 16:57:57	30.05.2006 16:58:00	Confirm Exit				3000	
msdt.partactive.xsd	30.05.2006 16:58:35	30.05.2006 16:58:35	Welcome	0				
msdt.partactive.xsd	30.05.2006 16:58:35	30.05.2006 16:58:37	Package Explorer					2000
msdt.windowactive.xsd	30.05.2006 16:58:35	30.05.2006 16:58:37	Java - Eclipse SDK					2000
msdt.partactive.xsd	30.05.2006 16:58:39	30.05.2006 16:58:47	Package Explorer					8000
msdt.windowactive.xsd	30.05.2006 16:58:39	30.05.2006 16:58:47	Java - Eclipse SDK					8000

Abbildung 11 - ECG- Inputdaten

type	timestamp	endtime	resourcename
GT: HHI	10.02.2006 16:00:02	10.02.2006 16:02:56	P.think aloud_activity
GT: P	10.02.2006 16:00:02	10.02.2006 16:02:56	P.think aloud_activity
GT: HHI	10.02.2006 16:02:56	10.02.2006 16:03:49	P.think aloud_problem
GT: P	10.02.2006 16:02:56	10.02.2006 16:03:49	P.think aloud_problem
GT: HHI	10.02.2006 16:07:17	10.02.2006 16:08:03	P.think aloud_problem
GT: P	10.02.2006 16:07:17	10.02.2006 16:08:03	P.think aloud_problem
GT: HHI	10.02.2006 16:08:11	10.02.2006 16:08:45	P.think aloud_problem
GT: P	10.02.2006 16:08:11	10.02.2006 16:08:45	P.think aloud_problem
GT: HHI	10.02.2006 15:57:58	10.02.2006 15:57:59	P1.agree_action
GT: HHI.P1P2	10.02.2006 15:57:58	10.02.2006 15:57:59	P1.agree_action
GT: P1	10.02.2006 15:57:58	10.02.2006 15:57:59	P1.agree_action
GT: P1.HHI	10.02.2006 15:57:58	10.02.2006 15:57:59	P1.agree_action
GT: HHI	10.02.2006 16:06:39	10.02.2006 16:06:44	P1.agree_action
GT: HHI.P1P2	10.02.2006 16:06:39	10.02.2006 16:06:44	P1.agree_action
GT: P1	10.02.2006 16:06:39	10.02.2006 16:06:44	P1.agree_action
GT: P1.HHI	10.02.2006 16:06:39	10.02.2006 16:06:44	P1.agree_action
GT: HHI	10.02.2006 16:07:03	10.02.2006 16:07:05	P1.ask_knowledge
GT: HHI.P1P2	10.02.2006 16:07:03	10.02.2006 16:07:05	P1.ask_knowledge
GT: P1	10.02.2006 16:07:03	10.02.2006 16:07:05	P1.ask_knowledge
GT: P1.HHI	10.02.2006 16:07:03	10.02.2006 16:07:05	P1.ask_knowledge
GT: Change Driver	10.02.2006 16:03:49	10.02.2006 16:03:54	P1.become_driver
GT: Change Driver	10.02.2006 16:07:21	10.02.2006 16:07:23	P1.become_driver
GT: HHI	10.02.2006 15:56:11	10.02.2006 15:56:13	P1.challenge_knowledge
GT: HHI.P1P2	10.02.2006 15:56:11	10.02.2006 15:56:13	P1.challenge_knowledge

Abbildung 12 - Atlas.ti- Inputdaten

III.3. Das Framework

Die Wiederverwendungvon Software wurde von einer großen Mehrheit der Wissenschaftler befürwortet. Die positiven Auswertungen kann man an vielen Schritten des Softwareentwicklungsprozesses erkennen. Ein positiver Effekt ist z.B., dass man auf existierende Problemlösungen zurückgreifen kann.Dadurch reduziert sich die Entwicklungszeit [JF88]. Ein weiterer positiver Aspekt ist, dass dadurch die Wartbarkeit der erstellten Software erhöht wird [JF88]. Die Erklärung dafür ist, dass wiederverwendete Komponenten bereits in verschiedensten Projekten bzw. Umgebungen eingesetzt worden sind. Auf diese Weise wurde bereits die härteste Testphase einer Software, nämlich das Testen durch Benutzer, absolviert. Wiederverwendete Komponenten sind in der Regel zuverlässiger als neu entwickelte. Ein Beispiel für die Zuverlässigkeit solcher Komponenten ist die Benutzung von Java-Bibliotheken aus dem JDK. Fast alle Javaentwickler verlassen sich auf die Korrektheit der verwendeten Softwarebausteine. Es kommen z.B. wenig Fragen (über die Korrektheit) auf, wenn Java-Swingkomponenten bei der Entwicklung einer Benutzeroberfläche verwendet werden. Diese vorausgesetzte Zuverlässigkeit, ermöglicht der Testphase der Softwareentwicklung, weniger Aufwand.

H. Balzert definiert ein Framework als eine Menge von zusammenarbeitenden Klassen, die einen wiederverwendbaren Entwurf für einen bestimmten Anwendungsbereich implementieren [BH05]. Eine der Haupteigenschaften eines Frameworks ist, dass der Benutzer die Möglichkeit hat, bestimmte Klassen des Frameworks zu erweitern. Der Benutzer des Frameworks kann den Gebrauch anpassen. Diese manipulierbaren Klassen des Frameworks sind die abstrakten Klassen, die die Schnittstelle zur Außenwelt darstellen. Der Einsatz solcher Klassen ist durch den Zweck des Frameworks definiert. Nicht die Codewiederverwendung, sondern die Entwurfswiederverwendung steht im Vordergrund [BH05].

Wenn man nur die Vorteile der Wiederverwendung von Software betrachtet, ist ihre Benutzung zweifellos vorteilhaft. Der Grundgedanke objektorientierter Software-Entwicklung ist, dass die abstrakte Abbildung in Klassen von allgemeinen Konzepten [HB98].Der Einsatz von Frameworks in der (objektorientierten) Softwareentwicklung basiert auf dieser Idee. Einsatz, der die zwei vielleicht wichtigsten Mechanismen der Objektorientierung anwendet:

Der zweite Anforderungspunkt dieser Diplomarbeit ist: „Suche nach wiederverwendbaren Komponenten“. Trotz der Vorteile der Frameworks gibt die Schwierigkeit des Auffindens eines „geeigneten“ Frameworks. Wie sollte man die Suche der wiederverwendbaren Komponente gestallten und evtl. optimieren, um sie effizient zu machen? Es hat sich bei der Entwicklung von Generic TV ergeben, dass dieser Punkt enorm wichtig ist, weil nur durch den Einsatz eines Framework. Nur dadurch kann Entwicklungszeit reduziert.

III.3.1. Motivation der Auswahl und Vorstellung von JFreeChart

Die Suche von wiederverwendbaren Komponente für dieses Projekt war nicht einfach. Es ist eine Aufgabe, die von mir am Anfang der Entwicklungsphase unterschätzt wurde. Mir war klar, dass, als Einflussfaktoren der Entwurfphase, der Einsatz von Frameworks empfohlen ist. Die Komplexität der Suche war mir aber nicht bewusst. Anfangs war meine Suche auf die Begriffe „Darstellung“ und „zeitabhängige Daten“ konzentriert. Der Fehler dabei war, dass meine Suche unstruktuiert verlief. Deshalb erarbeitete ich eine Roadmap mit abzuarbeitenden Fragen:

- **Was muss ich machen?** Die Hauptanforderung an Generic TV war Programmiertätigkeiten visuell darstellen zu können. Daraus ergibt sich die zweite Frage.

- **Wie kann bzw. möchte ich die Programmiertätigkeit darstellen?** Eine Literaturrecherche über Visualisierung ließ mich klare Entscheidungen treffen. Eine Hautpanforderung an eine Visualisierung ist die Effektivität[27] der grafischen Darstellung von abstrakten Daten. Außerdem ist die Visualisierung von abstrakten Daten nur durch die Benutzung von grafischen Elementen möglich. Aus diesem Grund traf ich die Entscheidung die unterschiedlichen Programmiertätigkeiten durch geometrische Figuren zu visualisieren. Um ihre Anfangs- und Endzeitpunkte zu erkennen werden sie über eine Zeitachse dargestellt. Aus diesen Entscheidungen ließ sich die dritte Frage ableiten.

- **Welches grafische Element möchte ich dafür benutzen?** Generic TV muss zeitabhängige Daten darstellen können. Dabei ist die Darstellung von zwei Hauptelementen wichtig

    a. Auf der einer Seite die Programmiertätigkeiten (Die in Anhang C durchgeführte Analyse begründet die Auswahl der Anwendung von bestimmten grafischen Elementen)

    b. Auf der anderen Seite die Zeit. In der Literatur gibt es viele Vorschläge für die Visualisierung der Zeit. Eine einfache und effiziente Möglichkeit der Zeitdarstellung ergab sich als Zeitdiagramm. Sie werden als Koordinatensystem definiert. Die Größe Zeit wird auf einer Koordinatenachse dargestellt.

Die Analyse und Beantwortung dieser Fragen erwiesen sich als sehr hilfsreich für die „Re-Orientierung“ bei der Suche nach Frameworks. Eine der Ausprägung von Zeitdiagrammen ist das Balkendiagramm. Balkendiagrammen eignen sich gut zur Darstellung von Rangfolgen. Eine besondere Form des Balkendiagramms ist das Gantt-Diagramm, das in der Zeitplanung von Aktivitäten oft angewendet wird. Wie man in Abbildung 13 erkennen kann, werden Soll- und Ist-Aktivitäten zeitabhängig dargestellt. Die Soll- und Ist-Aktivitäten werden farblich unterschieden. Falls eine Aktivität stattgefunden hat, wird ihre Gesamtdauer grafisch reproduziert.

[27] Der Begriff Effektivität sowie die Anforderungen an eine Visualisierung werden im nächsten Kapitel genauer gegliedert.

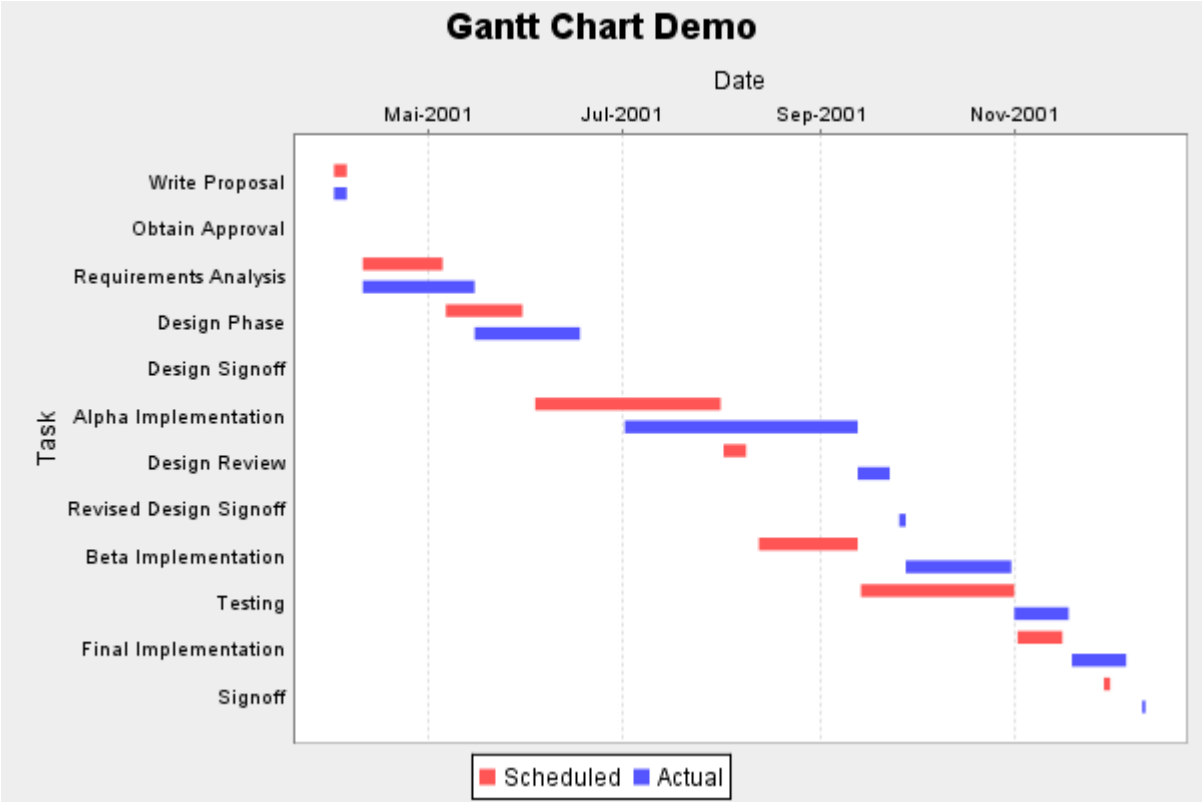


Abbildung 13 - Gantt- Diagramm von JFreeChart

Gantt- Diagramm

Ein Gantt-Diagramm ist ein nach dem amerikanischen Berater Henry L. Gantt (1861–1919) benanntes Instrument des Projektmanagements. Eine **zeitliche Abfolge von Aktivitäten grafisch in Form von Balken auf einer Zeitachse darstellt** [WIK_I]. Die Definition eines Gantt-Diagramms ist die genaue Wiederspiegelung der Basis der Darstellungsart, die ich mit Generic TV ereichen wollte. Meine Idee war der Begriff „Aktivitäten“, der in der Definition vom Gantt-Diagramm erwähnte wurde, durch „Programmiertätigkeiten“ zu ersetzen. Dadurch würde ich folgende neue Definition bekommen: Ein Genric TV-Diagramm ist ein [...], **das die zeitliche Abfolge von Programmiertätigkeiten grafisch in Form von Balken auf einer Zeitachse darstellt**. Dies würde bedeuten, dass die Balken des Generic TV-Diagramms die Spuren repräsentieren würden.

Diese Idee war meine Hauptmotivation für die Entscheidung, das Gantt-Diagramm als „Vorbild“ zu benutzen Die Suche nach Frameworks verfeinerte sich auf die Suche nach wiederverwendbaren Gantt-Diagrammen. Die Suche von Gantt-Diagramm ergab eine Vielzahl von Java Bibliotheken, die die Realisierung von Gantt-Diagramm implementieren (GTD-Manager [GM_I], ChartDirector [CD_I], NetCharts Graph [NC_I] und JFreeChart [JFC_I] etc.).

Die große Anzahl von Möglichkeiten erzwang mich die Suchkriterien weiter zu verfeinern. Da die Roadmap meine Suche besser strukturiert hatte, entschied ich mich zur Präzisierung meines Fragenkatalogs. Dabei ergaben sich folgende Fragen:

- Welche Benutzerinteraktion möchte ich nicht selbst implementieren?

- Erfüllt das Framework die gewünschten Benutzerinteraktionen bzw. Systemanforderungen? Wenn ja, welche?
- Wie sind Framework-Schnittstellen mit der Außenwelt implementiert?
- Wie ist die Dokumentation des Frameworks?
- Ist es erweiter- bzw. änderbar?
- Wie zukunftssicher ist das Framework?
- Wie groß ist die Benutzergruppe?
- Wie stabil ist es?

Neben der Erfüllung von Systemanforderungen sind die Stabilität, die Erweiterbar- und Änderbarkeit und die Dokumentation entscheidende Punkte für die Auswahl eines Frameworks. Die Größe der Benutzergruppe spricht für die Zuverlässigkeit und die Stabilität einer Software. Mit einer geschätzten Benutzeranzahl[28] zwischen 40000 und 50000 [JFC_I] hat JFreeChart jede Konkurrenz klar und deutlich geschlagen. JfreeChart wurde in renommierten Softwareprojekten, wie z.B. NetBeans und JBoss [JFCU_I], eingesetzt.Seine Zuverlässigkeit, Erweiterbar- und Änderbarkeit haben überzeugt. Dieses Framework wurde überpruft, ob es Generic TV-Anforderungen vorherein erfüllt (Siehe Tabelle 6). Dabei wurden die bekannten Anforderungen, an das zu entwickelnde System (Siehe Anhänge A und B), mit den durch JFreeChart angebotenen Funktionalitäten verglichen. Die folgende Tabelle beschreibt Funktionalität-Anforderung-Beziehung, also die JfreeChart-Generic TV-Beziehung.

Grafisches Element	**Anforderungen des Systems an diese Elemente**	**JFreeChart Funktionalitäten**
Diagramm	Zeitproportionale Darstellung von geometrischen Figuren.	**wird erfüllt X**
	Darstellung von unterschiedlichen geometrischen Figuren.	**wird nicht erfüllt**
	Unterschiedliche Farben für die geometrischen Figuren	**wird nicht erfüllt**
	Farbe der geometrischen Figuren dynamisch (in der Laufzeit) ändern.	**wird nicht erfüllt**
	eschriftung der geometrischen iguren	**begrenzt**
	Interaktivität beim Mauseklick auf der geometrischen Figuren	**begrenzt**
	Zoomen	**wird erfüllt**
	Ausdrucken (inkl. PDF)	**wird erfüllt**
	Speichern der Darstellung als Bilddatei	**wird erfüllt**

[28] Anzahl von Entwickler, die eine Software benutzen

	Filtern	**wird nicht erfüllt**
Text	Hinzufügen von Text an beliebigen Stellen des Diagramms	**begrenzt**
	Farbe für Text dynamisch auswählen und setzen	**wird nicht erfüllt**
	Interaktivität beim Mauseklick auf Text	**wird nicht erfüllt**
Grafik (Icon)	Hinzufügen von Grafiken bzw. Icons an beliebigen Stellen des Diagramms	**begrenzt**
	Interaktivität beim Mauseklick auf Grafiken	**wird nicht erfüllt**

Tabelle 6 - Analyse der erfüllten Systemanforderungen durch JFreeChart

Die progressive und systematische Suchweise von Frameworks brachte mich dazu, mich für JFreeChart zu entscheiden. JFreeChart ist eine grafische Java-Bibliothek, welche seit 6 Jahren angewendet wird. „Pie-Diagramme“, „Balken- Diagramme” (mit 3D Optionen), „Line-Diagramme” und natürlich Gantt- Diagramm können mit JFreeChart sehr einfach generiert werden. Die Funktionalitäten (Druck, Zoom etc.) und die sehr gut dokumentierten Schnittstellen sind ein Pluspunkt dieses Frameworks. Das sehr flexible und durchschaubare Design von JFreeChart ermöglicht, jede denkbare, nötige bzw. angeforderte Erweiterungen an das Framework zu realisieren.

III.3.2. Einblick in die Architektur von JFreeChart[29]

Ein großer Vorteil von JFreeChart ist sein sehr leicht verständliches Design. Es werden nur Frameworkklassen vorgestellt, die für Generic TV die wichtig sind. Die Abbildung 14 zeigt eine kurzgefasste Form der Generierung von Gantt-Diagrammen mit JFreeChart. Dabei ist das Verständnis folgender Klassen von entscheidender Bedeutung:

ChartFactory: Nach dem Prinzip des Fabrikmethode-Musters, generiert diese Klasse alle Diagramme, die durch JFreeChart erzeugt werden können. Welches Diagramm generiert werden soll, ist vom Typ seiner Inputdaten abhängig.

TaskSeriesCollection: Ist die Inputdaten des Diagramms

TaskSeries: Ist eine Kollektion aus keinem oder mehreren Task(s). Stellt eine Serie des Diagramms dar. Diese Klasse ist eine sehr wichtige Schnittstelle zur Generic TV. Die Klasse Track von Generic TV, die die Spuren erzeugt, ist eine Ableitung von TaskSeries.

Task: Stellen die einzelne Elemente der TaskSeries dar. Ist auch eine wichtige Schnittstelle zu Generic TV. Die Klasse GenTVTask, die die Programmiertätigkeiten modelliert, erbt von der Klasse Task.

GanttRenderer: Ist die Klasse, die das Malen des grafischen Objekts oder Text übernimmt. Ist eine weitere Schnittstelle zur Generic TV, bei der viele Änderungen durchgeführt werden

[29] Eine genaue Beschreibung der Schnittstellen zwischen JFreeChart und Generic TV wird im Kapitel V vorgestellt. Dabei wird der Entwurf von Generic TV präsentiert.

müssten. Die vom GanttRenderer abgeleitete Klasse GenTVGanttRenderer hat (unter anderem) die Aufgabe mögliche quantitative Visualisierungen zu realisieren und unterschiedlichen Farbe(n) und/oder Form(en) zuzuweisen.

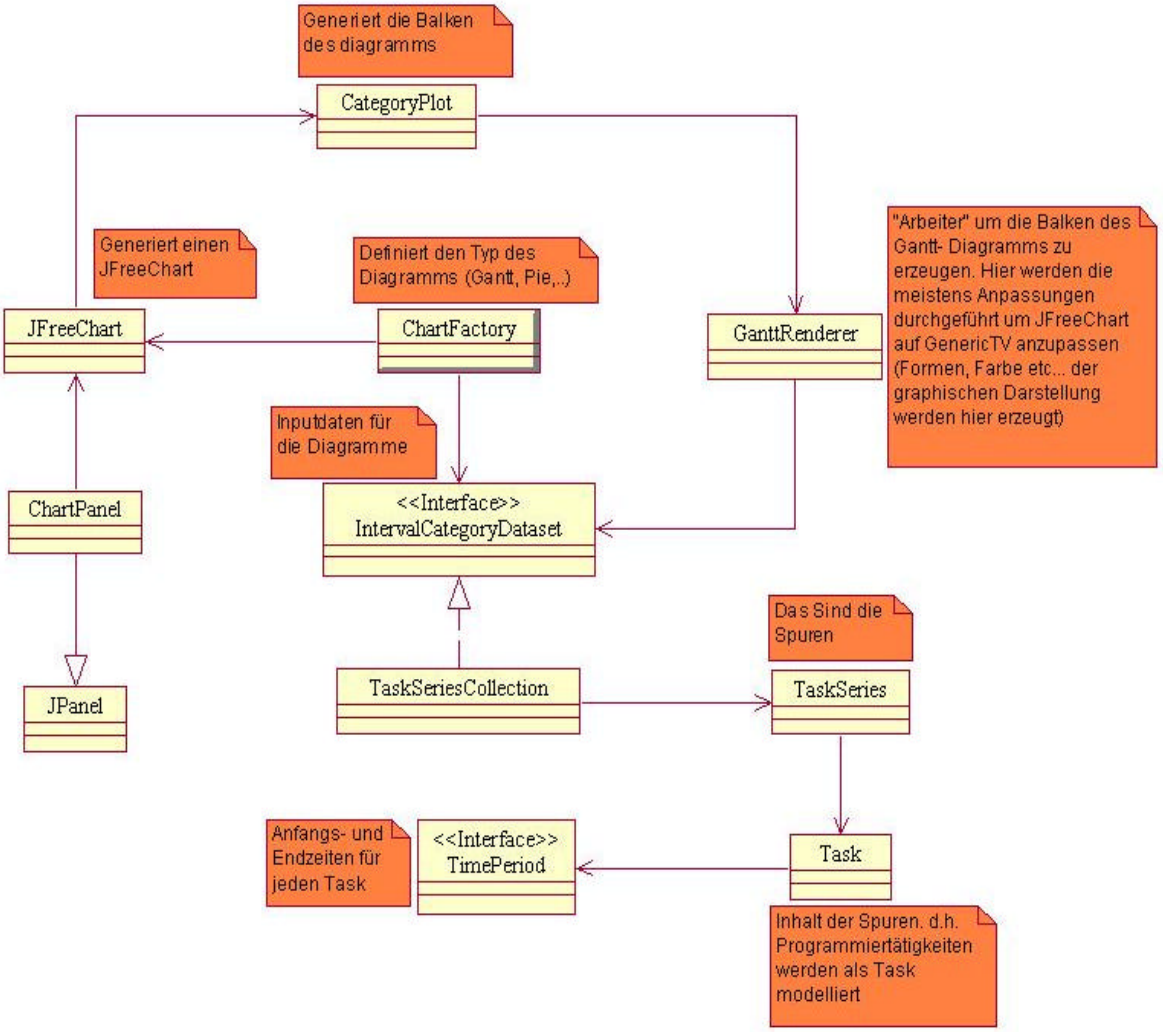


Abbildung 14 - Zusammenfassung der Generierung von Gantt- Diagrammen mit JFreeChart

IV. Visualisierung – Stand der Forschung und der Technik

Im folgenden Kapitel wird der Begriff der Visualisierung näher beleuchtet. Dazu gehört unter anderem eine Schwerpunktsetzung auf die Ziele, Gründe und vor allem auf die vorhandenen Techniken der Visualisierung.

Eine Visualisierung ist das Ergebnis eines komplexen Prozesses. Dieser Visualisierungsprozess wird in diesem Kapitel vorgestellt. Alle Phasen der Entstehung eines Bildes aus abstrakten Daten werden des weiteren behandelt. Der letzte Teil des Kapitels, bevor die Anforderungen an ein Visualisierungswerkzeug dargestellt werden, beschäftigt sich mit speziellen Anzeigetechniken.

IV.1. Begriffserklärung und Einführung

„Imagination or visualization, and in particular the use of diagrams, has crucial part to play in scientific investigation" **Descartes, 1637**

Während nach der allgemeinen Definition der Begriff „Visualisierung" eine optische Darstellung und Sichtbarmachung bezeichnet [DD00], versteht die Computertechnologie darunter eine Umwandlung von nicht-bildlichen Informationen in eine grafische Darstellung[SH04]. Auf dem Gebiet der Computertechnologie ist Visualisierung ein wichtiger Bestandteil von Bereichen wie:

- „image processing": Analyse, Manipulation, Speicherung und Darstellung von Bildern;
- „computer vision": Analyse eines Bildes und seine Konvertierung in eine abstrakte Beschreibung;
- „computer-aided design": das Modellieren von Objekten, wie zum Beispiel Häuser, Brücken, Autos usw. mittels spezieller Software.

Im Rahmen dieser Diplomarbeit wird ein Informationsvisualisierungswerkzeug entworfen und implementiert. Der Begriff Informationsvisualisierung (Infovis) ist der zentrale Punkt dieser Arbeit. Die Definition einer Infovis wurde in der Einleitung dieser Arbeit vorgestellt. Es ist eine Datenvisualisierung, die besondere Eigenschaften besitzt und wird als eine grafische Darstellung der Strukturen und Zusammenhänge von abstrakten Daten definiert [SM00]. Dabei ist ihr Ziel die abstrakte Daten und Informationen grafisch so zu präsentieren, dass ihre relevanten Charakteristika intuitiv erfasst werden können[SM04].

Eine Infovis ist also ein „Bild" für das menschliche Auge. Der menschliche Sinn, der durch eine Infovis zuerst beansprucht wird, das Sehen, begründet die „Fülle"(Mächtigkeit) einer grafischen Darstellung. Das Auge liefert nicht nur Daten zum Gehirn, sondern verarbeitet sie bereits bei der Aufnahme [MJ99]. Die enge und effiziente Zusammenarbeit zwischen menschlichem Auge und Infovis stellt K. Graw deutlich dar: *„Der Mensch kann über das Auge sehr viele optische Informationen auf einmal wahrnehmen und diese auch schnell mit dem Gehirn verarbeiten. Mit Hilfe einer grafischen Darstellung können deshalb viele komplexe Prozesse und Phänomene besser veranschaulicht und verstanden werden. Die visuelle Analyse von Daten erweitert damit die Möglichkeiten der Extraktion von Informationen aus Daten in Vergleich zu herkömmlichen Methoden."* [GK96].

Es wäre ein Irrtum zu glauben, dass die Benutzung jeder Infovis ihr Ziel immer erreicht. Die Zielerreichung eines Prozesses, der von einer Infovis unterstützt wird, ist durch diverse An-

forderungen an die Infovis definiert. Diese werden maßgeblich durch die Fähigkeiten und Eigenschaften des menschlichen Auges sowie des Gehirns bestimmt

„Betrachtet man den Zusammenhang zwischen bildlicher Darstellung und Wirklichkeit, so ist der Erfolg der Visualisierung […] in erster Linie davon abhängig, in welchem Maße der Betrachter fähig ist, den Kontext der realen Welt aus der Abbildung zu rekonstruieren und die wahrgenommenen Strukturen in der Abbildung mit den tatsächlich existierenden Korrelationen zwischen Parametern bzw. Störungen im Datenfeld in Verbindung zu bringen.“ [SM00].

Die Anforderungen an eine Infovis stellt H. Schumann, Leiterin des Lehrstuhls Computergrafik an der Universität Rostock, folgendermaßen vor: **Die Analyse, das Verständnis, die Kommunikation von Modellen, Konzepten, Daten sollen durch eine Infovis erleichtert werden.**

Die so genannten Anforderungselemente einer Infovis „bürgen“ für die Wahrung der Vorteile einer grafischen Darstellung. Zu den Anforderungselementen einer Infovis gehören unter anderem Bildqualität, Interaktion, Geschwindigkeit sowie Art und Anzahl der zu visualisierenden Objekte [SM00]. Die Infovis büßt ihre Vorteile ein, wenn die Anforderungselemente die Aufnahmefähigkeit des menschlichen visuellen Systems überschritten. Ihre Ziele würden dann verfehlt.

Zu diesem Zweck hat H. Schumann bestimmte Kriterien definiert, die zur Visualisierung einer Datenmenge berücksichtigt werden müssen [SH06]:

- Expressivität: Bilder sind expressiv, wenn die in einer Datenmenge enthaltenen Informationen (und nur diese) präsentiert werden.
- Effektivität: Bilder sind effektiv, wenn sie intuitiv wahrgenommen werden können.
- Angemessenheit: Bilder müssen in angemessener Zeit erzeugt und den Ressourcen des Ausgabegeräts angepasst ausgegeben werden.

H. Schumann stellt die Wichtigkeit dieser Kriterien für eine Infovis durch Beispiele klar dar. Infovis, bei denen die Kriterien nicht respektiert worden sind, sind sehr oft keine Hilfe. Das gewünschte Ziel zur Benutzung einer Infovis wird so verfehlt.

Beispiel zur Verletzung der Expressivität:

Die falsche Wahl von grafischen Elementen und ihrer Anordnungen ist die wesentliche Ursache für das Fehlen von Aussagekraft in einer Infovis.

Die Abbildung 15 zeigt eine Darstellung der Herkunft (in der X-Achse) von bestimmten Kraftwagen (in der Y-Achse). Die linke Seite von Abbildung 15 erweist sich als wenig geeignet, um die genaue Herkunft eines Wagens festzustellen (mittels Balkendiagramms). Stattdessen ist die Darstellung auf der rechte Seite viel mehr ausdrucksvoller. Sie ermöglicht eine bessere und vor allem einfachere Wahrnehmung der Information.

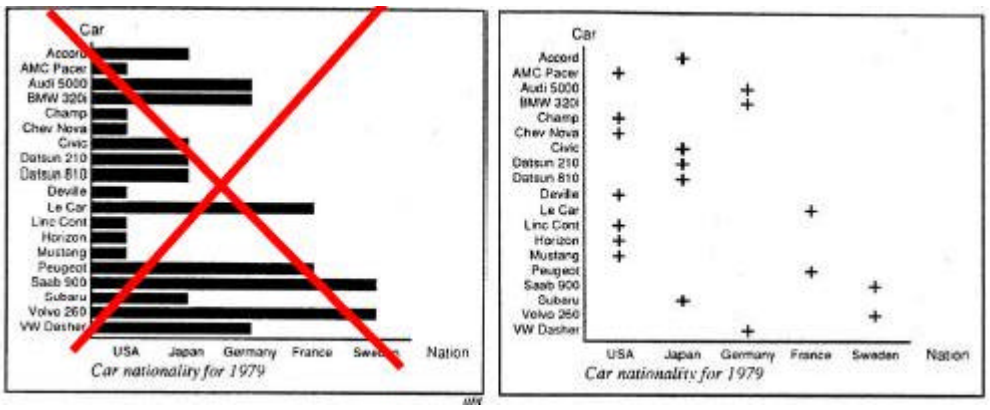


Abbildung 15 - Verletzung der Expressivität in einer visuellen Darstellung[SH06]

Beispiel zur Verletzung der Effektivität:

Die schnelle Zusammenarbeit zwischen Auge, Gehirn und Infovis wurde schon in dieser Arbeit vorgestellt. Eine Verletzung der Effektivität bei einer Infovis kann diese Zusammenarbeit schnell und vor allem stark behindern.

Die Abbildung 16 zeigt die Bewertung von Benchmarking-Ergebnissen für verschiedene Produkte. Die Ordnung der Bewertungsskala wird nicht durch eine adäquate Farbordnung wiedergegeben. Maximale und minimale Ausprägungen sind jeweils in der Farbe Blau (hell bzw. dunkel) abgebildet. Der Vergleich mit anderen Farben wird so erschwert. Aufgrund der Homogenität des Farbbildes ist ein erhöhter kognitiver Aufwand erforderlich. Eine Interpretation der Darstellung ist schwerer möglich.

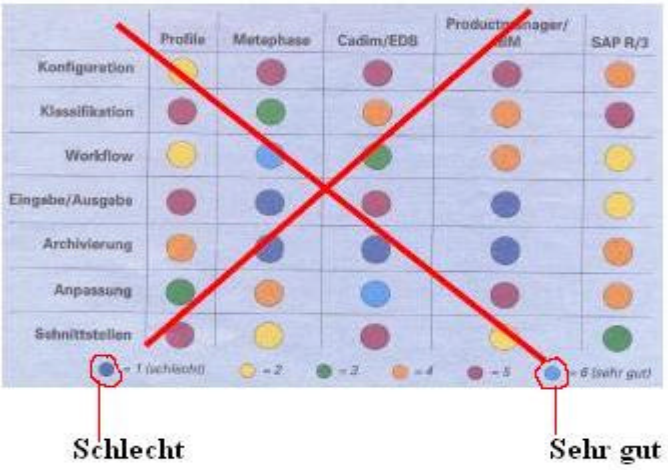


Abbildung 16 - Verletzung der Effektivität in einer Infovis[SH06]

Beispiel zur Berücksichtigung der Angemessenheit:

Im Gegenteil zu den beiden anderen Kriterien, lässt sich die Angemessenheit durch ihre Berücksichtigung besser darstellen.

Die Abbildung 17 zeigt wie die Darstellung derselben Daten durch die Ressourcen des Ausgabegeräts beeinflusst werden können bzw. sollten.

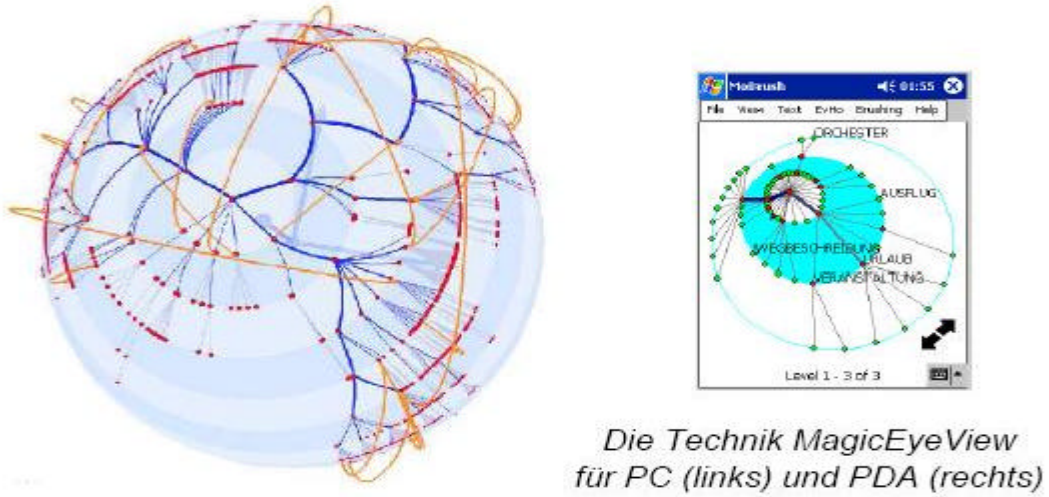


Abbildung 17- Berücksichtigung der Angemessenheit bei einer Infovis[SH06]

In der Fachliteratur wird stets klargestellt, dass die Verwendung einer Infovis nur ihre Ziele erreichen kann, wenn alle an die Infovis gestellten Anforderungen erfüllt sind. Was sind aber die verfolgten Ziele bei der Benutzung einer Infovis?

IV.2. Einflussfaktoren auf eine Visualisierung

Die verfolgten Ziele bei der Benutzung einer Visualisierung werden als Einflussfaktoren auf die Visualisierung bezeichnet.

Die Benutzung von Generic TV als Visualisierungswerkzeug verfolgt ein Analyseziel. Die Programmiertätigkeiten von Entwicklern sollen visualisiert werden, um so ihre Analyse zu ermöglichen. Die „Bearbeitungsziele“ und die „Wahrnehmungskapazitäten des Benutzer“ werden als Einflussfaktoren auf eine Infovis bezeichnet. Bei ihnen handelt es sich um Kriterien, die die Gestaltung und vor allem die ausgewählte Darstellungstechnik für die Infovis beeinflussen.

IV.2.1. Die Bearbeitungsziele einer Infovis

H. Schumann hat die Ziele bei der Benutzung einer Infovis klar unterschieden. Diese Unterscheidungen sind umso mehr wichtig, dass sie die grundlegende Technik des Entwurfs einer Infovis beeinflussen. Der wichtigste Punkt für den Entwurf einer Infovis ist zu wissen wo bzw. wofür sie eingesetzt wird. In diesem Zusammenhang hat H. Schumann die Zielstellungen beim Einsatz einer Infovis definiert. Für jedes Ziel sind der Ausgangspunkt, die Prozesse und das Ergebnis (wobei das Ergebnis, die zu erstellende Infovis ist) gegliedert [SH06]:

Zielstellungen beim Einsatz der Visualisierung:

- Explorative Analyse:

    a. Ausgangspunkt: Daten ohne Hypothese über die Daten.

b. Prozess: interaktive, ungerichtete Suche nach Informationen.

c. Ergebnis: Visualisierung, die Hypothese über die Daten unterstützt.

- Konfirmative Analyse:

  a. Ausgangspunkt: Hypothese über die Daten.

  b. Prozess: zielgerichtetes Überprüfen der Hypothese.

  c. Ergebnis: Visualisierung, die Bestätigung oder Verwerfen der Hypothese unterstützt.

- Präsentation:

  a. Ausgangspunkt: bereits bekannte Fakten.

  b. Prozess: Erkennen der Fakten.

  c. Ergebnis: Visualisierung, die Fakten hervorhebt.

Die explorative Analyse ist für Generic TV das Bearbeitungsziel. Infovis unterstützt die Wissenschaft, bei der Suche von „versteckten“ Daten in einer im Allgemeinen sehr großen Datenmenge. Die Infovis hat beim Bearbeitungsziel Analyseziel zwei grundsätzliche Aufgaben [SH06]:

- Präsentation von Ergebnissen und Sachverhalten;
- visuelle Analyse einer gegebenen Datenmenge.

IV.2.2. Wahrnehmungskapazitäten des Benutzer

Expressivität, Effektivität und Angemessenheit wurden als Kriterien definiert, die die Effizienz einer Infovis stark beeinflussen können. Das menschliche visuelle System besitzt Eigenschaften, welche man bei der visuellen Repräsentation berücksichtigen muss. Nur so hat ihre Effektivität auch tatsächlich Auswirkungen beim Benutzer. Aus der Vielzahl der Aspekte der menschlichen Wahrnehmung werden für Infovis nur zwei exemplarisch behandelt:

- Gruppierung und
- Farbwahrnehmung.

Gruppierung

Das Wahrnehmungssystem des Menschen verarbeitet visuelle Reize. Es werden Elemente in verschiedene Gruppierungsregeln zusammenfasst. Darauf basieren verschiedene Organisationsprinzipien der menschlichen Wahrnehmung, welche in Gestaltungsprinzipien erfasst werden [SH06]:

- Prinzip der Nähe:
  a. Nah beieinander liegende Formen werden als zusammengehörig empfunden.

- Prinzip der Ähnlichkeit:
  a. Gleiches oder fast Gleiches wird als zusammengehörig empfunden.
  b. Farbe gruppiert stärker als Form

- Prinzip der Überlagerung:
  a. Die visuelle Wahrnehmung erkennt einfache, mit den vorhandenen Erfahrungen am besten zu vereinbarenden, Formen.

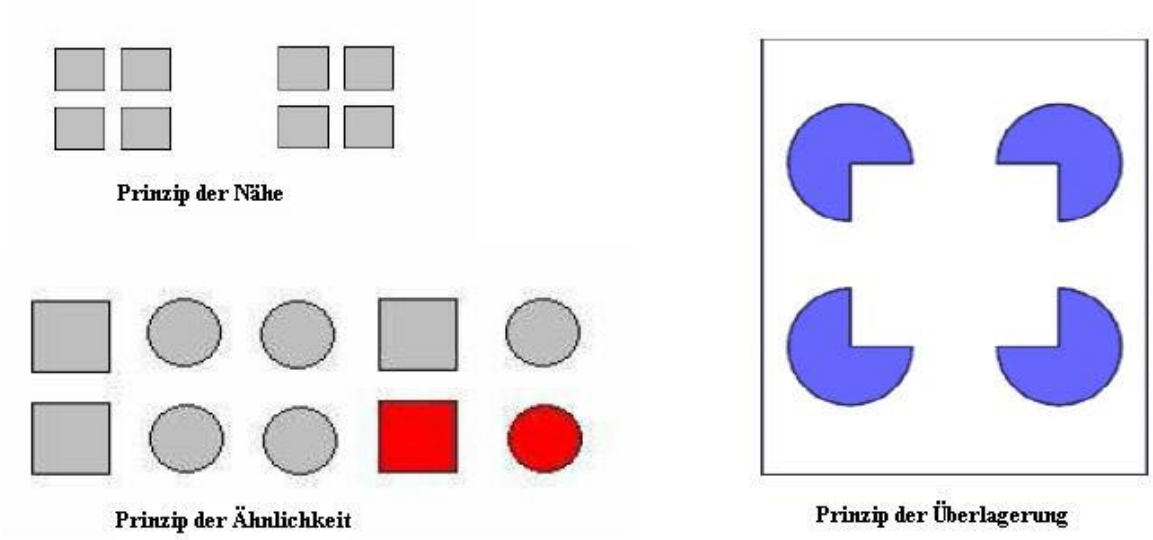


Abbildung 18 - Gestaltungsprinzipien[SH06]

Farbwahrnehmung

Farben können einen sehr großen Einfluss auf die Wahrnehmungskapazitäten eines Menschen haben. Aus diesem Grund ist ihre Anwendung in einer Infovis von Relevanz. Eine durch Farbe erweckte menschliche Empfindung ist in hohem Maß von der Umgebung abhängig H. Schumann definiert folgenden Abhängigkeiten [SH06]:

- Einfluss der Größe auf empfundene Leuchtkraft der Farbe:

  a. **Quantitätskontrast**: Eine helle Fläche kleinerer Größe hält Balance mit einer dunklen größeren Fläche.

- Einfluss der Helligkeit auf die empfundene Größe (**Irradiations-Effekt**):

  a. Ein weißes Rechteck wirkt neben einem schwarzen Rechteck gleicher Größe länger.

- Einfluss der Umgebungsfarbe auf das Farbempfinden (**Simultankontrast**):

  a. Ein gleichmäßig grauer Ring erscheint auf einem roten Hintergrund grünlich und auf einem grünen Hintergrund rötlich.

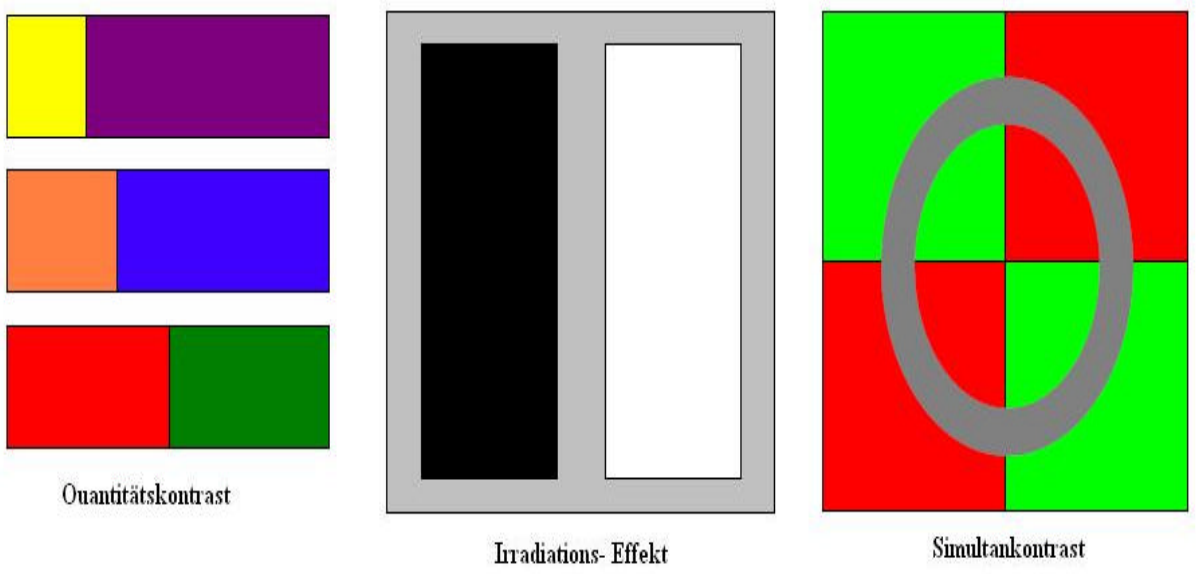


Abbildung 19 - Einfluss der Umgebung auf Farbenempfindung[SH06]

Die Wahrnehmungskapazitäten eines Menschen werden in einer Infovis hauptsächlich durch die dargestellten Formen und deren Farben beansprucht. Aus diesem Grund sind kurzgefasst folgende Regeln bei einer Infovis zu beachten:

- Ein Unterschied in der Form zweier Objekte ist einfacher zu sehen als in der Größe.
- Kontraste erleichtern genauso die Wahrnehmung wie die Beleuchtung innerhalb oberer und unterer Grenzen.
- Einfache reguläre geometrische Formen sind einfacher zu identifizieren als asymmetrische.

IV.3. Der Visualisierungsprozess

Der konzeptuelle Prozess der Datenvisualisierung kann durch eine Abfolge von drei wichtigen Schritten definiert werden: Filtering, Mapping und Rendering. Zusammen bilden sie die so genannte „Visualisierungspipeline“. Diese Pipeline setzt die zufolgende Richtlinie für die Arbeitsweise (und dadurch die Entwicklung) aller Visualisierungstools.

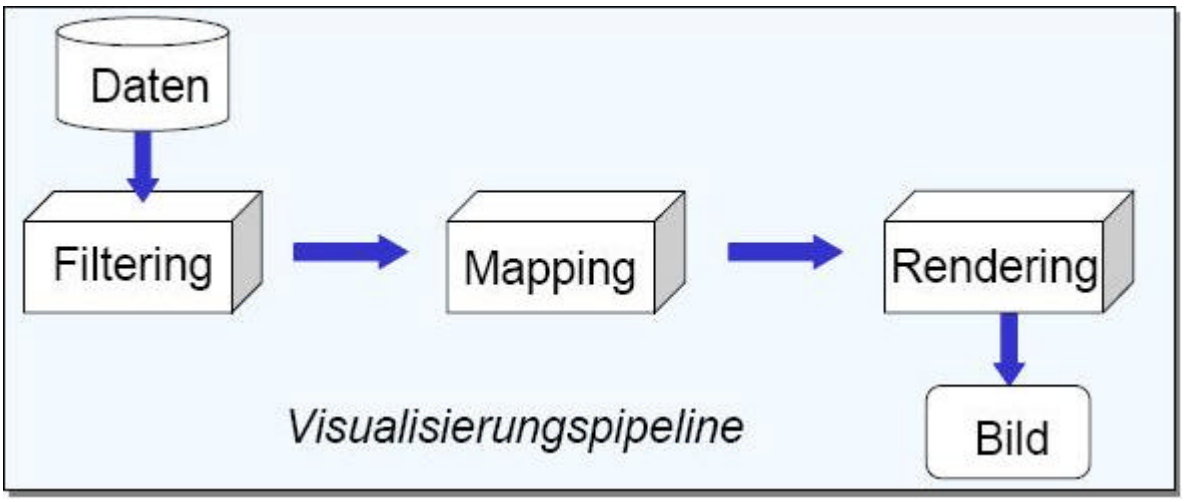


Abbildung 20 - Die Visualisierungspipeline[SH06]

Die drei Schritte der Visualisierungspipeline können folgendermaßen definiert werden:

Filtering (Datenaufbereitung)

Die Datenerfassung liefert die sogenannten Rohdaten. Ziel des Filterings ist es, die Rohdaten für die Visualisierung vorzubereiten. Üblicherweise sind Rohdaten aus Messungen, Berechnung, Datenbanken oder sogar manueller Eingabe entstanden. Nach ihrer Entstehung können sie beschädigt, unzureichend, unvollständig oder für die nachfolgenden Visualisierungsschritte einfach zu umfassend sein. Aus diesem Grund sind verschiedene Operationen notwendig, um die Vorbereitung der Rohdaten für die Visualisierung durchzuführen. In diesem Kontext wird der Begriff Filtern in einem sehr weiten Sinne verwendet. Es findet nämlich keine Filterung im wahrsten Sinn des Wortes statt, sondern es folgen Operationen mit den Daten:

- Vervollständigung
  a. Interpolation der Daten

- Reduzierung
  a. Selektion
  b. Projektion

- Glättung
  a. Fehlerkorrektur

- Bestimmung charakteristischer Eigenschaften
  a. Klassifizierung
  b. Informationsgehalt

Das Ergebnis von der Filterungphase sind aufbereitete Daten

Mapping (Daten zu Geometrie- Abbildung)

Das Datenmapping ist eine Daten-zu-Geometrie-Abbildung. In dieser Phase werden abstrakte Daten auf visuelle Attribute der geometrischen Objekte abgebildet. Unter geometrischen Objekten versteht man Kugeln, Zylinder, Oberflächengitter usw. Diese Objekte besitzen Erscheinungseigenschaften bzw. visuelle Attribute, auf welche abstrakte Datenwerte abgebildet

werden können. Diese Abbildung ermöglicht es später in der Darstellung abstrakte Datenwerte zu trennen, zu ordnen oder zu identifizieren. Unter den Erscheinungseigenschaften der geometrischen Objekte können zum Beispiel definiert werden:

- Position,
- Größe,
- Farbe und Helligkeit,
- Textur,
- Orientierung,
- Form.

Die Abbildung 21 zeigt, wie Datenwerte auf dem visuellen Attribut Farbe abgebildet werden können. Dabei werden sie auf die entsprechende Farbwerte abgebildet und der minimale Datenwert auf den ersten Wert des verwendeten Farbintervalls. Analog dazu wird der maximale Datenwert auf den letzten Farbenwert abgebildet.

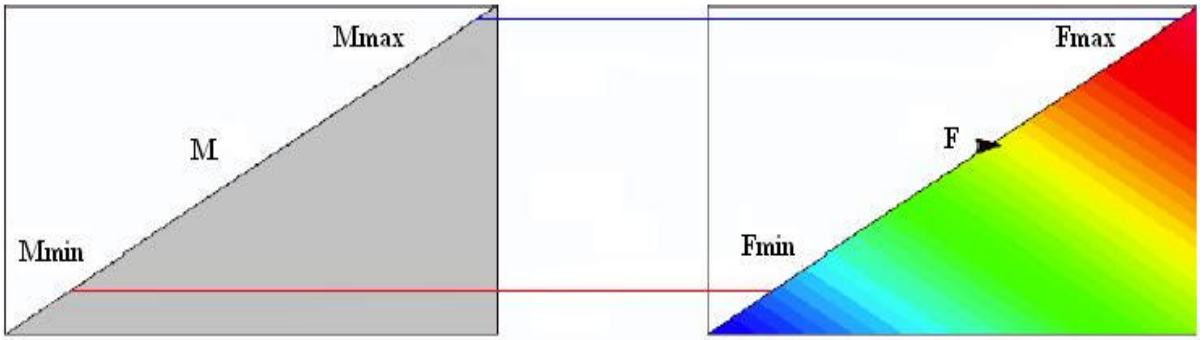


Abbildung 21 - Abbildung Datenwerten auf Farben[SH06]

Rendering: (Geometrie zu Bild- Abbildung)

Die Renderingphase funktioniert wie ein Wandler, welcher die Geometrie eines Objekts auf ein Bild auf dem Bildschirm darstellt.

Eine zusammengefasste Form der Visualisierungspipeline kann man an der Abbildung 22 erkennen.

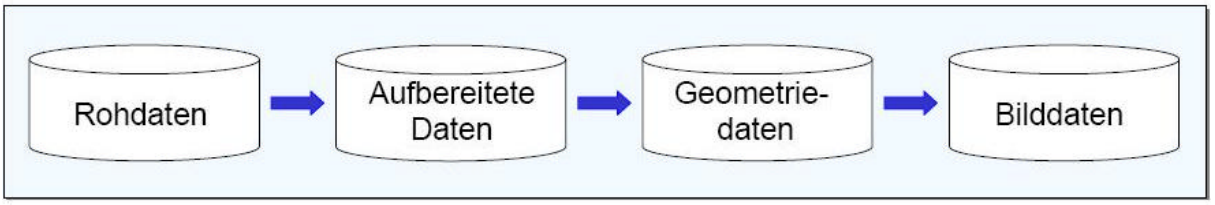


Abbildung 22 - Visualisierungspipeline Kurzfassung[SH06]

Für viele Visualisierungsaufgaben spielt die Interaktivität eine wichtige Rolle. Oft sind so genannte drill-down-Mechanismen erwünscht. Unter drill-down versteht man, dass der Benutzer auf die Daten (Roh- oder aufbereitete Daten) durch Interaktion mit dem Bild zugreift. Dass erfordert die Umkehrung der Schritte der Visualisierungspipeline, das heißt von Bilddaten in Roh- bzw. aufbereitete Daten. Dieser Umkehrweg benötigt durchdachte Datenstrukturen. Denn nur so können Verbindungen zwischen Pixeln, geometrischen Daten, Attributen, bearbeiteten Daten und den Rohdaten effizient verwaltet werden.

IV.4. Darstellungstechniken

Im Fall eine Computer-generierten Visualisierung werden die Daten, sobald sie den Visualisierungspipeline verlassen haben, auf dem Bildschirm dargestellt bzw. angezeigt. Die erzeugte Darstellung hängen sehr stark von den darzustellenden Informationen ab. Häufig kommt es vor, dass die Menge der Informationen, die visualisiert werden sollen sehr groß ist. Als wichtigstes Kriterium für die Informationsvisualisierung formuliert B. Shneiderman das so genannte Mantra der Informationsvisualisierung [SB96]: „*overview first, zoom and filter, the detail on demand*“. Basierend auf diesem Mantra wurden spezielle Techniken entwickelt, die das Problem der Darstellung großer Datenmenge lösen. Die Leitidee bei diesen Techniken ist, nicht alle Information gleichzeitig darzustellen, sondern die Genauigkeit der Darstellung „dynamisch“ variieren lassen. Somit kann der Benutzer, den relevanten Bereichen der Darstellung eine höhere Genauigkeit zuweisen. Dies führt automatisch dazu, dass die anderen Bereiche eine niedrigere Genauigkeit bekommen. Es besteht nicht die Möglichkeit, alle Darstellungstechniken im Rahmen dieser Arbeit vorzustellen. Stattdessen werden die Techniken vorgestellt, die für die Entwicklung von Generic TV angesetzt worden sind.

IV.4.1. Übersicht und Detail

Das Prinzip dieser Technik ist es, zwei Bilder aus der Datenmenge zu generieren. Die Bilder können sequentiell oder parallel angezeigt werden. Dabei wird eine getrennte Übersichts- und Detaildarstellung realisiert. Mittels der Funktion „Compress view“ in Generic TV kann man aus der detaillierten Darstellung eine neue grobe erzeugen. Das heißt, dass alle Spuren in einem neuen InternalFrame (Window) angezeigt werden.

Die Abbildung 23 zeigt die Visualisierung von Programmiertätigkeiten in Generic TV in der Detailansicht. Die Programmiertätigkeiten sind mit größeren grafischen Objekten dargestellt. Der Benutzer hat die Möglichkeit vertikal zu scrollen, um so alle vorhanden Spuren anzusehen. Die Abbildung 24 zeigt eine Übersicht derselben Datenmenge. Die Programmiertätigkeiten sind schon „schmaler“ dargestellt und alle vorhandenen Spuren sind in einer einzigen „(An)Sicht“ dargestellt. Es besteht keine Möglichkeit in der Vertikal zu scrollen.

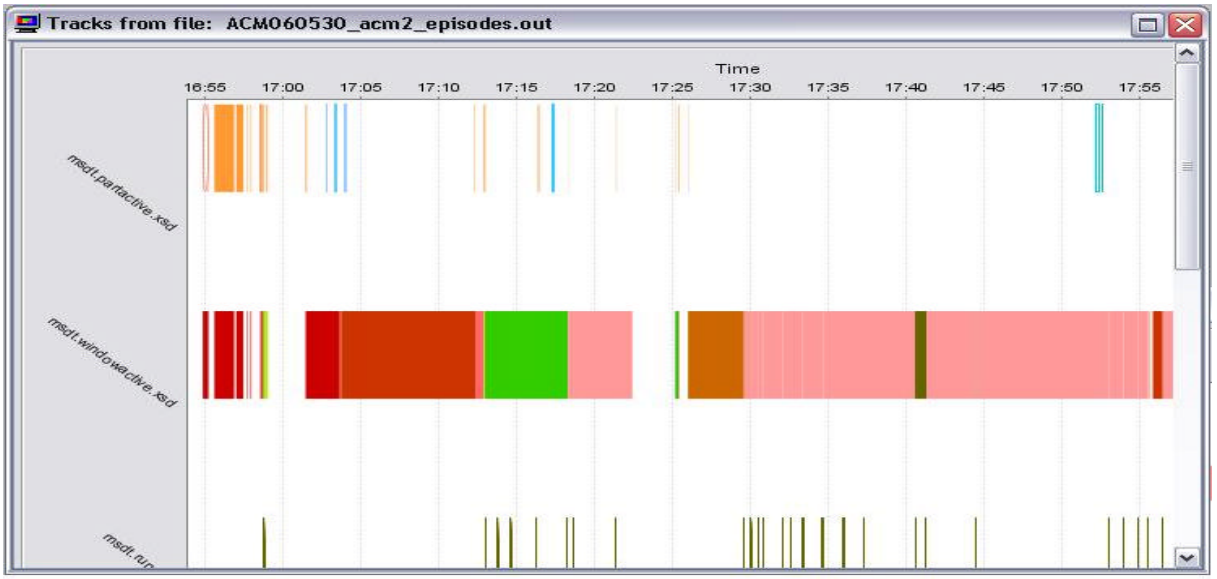


Abbildung 23 - Ausschnitt eines „Detailsicht“ aus Generic TV

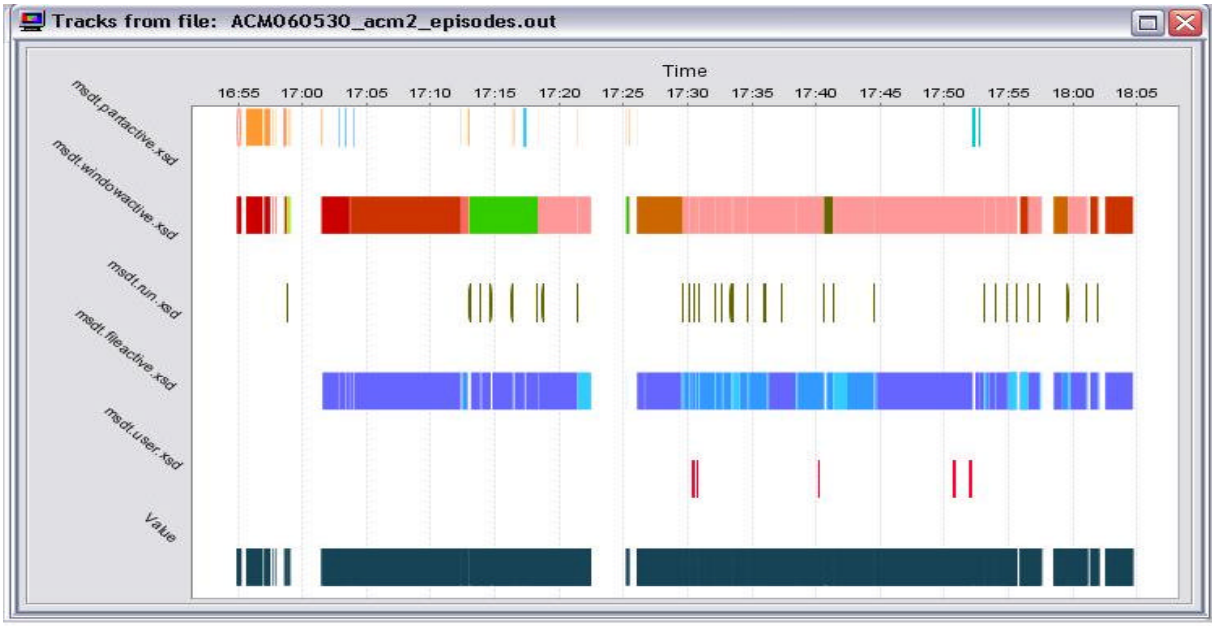


Abbildung 24 - Ausschnitt eines „Übersicht“ in Generic TV

IV.4.2. Scrolling

Die Technik des Scrollings benötigt die Nutzerinteraktion, um die aktuelle Darstellung zu verschieben. Durch die Möglichkeit zu scrollen, hat der Benutzer den Zugang zu allen dargestellten Daten. Die Abbildung 23 zeigt diese Variante in Generic TV.

IV.4.3. Information Hiding

Bei der Technik des Information Hiding wird die darzustellende Datenmenge reduziert. Alle nicht relevanten Daten werden aus der Darstellung herausgefiltert. Erst auf Benutzerwunsch werden sie dargestellt bzw. entfernt. Abbildung 25 zeigt, wie weitere Informationen eines bestimmten Datensatzes durch Mausklick eingeblendet werden.

Abbildung 25 - Information Hiding in Generic TV

IV.5. Visualisierungstechniken für zeitabhängige Daten

Die fünf menschlichen Sinne ermöglichen uns jeden Tag unsere Tätigkeiten durchzuführen. Sie werden bei der Benutzung einer Infovis genutzt, um unterschiedliche Ziele zu erreichen. Das Sehen und Fühlen ermöglichen uns z.B. eine Raumwahrnehmung. Leider haben die Menschen keinen Sinn für die Wahrnehmung der Zeit. Raum und Zeit sind zwei wichtige Faktoren für die Interpretation von zeitabhängigen Daten. W. Aigner aus der TU Wien ist der Meinung, dass es keine konkrete Definition für zeitabhängige Daten gibt. *„What is considered as time-oriented data depends on the intended task. […] a possible definition: Data, where changes over time or temporal aspects play a central role or are of interest.”* [AW04]. Die Darstellung von Daten in einer Infovis wird im so genannten Beobachtungsraum durchgeführt. In diesem Zusammenhang stellt H. Schumann folgende Definition: *„Daten weisen einen Zeitbezug auf, wenn mindestens eine Dimension des Beobachtungsraumes mit der physikalischen Größe Zeit assoziiert ist“*[SH06]. Sobald eine Zeitachse eine Dimension des Beobachtungsraums, welcher zweidimensional, dreidimensional oder mehrdimensional sein kann, ist, besteht eine Zeitabhängigkeit der dargestellten Daten. Dabei muss man aber ganz genau zwischen dem Zeitbezug der Daten und dem Zeitbezug der grafischen Repräsentation unterscheiden. Die Darstellung des Zeitbezugs der Daten ist nicht gleichzusetzen mit der physikalischen Größe Zeit. Sie lässt sich durch Annotation bzw. spezielle Kodierung einer Koordinatenachse darstellen.

IV.5.1. Visualisierung von Daten mit Zeitbezug

Die Erschaffung eines Bildes von abstrakten Daten, die einen Zeitbezug aufweisen hängt von vielen Parametern ab. Die Visualisierung solcher Daten ist den meisten Fällen durch folgendes motiviert [SM03]:

- Die Analyse von so genannten „Zeitmustern“
- die Analyse vom Verhalten der Datensätze in Abhängigkeit von der Zeit.

Der Zeitbezug der Daten hat unterschiedliche Ausprägungen. Von diesen Unterschieden lassen sich 3 Klassen (bezüglich der Zeitachse, die für ihre Darstellung benutzt wird) ableiten (Tabelle 7).

Klasse	Eigenschaften
Statisch: Datensätze haben einen statischen Bezug zur Größe Zeit. (z.B. Geburtszeiten)	Zeitachse enthält Zeitpunkte bzw. Zeitintervall, in dem die Datensätzen gültig sind. Es findet eine zeitbetzügige Annotation statt. (z.B. Geburtszeit: 19/11/1999 um 16H54)
Quasistatisch: Datensätze haben einen quasistatischen Bezug zur Größe Zeit. (z.B. eine einmalige Messung jedes Monats)	Die Zeitachse enthält mehrere Zeitpunkte oder Zeitintervalle, wobei angenommen wird, dass die Daten in kleinen Zeitintervallen konstant sind (z.B. 4 Messungen pro Jahr).
Dynamisch: Datensätze haben einen statischen Bezug zur Größe Zeit. (z.B. kontinuierliche Messungen)	Die Zeitachse wird kontinuierlich skaliert. (z.B. Temperaturabmessungen oder Simulationen)

Tabelle 7- Klassen des Zeitbezugs der Daten

Im Gegensatz zum Zeitbezug der Daten hat der Zeitbezug der Darstellung, also die Darstellung der Zeit, nur zwei Ausprägungen. Dies führt wiederum zu zwei Repräsentationsklassen:

Klasse	Eigenschaften
statische Repräsentation	Es gibt keine automatische Veränderung der dargestellten Daten über die Zeit. Änderungen resultieren nur durch Interaktion des Benutzers. (z.B. Einzelbilder oder Bildsequenzen)
dynamische Repräsentation	Die grafische Repräsentation der Daten ändert sich automatisch übe die Zeit. (z.B. Animationen)

Tabelle 8 - Klassen des Zeitbezugs der Darstellung der Daten

Es ist leider im Rahmen dieser Arbeit nicht möglich, die aus Tabelle 7 und 8 resultierenden sechs möglichen Darstellungen von Daten zu präsentieren. Da Generic TV eine statische Repräsentation von abstrakten Daten realisiert, werden hier nur einige Beispiele der statischen Repräsentation vorgestellt.

Statische Repräsentation

- Statische Daten:

  Für Daten, die sich über die Zeit nicht ändern, ist eine statische Repräsentation eine sehr adäquate Darstellungsform. Der Bezug der Daten zur Zeit lässt sich sehr einfach über eine Annotation verdeutlichen. Ein aussagekräftiges Beispiel für statische Darstellung von statischen Daten ist die Kalenderdarstellung (van Wijk, van Selow, 1999). In dem Beispiel wird das Grundkonzeptes eines Kalenders, um über eine Achse Zeitpunkte und die dazugehörigen Daten darzustellen, angewendet. Eine spezielle

Form der Kalenderdarstellung ist das Gantt-Diagramm das schon im Kapitel III vorgestellt wurde.

- Quasistatische Daten:

  Solche Daten lassen sich, wie die statische Daten, durch eine statische Repräsentation sehr gut darstellen. Ein bekanntes Beispiel dafür ist die Mehrfenstertechnik-Darstellung. Bei dieser Technik werden mehrere Bilder generiert, die eine Evolution über eine bestimmte Zeit darstellen sollen. Um aber eine Orientierung über den zeitlichen Verlauf zu gewinnen, müssen besondere Techniken eingesetzt werden, um so eine Anordnung der Fenster zu garantieren (z.B. Filmrollen wie in unten abgebildet)

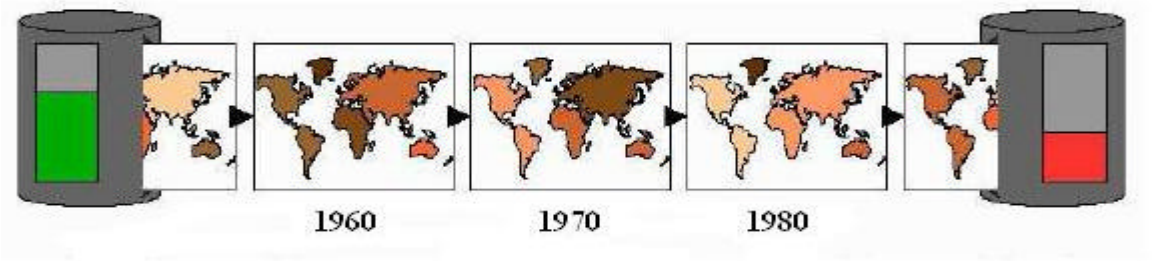


Abbildung 26- Mehrfenstertechnik mit Anwendung der Filmrollen für die Orientierung[SH06]

- Dynamische Daten:

  Eine statische Repräsentation von dynamischen Daten ist sehr geeignet, um quantitative Aussagen in der Darstellung zu generieren. Darstellungen von Datenwerten einer Datenmenge durch grafische Objekte sind quantitativen Aussagen. Ein gutes Beispiel dieser Darstellungstechnik ist der so genannte ThemeRiver (von Havre 2000). Der ThemeRiver ist eine Visualisierungsmethode, welche die sich veränderten Daten über die Zeit darstellt. Es ist eine Art „Fluss“. Die Datenwerte „fließen“ kontinuierlich entlang der Zeitachse und Veränderungen werden durch ihre Höhen-Darstellung kontinuierlich widergespiegelt (siehe Abbildung 27).

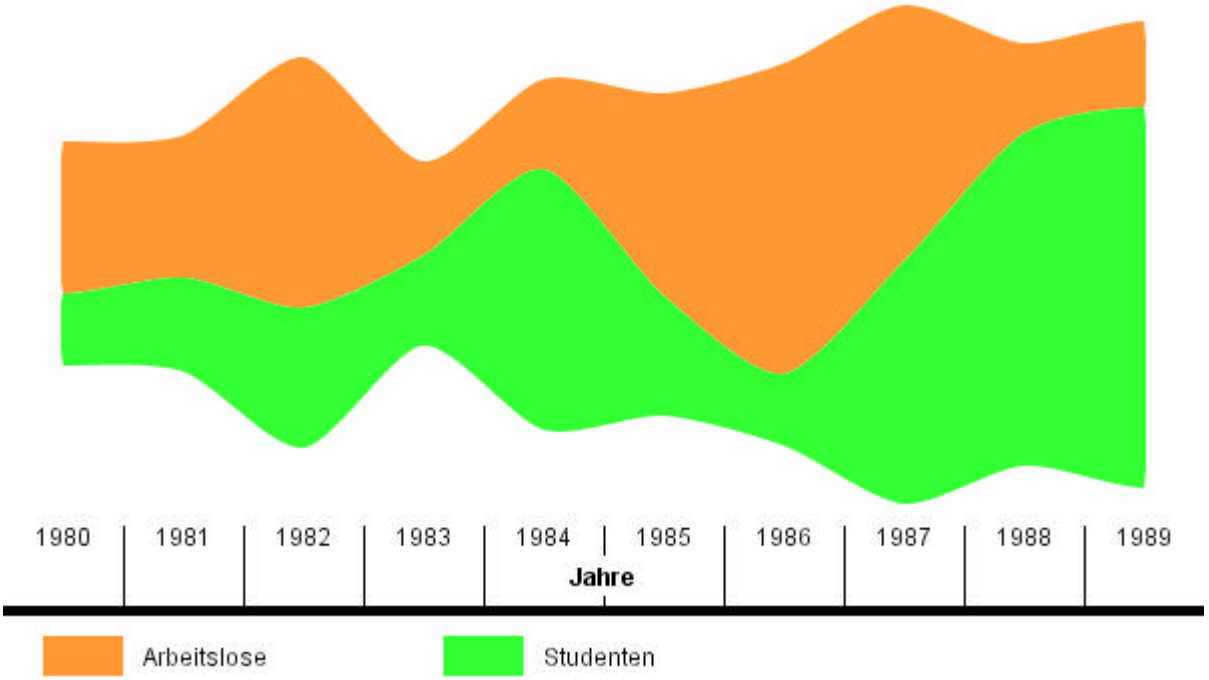


Abbildung 27 - Fiktiver Vergleich zwischen der Anzahl der Arbeitslosen und der Anzahl der Studenten[TR_I]

IV.5.2. Visueller Attribute zur Darstellung von Datenwerten. Quantitative Darstellung

Eine „gute“ Infovis sollte in der Lage alle Eigenschaften der Datenmenge zu repräsentieren. Aus diesem Zweck findet in der Mappingphase der Visualisierungspipeline die Abbildung bzw. Zuweisung von abstrakten Daten und ihrer Werte auf grafische Elemente statt. Durch diese Abbildung ist die Realisierung einer so genannten quantitativen Darstellung der Datenmenge möglich. Dabei werden die Datenwerte der Datensätzen durch visuelle Variablen dargestellt. Um eine optimale Abbildung dieser Daten zu bekommen, hat Bertin im Jahr 1982 die Theorie der Grafischen Semiotik aufgestellt [SH06]. Er definierte in seiner Theorie acht visuelle Variable. Diese können zur Erstellung von Bildern eingesetzt werden:

- Größe,
- Helligkeitswert,
- Farbe,
- Richtung (Orientierung),
- Form des Elements,
- Musterung (Textur),
- Position auf der Ebene.

Nach Bertin soll jede einzeln definierte Variable besondere Wirkungen in der Darstellung verkörpern [SH06]:

- Selektiv: unterschiedliche Werte werden vom Betrachter spontan in Gruppen aufgeteilt.
- Ordinal: unterschiedliche Datenwerte werden spontan in eine Ordnung gebracht.
- Proportional: unterschiedliche Datenwerte werden nicht nur spontan in eine Ordnung gebracht, sondern zusätzlich erlauben diese Variablen auch eine Assoziation mit den konkreten Werten.

Datenwerte der Datensätzen lassen sich auf alle durch Bertin definierten visuellen Variablen abbilden und dadurch darstellen. Es ist dann für den Betrachter der Darstellung möglich, eine visuelle Repräsentation der dargestellten Datenwerte zu bekommen. Dadurch lässt sich eine Analyse der Datenmenge besser durchführen. Es ist nicht möglich im Rahmen dieser Arbeit Beispiele von allen Abbildungen von Datenwerten zu Variablen vorzustellen. Aus diesem Grund werden nur die in Generic TV angewendeten Techniken und einige weitere vorgestellt.

Abbildung der Datenwert auf die Größe

Für die Größenabbildung gibt es unterschiedliche Möglichkeiten. Der Flächeninhalt eines Kreises oder die Höhe eines Rechtecks können sehr deutlich den Wert eines Datensatzes grafisch darstellen (Siehe Abbildung 28 und 29).

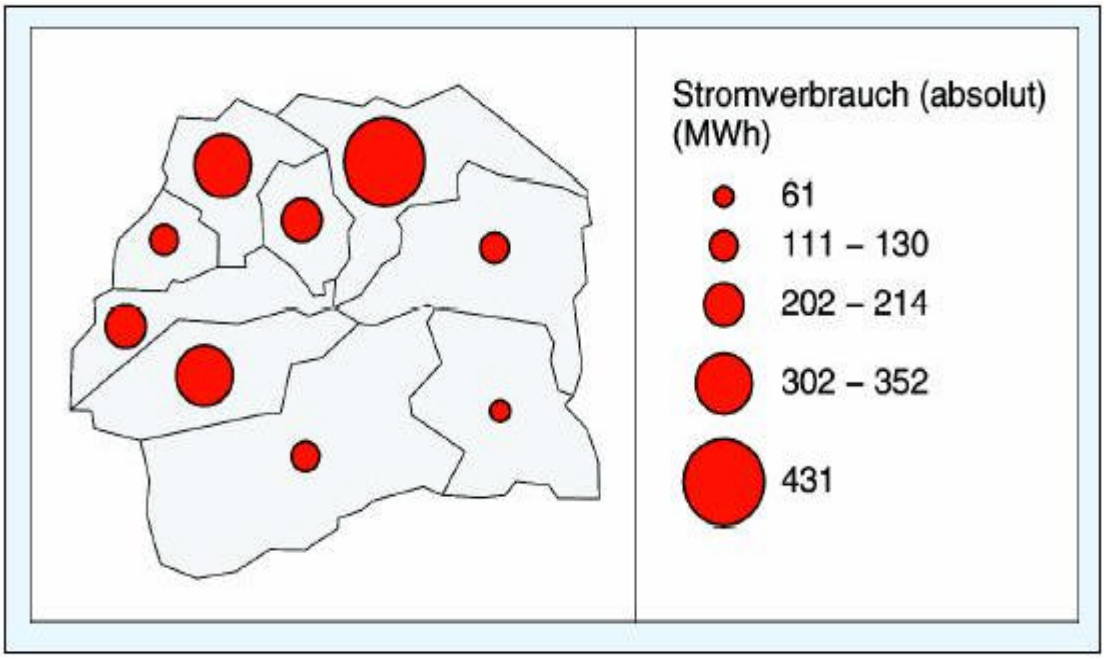


Abbildung 28 - Darstellung einer Karte mit Kreissignatur [SH06]

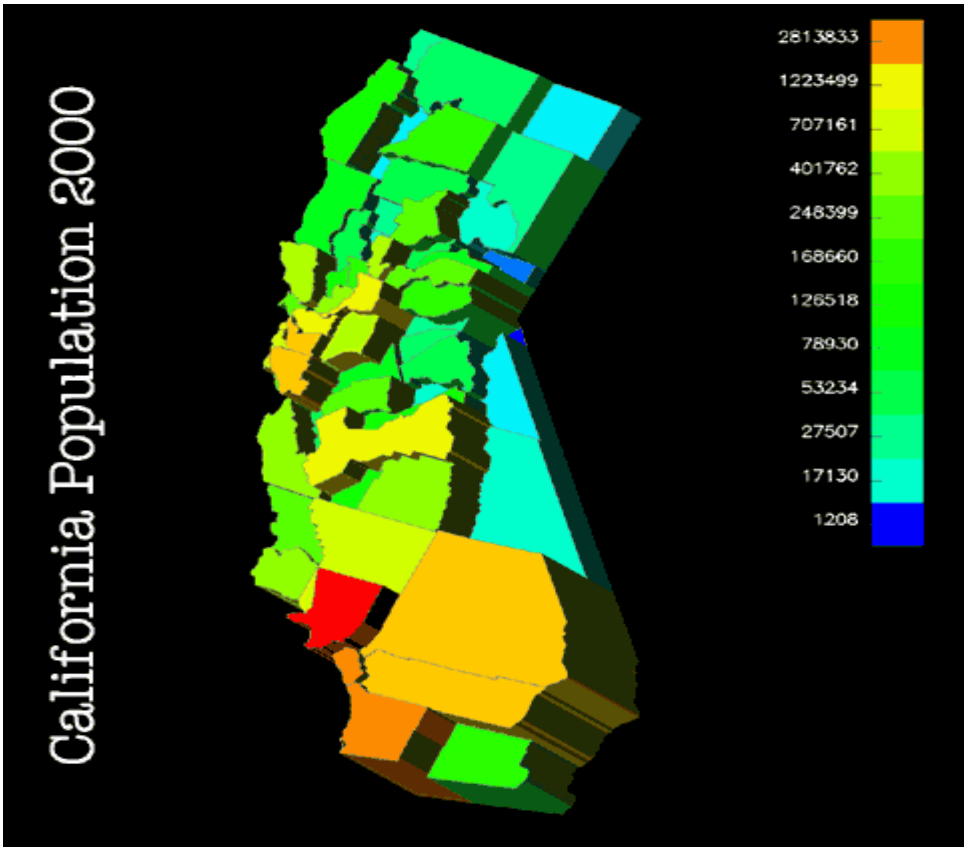


Abbildung 29 – Fiktive quantitative Darstellung der Kalifornischen Bevölkerung durch eine unterschiedliche Höhe der grafischen Elemente und eine Kodierung der Datenwerten durch Farben [MJ99]

Abbildung der Datenwert auf die Farbe

Farbe ist eine der wichtigsten visuellen Variablen für die Abbildung von Datenwerten. Dies liegt daran, dass Farbe in einer sehr frühen Stufe der menschlichen Wahrnehmung erkannt wird. Diese frühe Wahrnehmung des Menschen nutzt die Farbe, als besten Garant für die Effektivität einer Visualisierung. Farbe wird hauptsächlich in einer Darstellung zur:

- Unterscheidung von Datenwerten,
- Betonung von Elementen und Regionen,
- Repräsentation von symbolischen Assoziationen eingesetzt.

Die Benutzung von Farbe ist sehr geeignet für die Ordnung von Datensätzen. Sie übernimmt also eine „ordinale“ Funktion. Man kann mit der Anwendung von Farbe die Zugehörigkeit zu einer „Klasse“ von Datensätzen herausheben. Durch z.B. das Variieren der Helligkeit der spezifischen Klassenelemente, lassen sich Unterschiede in einer Klasse eindrucksvoll veranschaulichen. Die Abbildung 30 zeigt eine Anwendung der Farbe als „ordinale“ Funktion in Generic TV. In diesem Beispiel werden mit der Farbe Blau Javadateien klassifiziert und dargestellt. Ein Variieren der Farbe Blau macht den Unterschied der jeweiligen Javadateien erkennbar.

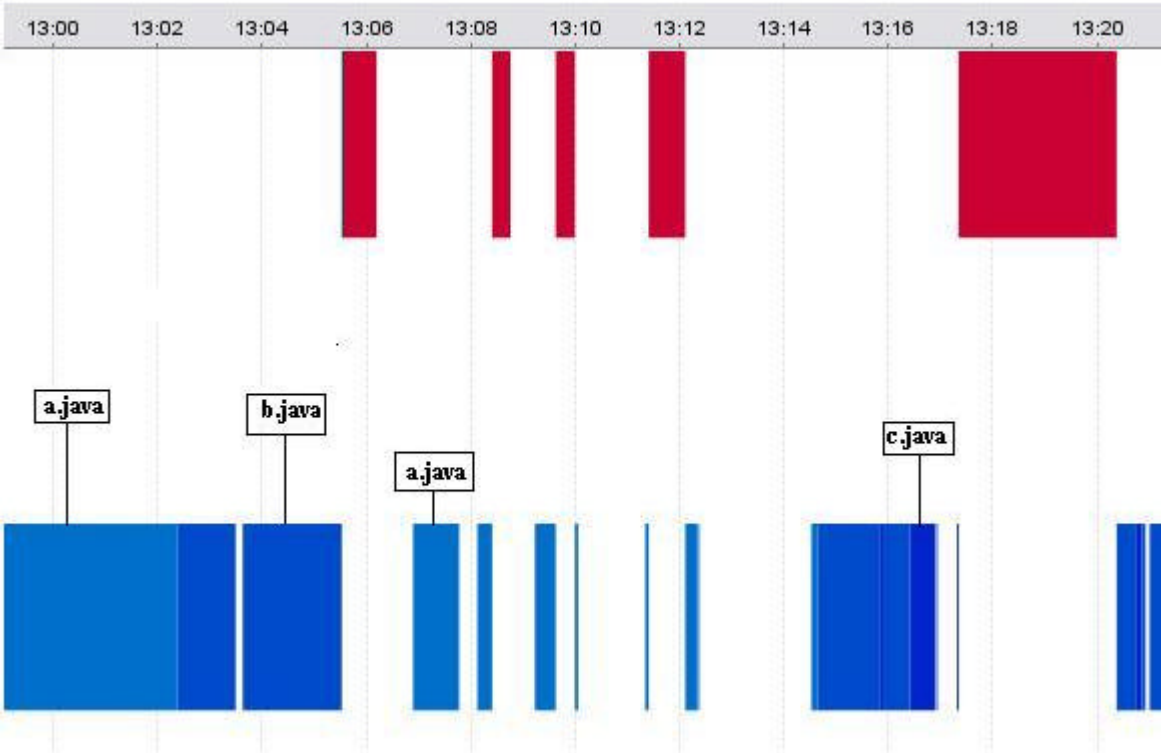


Abbildung 30 - Anwendung der ordinalen Funktionalität der Farbe in Generic TV

Quantitative Daten[30] können ebenso farblich abgebildet werden. In diesem Fall muss nur der Wertbereich auf einen Ausschnitt einer Farbskala abgebildet und dargestellt werden (Siehe Abbildungen 21 und 31).

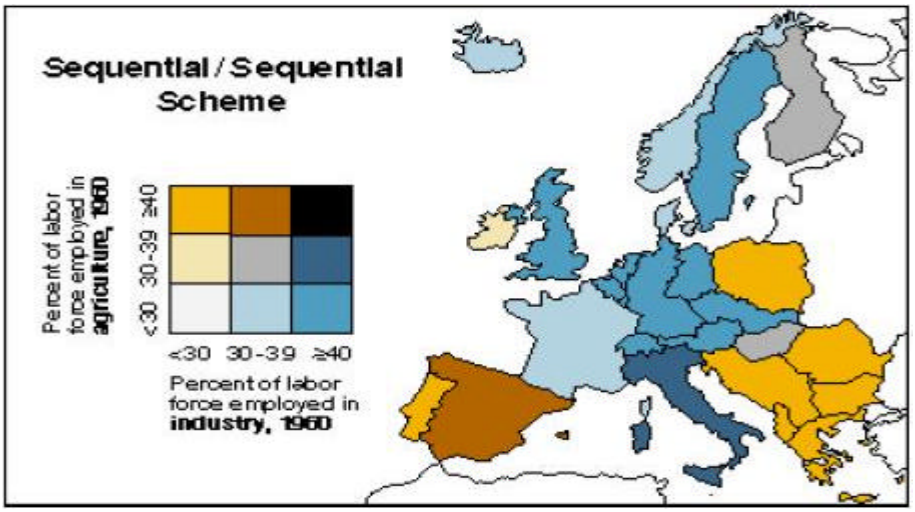


Abbildung 31 - Quantitative Darstellung durch Bivariatefarbskalenmethoden [SH06]

Es existiert eine weitere Abbildungsmöglichkeit der Datenwerte auf visuelle Variablen. Ein Beispiel dafür wäre die Abbildung durch Formvariation, Texturen oder ähnliches. Spence hat im Jahr 2001 ein Bild vorgestellt, welches alle Abbildungsmethoden zusammenfasst (Siehe Abbildung 32).

[30] Daten, die Datenwerten besitzen

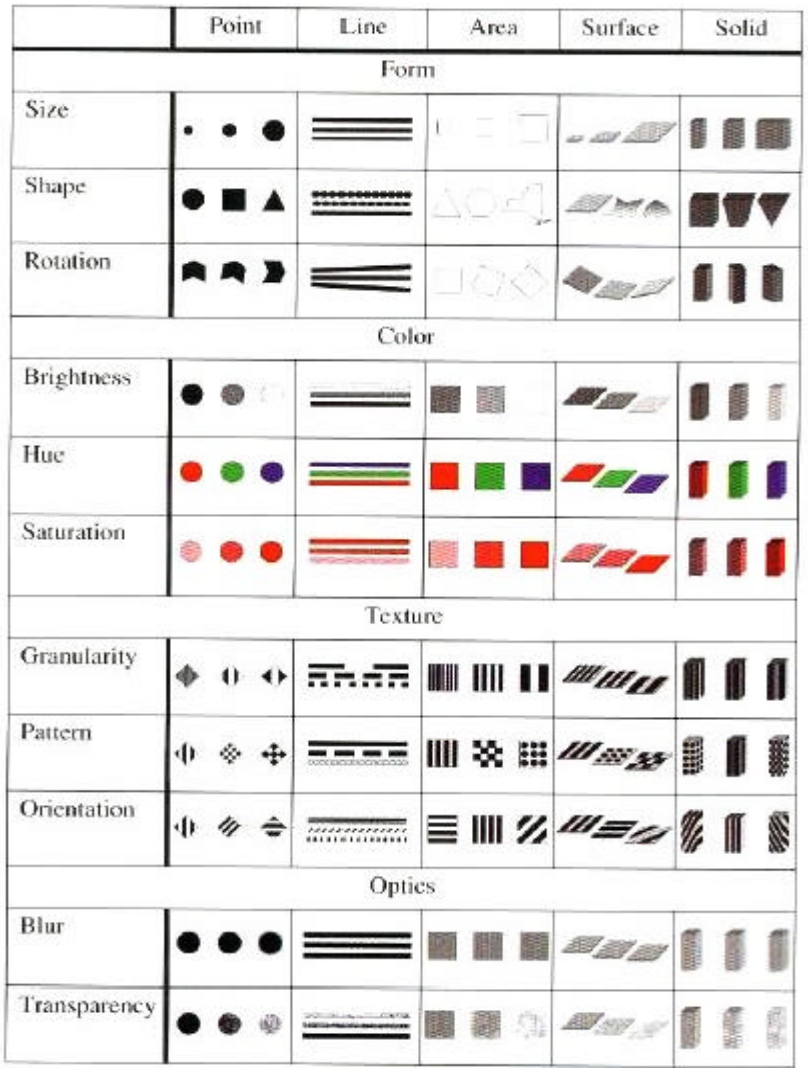


Abbildung 32 - Zusammenfassung der verschiedenen Abbildungsmethoden[SH06]

Die Erstellung einer Visualisierung ist ein kreativer, iterativer und interaktiver Prozess. Um die Eigenschaften einer Datenmenge zu ermitteln, braucht es den Einsatz von unterschiedlichen Darstellungstechniken. Dadurch sind unterschiedliche Ansichten auf die Datenmenge möglich. Im besten Falle bringt dies unterschiedlichste Aspekte der Datenmenge ans Licht. B Schneidermann hat im Jahr 1996 das so genannte Mantra der Infovis vorgestellt. Dabei werden die „Sieben Goldenen Regeln einer Informationsvisualisierung“ wie folgt gegliedert:

- Der Überblick (overview): Sich einen Gesamtüberblick über die darzustellende Datenmenge verschaffen.
- Zoom: Rein- bzw. Rauszoomen in/aus bestimmte(n) Bereichen der Darstellung.
- Filtern: Herausfiltern von (zu einem bestimmten Zeitpunkt) nicht relevanten dargestellten der Daten.
- Einzelheiten der Anforderungen (details on demand): Selektieren eines oder mehrerer dargestellte Elemente und evtl. Detaillierung.
- Beziehungen (relate): Die Darstellung von eventuellen Beziehungen zwischen den dargestellten Datensätzen.

- History: Die Historie von den durchgeführten Tätigkeiten verwalten („undo“).
- Speichern (extract): Die Möglichkeit haben einen Ausschnitt aus den dargestellten Elementen des Datenbestands in einem Formaten zu speichern bzw. auszudrucken.

V. Analyse und Konzeption der Generic TV

In diesem Kapitel werden die wichtigsten Schritte des Entwurfs und der Konzeption von Generic TV vorgestellt. Die Inputdaten sind entscheidende Punkt der Visualisierung. Die bestmöglichste Lösung für das Problem, wie sich Programmiertätigkeiten visuell darstellen lassen, zu finden, wird in ersten Abschnitt dieses Kapitels behandelt. Der Abschnitt V.2 stellt den Arbeitsablauf mit Generic TV dar. Die wichtigsten Funktionalitäten der Software sind dort gegliedert. Der dritte Abschnitt des Kapitels gibt eine konzeptuelle Beschreibung des Entwurfs von Generic TV. Desweiteren erfolgt eine Beschreibung der Abstraktionsschichten. Diese Abstraktionsschichten repräsentieren bzw. führen alle wichtigen Phasen des Visualisierungsprozesses (durch).

V.1. Analyse

Die Kapitel II und III beschrieben genau die Begriffe Programmiertätigkeiten bzw. Programmieraktivitäten. Es wurde auch erklärt, dass diese Tätigkeiten die Inputdaten der Generic TV darstellen. Dabei haben die Tätigkeiten bestimmte Eigenschaften. Sie müssen zusammen mit ihren Werten örtlich dargestellt werden. H. Schumann von der Universität Rostock gab in einer Vorlesung über Visualisierung genaue Beschreibungen für Begriffe wie „Tätigkeiten“, „Eigenschaften“ und Örtlichkeit. Sie definierte genaue Abgrenzungen zwischen Begriffen. Die allgemeine Festlegung von H. Schumann lautet folgendermaßen [SH06]:

- *Daten* bezeichnen abstrakte Größen, die unterschiedliche Werte repräsentieren. Durch Zuordnung einer Semantik und Pragmatik werden Daten zu Informationen. Diese sind in einem gewissen Kontext interpretierbar.

- *Informationen* können als Objekte repräsentiert werden, denen Attribute zugeordnet sind. Die Attribute stehen in gewissen Beziehungen.

- *Informationsobjekt* **IO** ist eine Abstraktion von Daten, welche eine gewisse Information repräsentieren

- Die *Informationsmenge* **IM** ist eine diskrete Menge von Informationsobjekten

$$\mathbf{IM = \{IO_1, IO_2, ..., IO_n\} \text{ mit } IO_i = IO_j \;?\; i = j \text{ und } i, j, n \;\hat{I}\; N}$$

- Die *Attributmenge* **AM** umfasst alle Attribute, die einer Informationsmenge zugeordnet sind

$$\mathbf{AM = \{A_1, A_2 ,...A_m\} \text{ mit } A_i = A_j \;?\; i = j \text{ und } i, j, m \;\hat{I}\; N.}$$

- Die *Informationsstruktur* **IS** steht in Relation zur Informationsmenge IM. Die Elemente von IS repräsentieren die Beziehungen der Informationsobjekte untereinander. Diese können entweder explizit oder implizit gegeben sein.

$$\mathbf{IS = IM \times IM}$$

- Die *Attribute* der Attributmenge **AM** definieren den *Informationsraum* **IR**. Hierbei wird für jedes Attribut A_i eine Dimension von IR aufgespannt und entsprechend des zugehörigen Wertebereichs skaliert. Die Dimensionalität von IR ergibt sich aus der Kardinalzahl von AM, also

$$\mathbf{dim(IR) = |AM|.}$$

Im oberen Abschnitt dieses Kapitels wurde erläutert, dass die Programmiertätigkeiten bestimmte Eigenschaften besitzen. Die Schumannschen Begriffsdefinitionen können bei der örtilichen Abbildung der Werte und Eigenschaften von Programmiertätigkeiten angewendet werden.

Die Abbildung 33 stellt eine zusammenhängende Darstellung der eingeführten Begriffe.

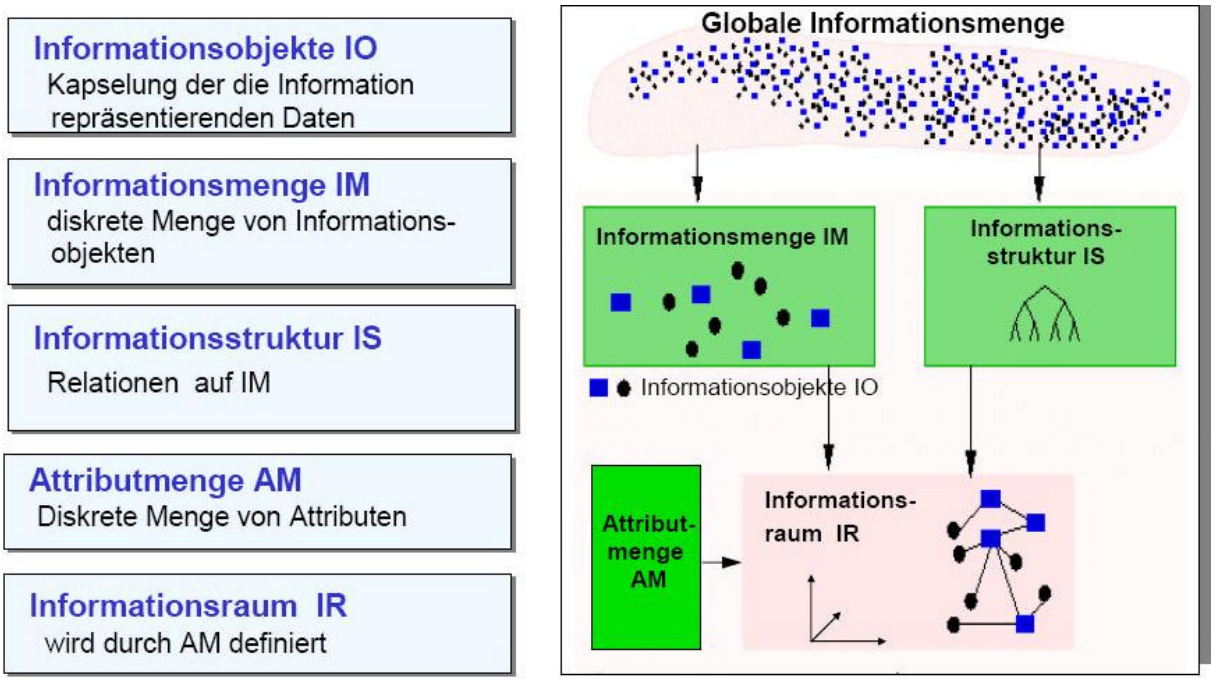


Abbildung 33 - Zusammenfassende Veranschaulichung der Begriffsbildung [SH06]

In der Informationsvisualisierung werden Informationsobjekte, ihre Attributausprägungen und Beziehungen grafisch veranschaulicht. In Kapitel IV wurde erwähnt, dass die immer größer werdende Datenmenge für Infovis problematisch ist. Dies erklärt die Komplexität des Informationsraum IR. Die Entwicklung spezieller Darstellungstechniken[31] waren deshalb vonnöten, um eine brauchbare Infovis zu erstellen. Der Grundidee dieser Techniken ist, nur die notwendigen Informationen mit einer sehr hohen Genauigkeit darzustellen. Die allgemeinen Kontextinformationen werden durch einen groben Überblick wiedergegeben. Die Realisierung einer Infovis benötigt also eine genau Analyse der Inputaten und ihrer Eigenschaften, damit diese Daten in der Welt der Visualisierung eingeordnet werden können
Das in Kapitel III vorgestellte Metamodell bestimmt die Informationsobjekte der für dieses Projekt vorhandenen Datenmenge.

[31] Z.B. „Übersicht und Detail“, Scrolling und Information Hiding. Sie sind im Abschnitt IV.1 erklärt.

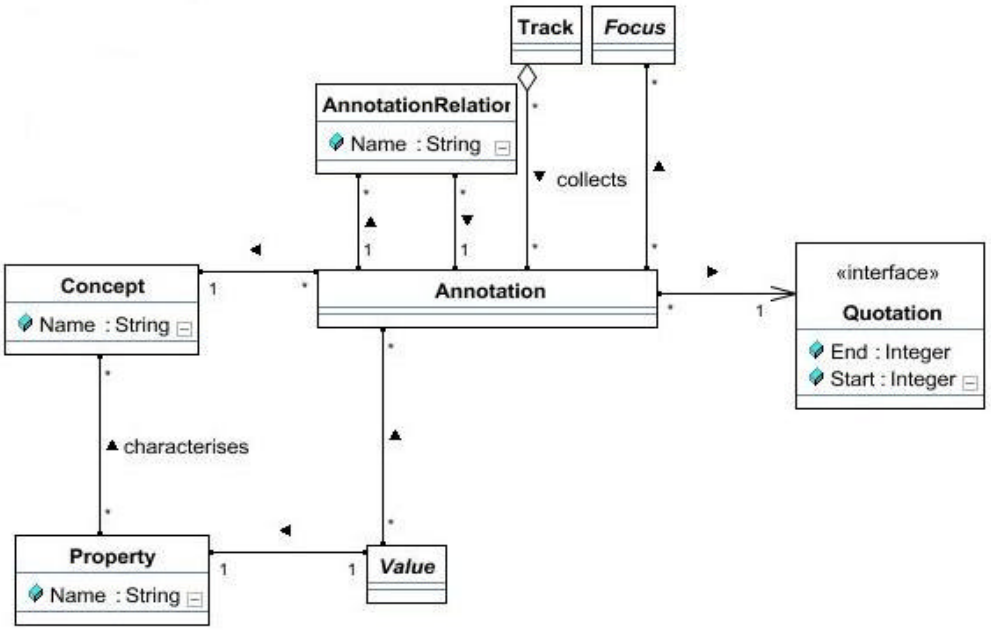


Abbildung 34 - Abzug aus dem Metamodell der generischen Schnittstelle von Generic TV. Modellierung der zu visualisierende Informationsobjekte[32]

Aus der Abbildung 34 kann man erkennen, dass einTrack (Spur) aus Annotationen besteht. Desweiteren ist aus dem Metamodell erkenbar, dass eine dargestellte Spur in Generic TV aus Concepts, Values, Focus und AnnotationRelations bestehen kann.

Die Umstände dieser Phase der Softwareentwicklung machen es meiner Meinung nach nötig, eine Stellungnahme abzugeben. Hiermit möchte ich ausdrücklich betonen, dass die Begriffe Programmieraktivität sowie Programmiertätigkeit nicht im ursprünglichen Wortsinne zu verstehen sind. Die Semantik dieser Ausdrücke kann im Rahmen dieser Diplomarbeit nur entstellt wiedergegeben werden. Der Titel der Arbeit spricht von „Programmieraktivitäten“. Aus dem obigen Metamodell ist aber erkenbar, dass eine dargestellte Spur in Generic TV aus Concepts, Values, Focus und AnnotationRelations bestehen kann. Dies widerpricht dem eigentlichen[33] Sinn des Wortes. In diesem Projekt werden die Begriffe Programmiertätigkeiten bzw. Programmieraktivitäten mit den darzustellenden Informationsobjekten gleichgestellt[34] .

Einzeln betrachtet besitzt jede Programmieraktivität zwei wichtige Charakteristika

1. Sie wird zu einem bestimmten Zeitpunkt durchgeführt,

2. Sie besitzt eine Menge von weiteren Informationen, wie z.B. ihre Beschreibung oder ihr Name, ihre Dauer etc. Diese weiteren Informationen werden als Menge Eigenschaft zusammengefasst.

Die Abbildung 35 zeigt ein einfaches Modell einer Programmieraktivität und ihre Charakteristika.

[32] Die dargestellten Begriffe wurden im Kapitel III genau definiert
[33] In der realen Welt
[34] Analog zur Abbildung 35

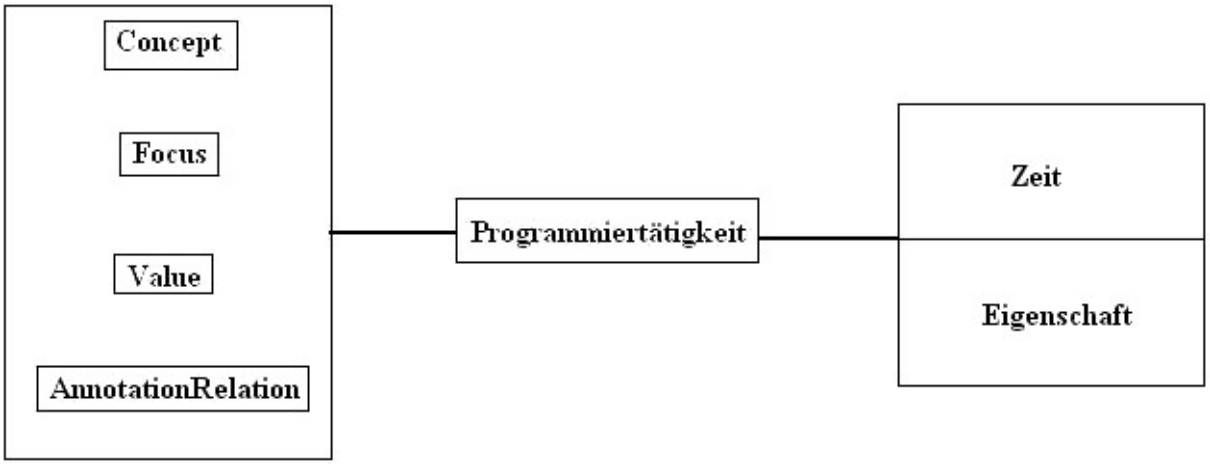


Abbildung 35 - Programmiertätigkeit und ihre Komponente als einfaches Modell

Diese Modellierung der Programmiertätigkeit ermöglicht ihre leichtere Überführung in die Begriffswelt der Infovis. Durch die von H. Schumann festlegten Begriffe Informationsmenge und Informationsobjekt, lassen sich die Termini *Programmiertätigkeitsmenge* **PM** und *Programmiertätigkeitsobjekt* **PO** herleiten. Analog zur Schumann-Definition ergibt sich, dass **PM** eine diskrete Menge von **PO** ist. Es gilt:

$$\mathbf{PM = \{PO_1, PO_2, ..., PO_n\} \text{ mit } PO_i = PO_j \;?\; i = j \text{ und } i, j, n \hat{I} N}.$$

Jedes Programmiertätigkeitsobjekt **PO** repräsentiert eine einzelne Programmiertätigkeit[35]. Sie wird durch die Elemente ihrer Menge Eigenschaft und der Größe Zeit beschrieben.

Die Attributmenge **AM** besteht im Falle des Programmiertätigkeitsobjekts aus den Elementen Zeit und Eigenschaft. Für die Attributmenge folgt:

$$\mathbf{AM_{Programmiertätigkeit}} = \{\text{Zeit, Eigenschaft}\}.$$

Aus der Attributmenge $\mathbf{AM_{Programmiertätigkeit}}$ ergibt sich die Dimension des Informationsraums. Sie lässt sich folgendermaßen berechnen:

$$\mathbf{Dim(IR) = | AM_{Programmiertätigkeit} | = 2}.$$

Aus dieser Herleitung ergibt sich, dass der Informationsraum einer Programmiertätigkeit durch die Attributen „Zeit“ und „Eigenschaft“ bestimmt ist. Zur Visualisierung der Programmiertätigkeiten müssen die Programmiertätigkeitsobjekte im Informationsraum dargestellt werden. Jede einzelne Programmiertätigkeit muss durch ein Programmiertätigkeitsobjekt repräsentiert werden. Da der Informationsraum der Programmiertätigkeit vom Attribut „Zeit“ abhängt, ergibt sich die Notwendigkeit der zeitlichen Abbildung in Generic TV.

In den Abschnitten IV.4 und IV.5 wurden bestimmte Raum- und Zeitdarstellungstechniken vorgestellt. Eine Analyse für ihre Umsetzung in Generic TV wird hier vorgestellt.

V.1.1. Analyse und Umsetzung der Raumdarstellungstechniken

[35] Also ein Concept, Focus, Value und AnnotationRelation.

Der Darstellungsraum muss in einer Visualisierung die Informationsobjekte enthalten. Für Generic TV bedeutet dies, dass im Informationsraum Programmiertätigkeitsobjekte abgebildet werden. Das Kernproblem, die Visualisierung der enormen Menge von Informationsobjekten, wurde in dieser Arbeit mehrfach erwähnt. Eine Zusammenfassung der Lösungsansätze ist, die Menge von darzustellenden Informationsobjekte zu reduzieren und dabei die Genauigkeit der Darstellung dynamisch variieren lassen. Ein Darstellungsraum kann auf zweierlei Weise Informationsobjekte wiedergeben:

1. Bei einer vollständigen Darstellung werden alle Informationsobjekte und ihre Eigenschaften dargestellt. Für Generic TV würde dies bedeuten, dass alle Programmiertätigkeitsobjekte und der Inhalt der Menge Eigenschaft für jedes einzelne Programmiertätigkeitsobjekt (Siehe Abbildung 35) dargestellt werden müssten. Für jedes einzelne Programmiertätigkeitsobjekt müsste unter anderem folgendes abbilden werden:

   a. Das Programmiertätigkeitsobjekt selbst,
   b. seine Beschreibung,
   c. seine Name,
   d. seine Dauer,
   e. die Informationen über die dazugehörige ConceptClass,
   f. seine AnnotationRelation usw.

2. Bei unvollständiger Wiedergabe werden nur Teile der Programmiertätigkeitsobjekte und der Inhalt der Menge Eigenschaft ausgegeben. Nur durch Interaktionen des Benutzers werden weitere Informationsobjekte dargestellt.

Der Vergleich beider Möglichkeiten lässt schon erahnen, dass bei der vollständigen Darstellung die Darstellungsfläche schnell verbraucht ist. Für Generic TV ist deshalb nur die unvollständige Darstellung der Programmiertätigkeitsobjekte praktikabel. Im Abschnitt IV.4 wurden Raumdarstellungstechniken vorgestellt. „Detail und Übersicht“, „Scrolling“ sowie „Information Hiding“ sind meiner Meinung nach die Techniken, die am geeignetsten für Generic TV-Darstellung sind.

Es werden in einem Hauptfenster[36] interne Fenster[37] generiert. Letzterer beinhaltet dann die dargestellten Programmiertätigkeitsobjekte. Siehe dazu die Abbildung [38]36.

Jedes interne Fenster visualisiert eine vom Benutzer gewählte Inputdatei. Es besteht die Möglichkeit, weitere Inputdateien in eine Darstellung zu importieren. Dem Benutzer soll so die Gelegenheit gewährt werden, jede Datei der Inputdaten sich einzeln darstellen zu lassen. Diese Eigenschaft der Software bringt zwei Vorteile hervor:

1. Mehr Inputdaten können dargestellt werden.

2. Eine bessere Analyse der Daten ist durchführbar.

Es bot sich die Anwendung der Darstellungstechnik „Detail und Übersicht“ an. Das InternalFrame-Verfahren bietet eine Detail- und eine Grobansicht der Inputdaten. In der detaillierten Ansicht gibt Generic TV Programmiertätigkeitsobjekte größer (und dadurch besser sichtbar) aus.

[36] dem Hauptfenster der Applikation
[37] sogenannte InternalFrames
[38]

Eine Spur wurde als das Auftreten der Programmiertätigkeiten definiert. Die Programmiertätigkeiten sind durch ihre Anfangs- und Endzeitpunkte charakterisiert. Entstehende Überlagerungen von Programmiertätigkeiten waren ein großes Problem. Dieses Überlappen ist dadurch gekennzeichnet, dass sich ein Tätigkeitsende mit der Anfangszeit der zweiten Programmiertätigkeit sich überschneidet. Also das zum Beispiel die Programmiertätigkeit A um 13:35:15[39] aufhört und die nächste darzustellende Programmiertätigkeit B schon um 13:35:05 anfängt. Durch eine Detailansicht können diese Überlappungen besser erkannt werden. Weil die einzelnen Programmiertätigkeiten vergrößert sind. Es ist an dieser Stelle wichtig zu erwähnen, dass die Technik „Detail und Übersicht" allein nicht ausreicht, um das Problem der Überlappung zu lösen. Im Abschnitt VI.1 dieser Ausarbeitung werden das Problem der Überlappung und ein Lösungsansatz ausführlich besprochen.

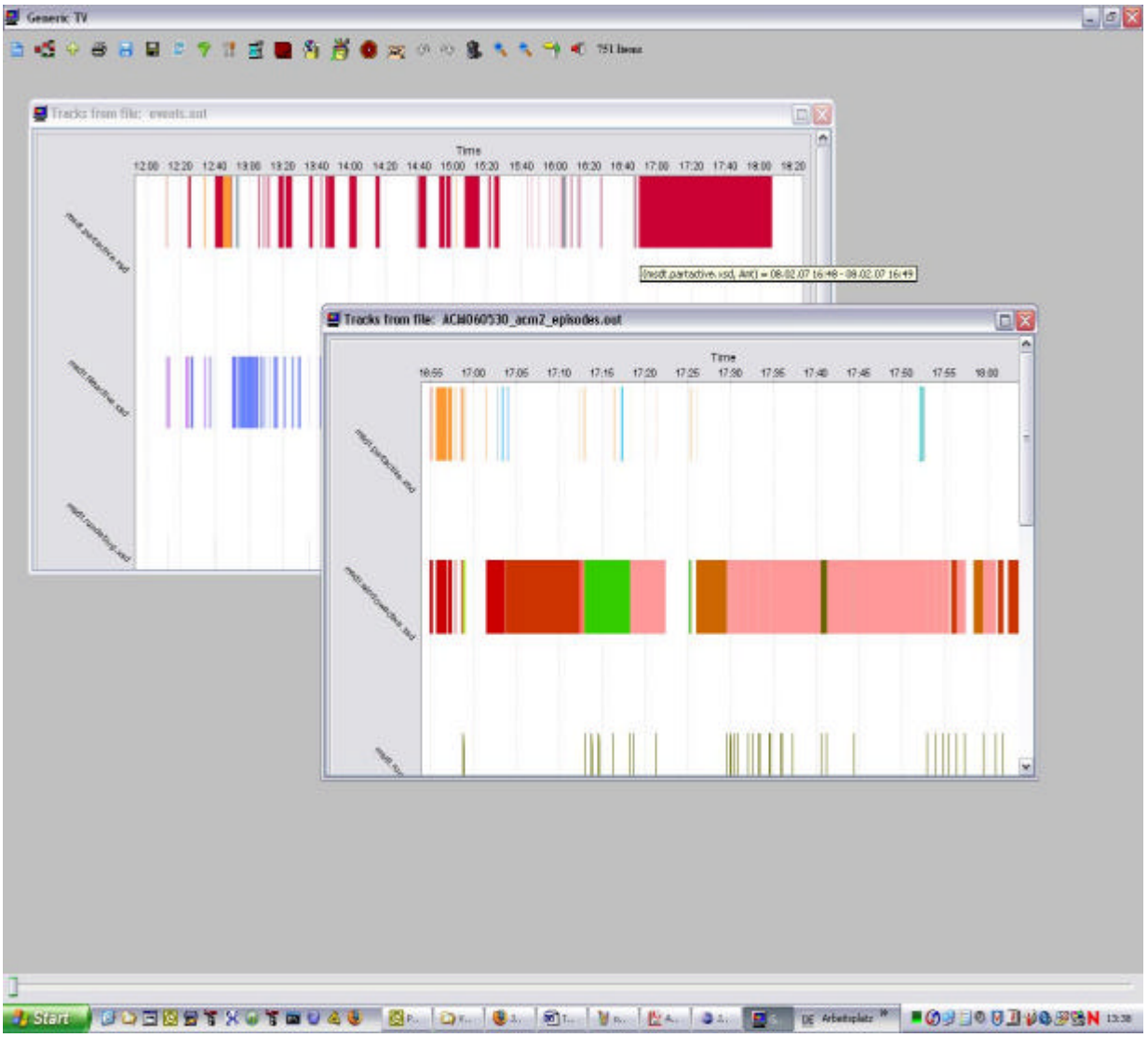


Abbildung 36 - Darstellung von Programmiertätigkeiten durch Generic TV durch die Anwendung von InternalFrame

In der Detailansicht bekommt der Benutzer durch die „Scrolling"-Technik einen Gesamtüberblick der Programmiertätigkeiten der Inputdatei. So wird die Technik „Detail und Übersicht" unterstützt.

[39] Zeitformat: HH:MM:SS

Letztendlich hat die Technik „Information Hiding“ einen wichtigen Beitrag gebracht. Alle Informationen der *Eigenschaft* gleichzeitig darzustellen, war meiner Meinung, nicht umsetzbar. Durch Benutzerinteraktion (wie zum Beispiel Mausklick) erzeugt Generic TV die Darstellung von weiteren Elementen dieser Menge.

V.1.2. Analyse und Umsetzung der Zeitdarstellungstechniken

Die Möglichkeit der Darstellung des Zeitbezugs wurde im Abschnitt IV.5.1 vorgestellt. Im Allgemeinen sind eine statische und eine dynamische Darstellung möglich. Die von Generic TV dargestellten Programmiertätigkeiten verändern sich nicht über die Zeit. Eine Programmiertätigkeit, die z.B. um 15:15:02 bis 15:16:10 stattgefunden wird sich nie wieder ändern. Die Anfangs- und Endzeiten der Programmiertätigkeiten sind unveränderbare Eigenschaften. Aus diesen Inputdaten werden statischen Daten in Generic TV. Es ist keine „Dynamik“ im Visualisierungsprozess erforderlich. Aus diesem Grund ist eine statische Darstellung der Daten bestens geeignet. Die Literatur empfiehlt i.d.R. den Zeitstrahl als beste statische Darstellungsmöglichkeit des Zeitbezugs. Die Hauptanforderung an Generic TV war es, die Spuren der Programmiertätigkeiten darzustellen. Der Zeitbezug der Spurelemente war nur durch einen Zeitstrahl deutlich wiederzugeben. Die Auswahl der Darstellungstechniken für Generic TV war keine leichte Aufgabe. Die Kalenderdarstellung[40] war am besten geeignet. Sie war am unaufwendigsten einzusetzen. Diese Technik hat aber Vor- und Nachteile:

- Ein Vorteil ist, dass durch die Kalendertechnik der Zeitbezug intuitiv vermittelt. Die Reihenfolge der Programmiertätigkeiten wird dem Benutzer schnell vermittelt. Der Anwender hat eine zeitliche Lokalisierung[41]. Dadurch ist eine explorative Analyse von Häufungen in einer Datenmenge leichter zu realisieren.

- Ein großer Nachteil dieser Darstellungstechnik ist die schnelle Ausschöpfung der Anzeigefläche. Dies ist ein schwerwiegendes Problem. Bei der Darstellung von sehr vielen Programmiertätigkeiten in einer Spur besteht die Gefahr von Informationsverlusten. Die Größe der Spurenanzeige müsste so skaliert werden, dass der Verlust von vielen Information nicht verhindert wird.

- Die in dieser Arbeit vorgestellten Darstellungstechniken „Detail und Übersicht“ und „Scrolling“ sind eine große Hilfe, um Nachteile der Kalendertechnik zu minimieren. Desweiteren trägt die Verfügbarkeit eines Filtertools[42] in Generic TV zur Verringerung der Nachteile der Kalendertechnik bei. Es ermöglicht eine Datenreduktion in der Darstellung.

V.1.3. Analyse und Umsetzung der Realisierung einer quantitativen Darstellung

Unter quantitative Darstellung versteht man eine Repräsentation von Datenwerten einer Datenmenge durch grafische Objekte. Die Realisierung einer quantitativen Darstellung, d.h. einer Visualisierung der Datenwerte der Inputdaten, ist eine wichtige Aufgabe von Generic TV. Die Visualisierung von Datenwerten vereinfacht die Analyse der Daten in dem Sinne, dass wichtige Eigenschaften der Inputdaten (nämlich ihre Datenwerte) dem Benutzer

[40] im Abschnitt IV.5.1 vorgestellt
[41] Er weißt genau, wann etwas stattgefunden hat.
[42] Das Filtertool und seine Funktionalität werden im Abschnitt V.2 vorgestellt

intuitiv vermittelt werden. Unterschiedliche Visualisierungstechniken der Datenwerte wurden im Abschnitt IV.5.2 vorgestellt. Die Datenwerte werden durch acht definierte Variablen abgebildet:

- Größe,
- Helligkeit,
- Farbe,
- Richtung (Orientierung),
- Form des Elements,
- Musterung (Textur),
- Position auf der Ebene.

Generic TV stellt die Programmiertätigkeiten als geometrische Figuren[43] dar. Die Datenwerte, die im Rahmen dieses Projekts visualisiert werden, müssen sind numerische und Booleschen Werte[44].Die quantitative Darstellung in Generic TV erfolgt durch Variation von Größe, Farbe und Helligkeit der Datenwerte.

Die Anwendung der Farbe in Generic TV hat hauptssächlich eine selektive und ordinale Funktionalität:

- Die Programmiertätigkeiten sind durch ihre Farben unterschieden (z.B. „Kodieren“, „Hilfe suchen“, „Telefonat durchführen“).
- Die Gruppierung durch bestimmte Eigenschaften oder die Zugehörigkeit zu einer bestimmten Klasse wird Farben betont[45].

Durch die selektive und ordinale Funktionalität der Farbe in Generic TV ist eine Abbildung der Datenwerte durch Farbe und die Farbhelligkeit nur mit Zurückhaltung anzuwenden. Denn die abzubildenden Datenwerte unterscheiden sich:

- Numerische Werte können unendlich viele Ausprägungen haben.
- Boolesche Werte können stattdessen nur zwei Ausprägungen besitzen: True oder False

Aus diesem Grund würden Boolesche Werte durch Farbeinsatz in Generic TV nicht sehr beeinflusst werden[46].In Generic TV wird die Abbildung von Booleschen Werten durch eine „farbliche Markierung“ in der Darstellung der Programmiertätigkeit (d.h. in der geometrischen Figur) ohne dabei die ursprünglichen Farben der Tätigkeit zu überdecken

[43] Zurzeit werden drei Figuren unterstützt: Recheck, Ellipse, und abgerundetes Rechteck (RoundRectangle).
[44] Eine Erweiterung der zuvisualisierende Datenwerten ist natürlich nicht auszuschließen
[45] Siehe am Beispiel in der Abbildung 30
[46] Der Grund dafür liegt in der nur zwei-Ausprägungen-Eigenschaft von Boolschen Werten.

(Siehe Abbildung 37). Die ursprünglich selektive und ordinale Funktionalität der Farbwiedergabe geht nicht verloren.

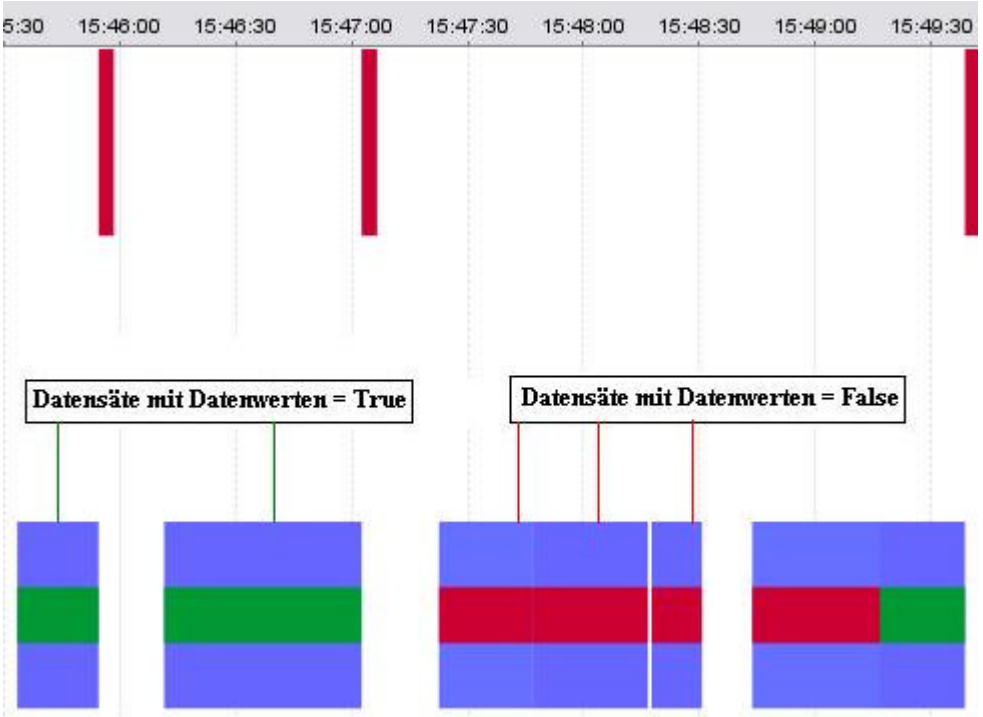


Abbildung 37 - Quantitative Visualisierung von Booleschen Werten in Generic TV

Die Abbildung von numerischen Werten durch Größenunterschiede von geometrischen Figuren ist sinnvoll. Durch die unterschiedlichen Größen der geometrischen Figuren kann der Anwender sofort die Datenwerte wahrnehmen und unterscheiden. Generic TV stellt mehrere übereinanderliegende Spuren in einem Informationsraum (IR) dar. Eine Dimension des IR, nämlich die X-Achse stellt die Zeit dar. Die Breite jeder einzelnen Spur ergibt sich durch die Formel:

Breite des IR / Spuranzahl.

In der Renderingphase der Visualisierungspipeline werden die geometrischen Figuren zur Darstellung von Programmiertätigkeiten auf dem Bildschirm gemalt. Dafür sind vier Parameter notwendig (Siehe Abbildung 38):

1. Die X-Position auf die Ebene, die von der Anfangzeit der darzustellenden Programmiertätigkeiten abhängig ist.

2. Die Y-Position auf die Ebene, die von der Spurgehörigkeit der darzustellenden Programmiertätigkeiten abhängig ist. Die Y-Koordinate entspricht dem Anfangspunkt der Spur im IR. Ihre Breite bestimmt den Endpunkt.

3. Die Länge, die von dem Endzeitpunkt der darzustellenden Programmiertätigkeiten abhängig ist.

4. Die Höhe, die von der Spurbreite und ggf. von der Darstellung eines numerischen-Werts abhängig ist.

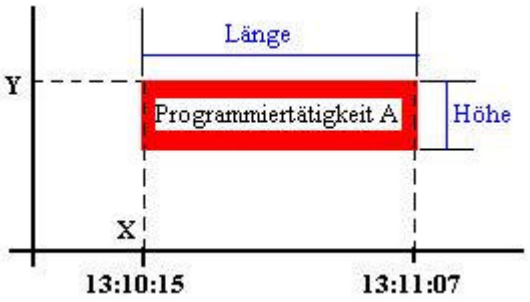


Abbildung 38 - Darstellung einer Programmiertätigkeit in der Ebene

Die X-, Y-Koordinaten und Länge ergeben sich durch die Eigenschaften der Daten (Anfangs- und Endzeitpunkt). Die Berechnung der Höhe ist kompliziert. Die Höhenabbildung numerischer Werte durch geometrische Figuren wird durch die Spurbreite und den darzustellenden Wert bedingt. Aus diesem Grund findet in der Renderingphase die Berechnung eines sogenannten Höhemultiplikators statt. Dieser berechnet in Abhängigkeit von der usprüngliche Breite der Spur und dem numerischen Wert eine darzustellende Höhe. Dabei werden folgende Parameter gebraucht:

- X = numerischen Wert, der darzustellen ist
- M = maximale Ausprägung des numerischen Wertes in den Daten
- E = Prozentuelle Eingabe des Benutzers für die Höhe der kleinsten Ausprägung des numerischen Wertes (also 0)
- B = usprüngliche Breite der Spur
- H = neuberechnete Höhe der Spur

Die neuberechnete Höhe einer geometrischen Figur ergibt sich dann aus der Anwendung der folgenden Formel:

$$H = B * \left(\left(1 + \frac{X}{M} \right) * \left(\frac{E}{100} \right) \right)$$

Die Abbildung 39 zeigt die Umsetzung einer quantitativen Visualisierung von numerischen Werten durch Generic TV.

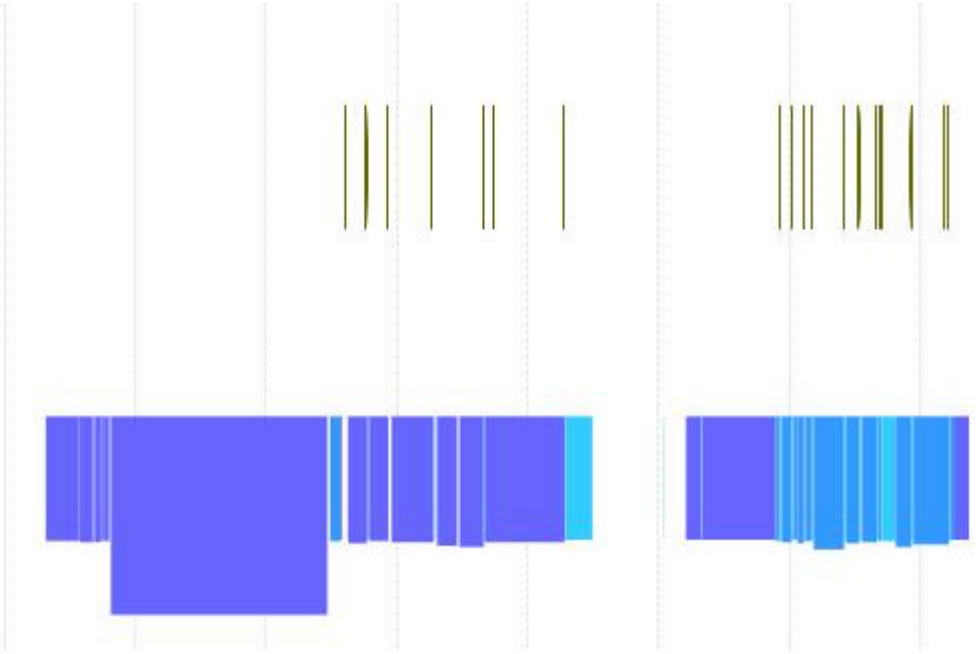


Abbildung 39 - Quantitative Visualisierung von numerischen Werten in Generic TV

V.2. Arbeitsablauf mit Generic TV (Produktsübersicht)

In diesem Abschnitt wird Generic TV als fertiges Produkt vorgestellt. Das Frage, was man mit Generic TV machen kann, soll jetzt beantwortet werden. Außerdem werden die „wichtigsten“ Aktivitätsdiagramme und Use-Case-Diagramme vorgestellt. Sie enstanden während der Definitions- und Entwurfsphase des Entwicklungsprozesses.

Definition des Begriffs Konstellation:

Unter Konstellation wird in dieser Arbeit der aktuelle Zustand einer Darstellung definiert. Gemeint sind:

- die aktuell im System geladenen Inputdateien,
- die aktuell im System dargestellten Spuren,
- die aktuell im System sichtbaren Programmiertätigkeiten,
- die aktuellen Farben der dargestellten Programmiertätigkeiten im System,
- die aktuellen Einstellungen im System für die quantitative Visualisierung etc.

Die benötigten Informationen, die zu der aktuellen Konstellation geführt haben, können als gtv-Dateien gespeichert bzw. aus gtv-Dateien geladen werden. Beim Laden einer Konstellation werden die Information bzgl. der Farben, vorhandenen Spuren und Filterkriterien auf die Darstellung angewendet.

Datei einlesen

Die Abbildung 40 zeigt ein Aktivitätsdiagramm für das Einlesen und die Darstellung einer Inputdatei durch das System. Nachdem das System gestartet wurde, kann der Benutzer durch ein Dateiauswahlmenü eine Datei aus den Inputdaten auswählen.

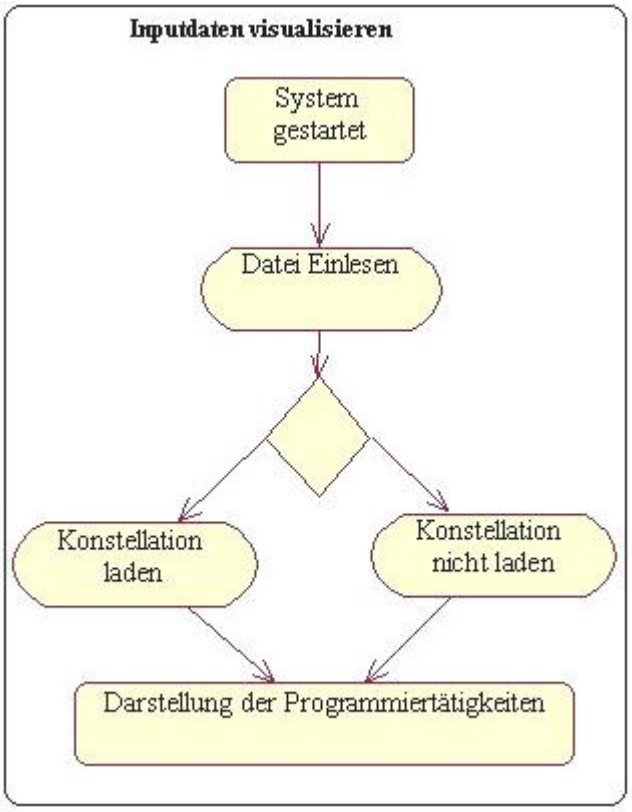


Abbildung 40 - Aktivitätsdiagramm der Aktivität „Inputdatei visualisieren“

Das System führt zuerst die Filterungphase der Visualisierungspipeline durch. Dies findet in der Abstraktionsschicht DataReader[47] statt. Danach wird der Benutzer gefragt, ob eine Konstellation geladen werden soll. Nach der Beantwortung der Frage wird die Mappingsphase, durch die Abstraktionsschicht DataLoader des Systems realisiert.Die dritte Phase, Rendering der Pipeline, ermöglicht dann dem System die eingelesenen Programmiertätigkeiten als Spuren grafisch darzustellen. Nachdem die Spuren dargestellt wurden, sind dem Benutzer mehrere Interaktionsmöglichkeit vom System zur Verfügung gestellt. Die Use-Case-Diagramme in Abbildungen 41 bis 43 geben einen kleinen Überblick der Möglichkeiten, die dem Benutzer zur Verfügung gestellt sind um die Darstellung durch Interaktionen zu beeinflussen. Durch diese angebotenen Interaktionen kann der Benutzer seine Sichten auf die Inputdaten variieren. Dadurch ist evtl. eine bessere Analyse der Daten möglich.

[47] Siehe Abschnitt V.3.1.1

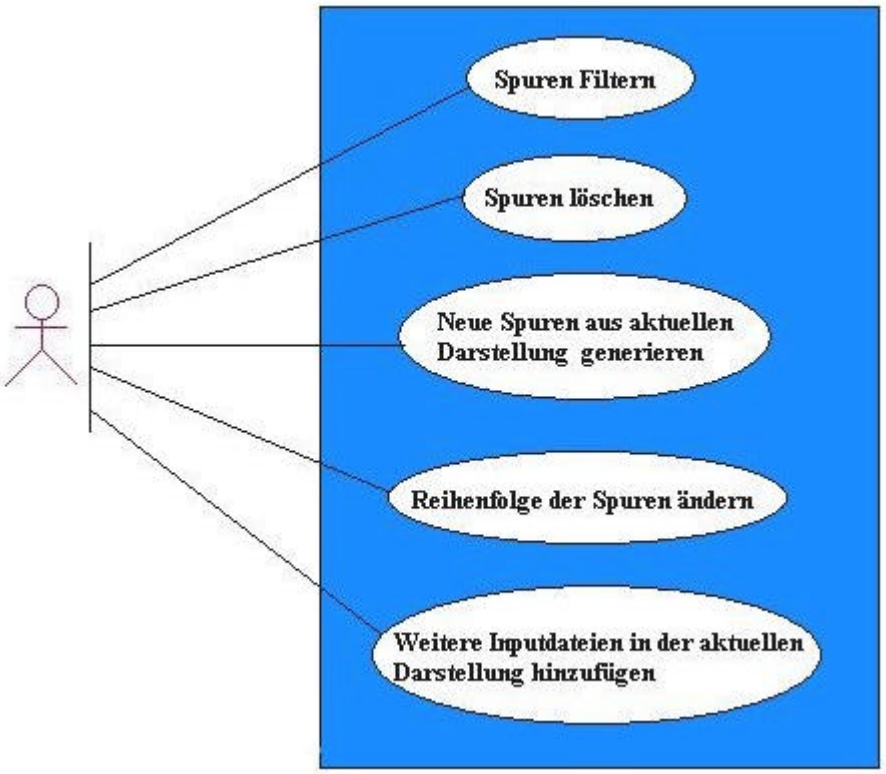


Abbildung 41 - Use-Case zur Beeinflussung der Spuren

Der Benutzer kann Spuren durch von ihm definierten Kriterien filtern (Abbildung 44 zeigt ein Aktivitätsdiagramm der Aktivität Spur Filtern). Er hat die Möglichkeit gesamte Spuren aus der Darstellung zu löschen oder die Darstellung mit neuen geladenen Inputdaten zu ergänzen. Über ein Farbetool kann der Benutzer die Farbe der dargestellten Programmiertätigkeiten dynamisch ändern. Er hat die Möglichkeit eine Quantitative Visualisierung der dargestellten Daten vom System realisieren zu lassen. Die Visualisierung von numerischen und Booleschen Datenwerten werden jeweils durch Größe- und Farbeabbildung realisiert (Siehe Abschnitt IV.5.2). Die Abbildung 45 zeigt das Aktivitätsdiagramm der Aktivität „Quantitative Visualisierung generieren“.

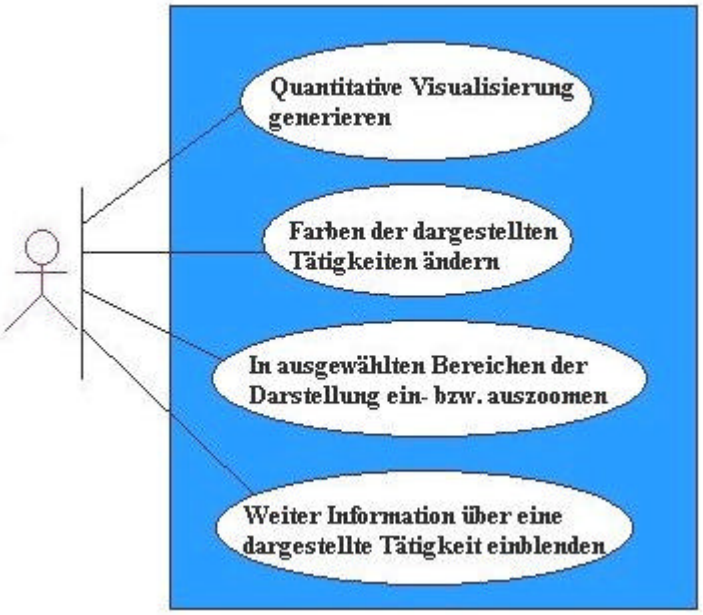


Abbildung 42 - Use-Case zur Beeinflussung des Layouts der Darstellung

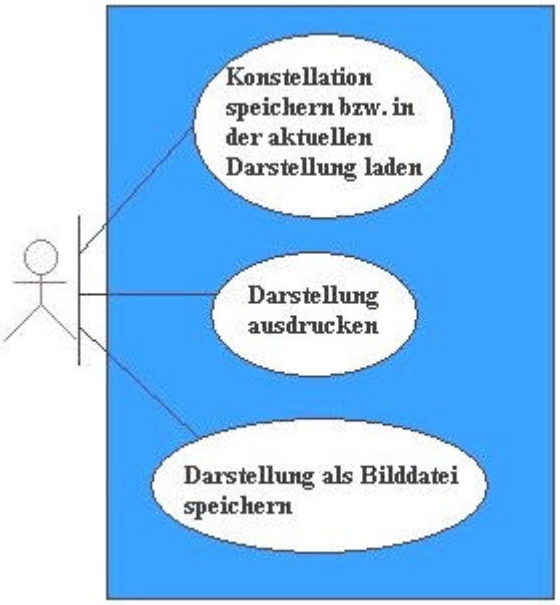


Abbildung 43 - Use-Case zu Extraktionsmöglichkeiten

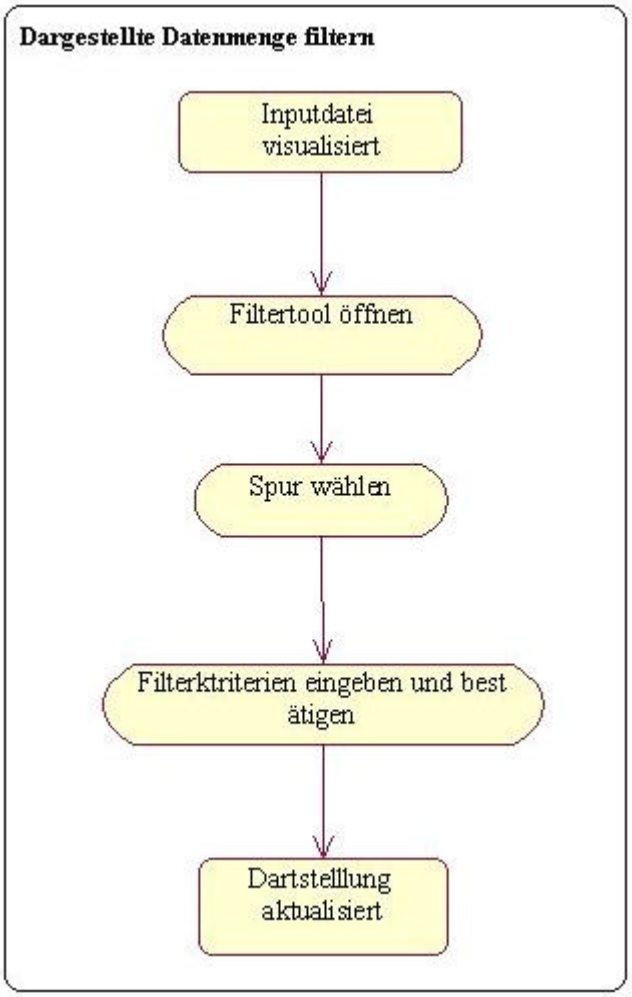


Abbildung 44- Aktivitätsdiagramm der Aktivität „Spur filtern“

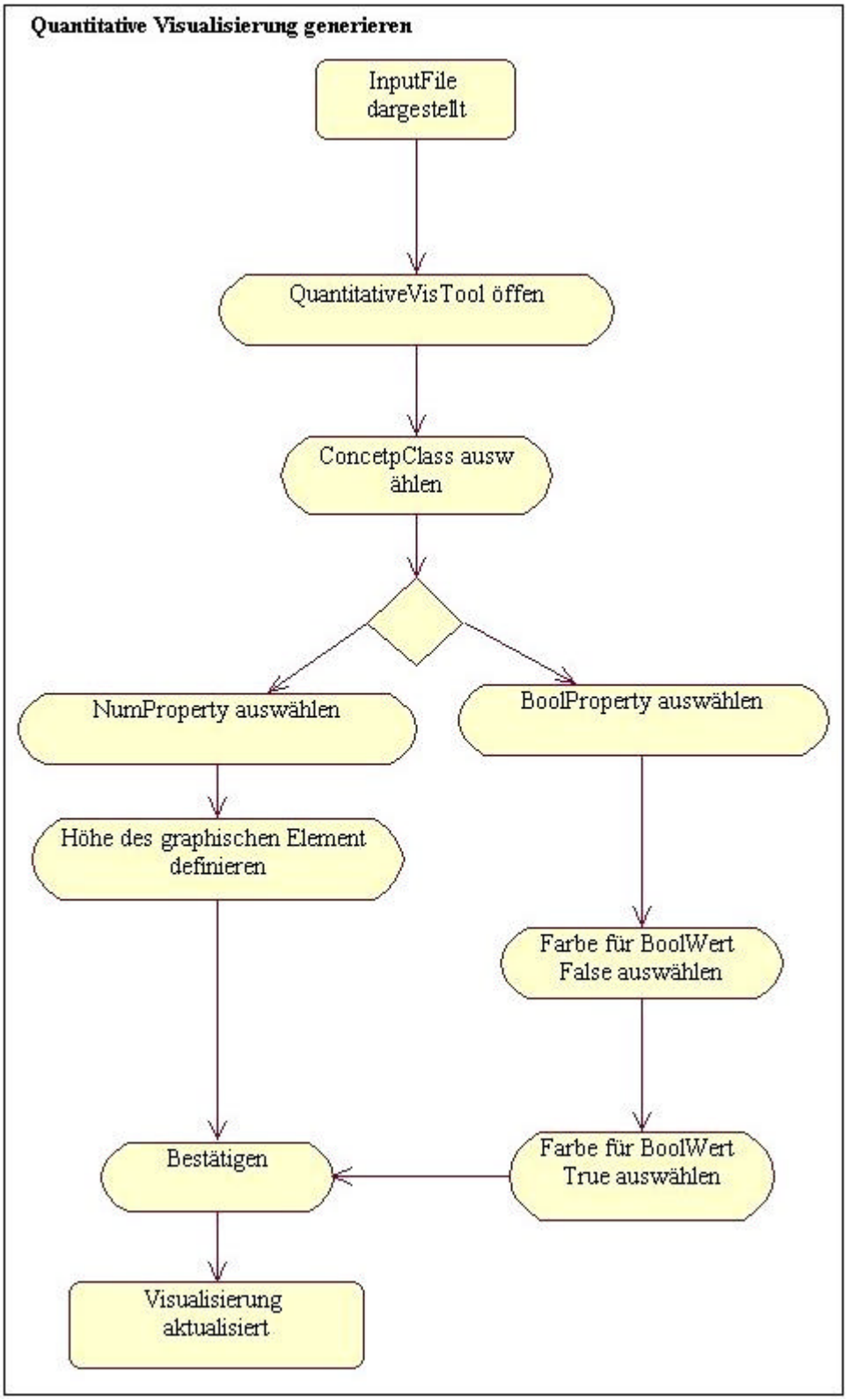


Abbildung 45 - Aktivitätsdiagramm der Aktivität „Quantitative Visualisierung generieren“

Die Benutzung von Use-Case- und Aktivitätsdiagrammen war sehr hilfsreich, um die Definition- und Entwurfsphase von Generic TV durchzuführen. Sie haben dazu beigetragen, die Anforderungen an Generic TV besser zu verstehen. Sie offerierten Lösungsansätze für Anforderungen an die Software. In dieser Phase haben sich vier Abstraktionsschichten ergeben, die im letzten Abschnitt dieses Kapitels vorgestellt werden.

V.3. Konzeption

In der Analysephase sind die Anforderungen an das System auf Konsistenz, Eindeutigkeit und Vollständigkeit geprüft worden. In der nächsten Stufe wird die Software entworfen. Es ist eine entscheidende Entwicklungsphase. H. Balzert bezeichnet diese Etappe als das „Programmieren im Großen“ [BH98]. Die Entwicklung von Generic TV ist die Umsetzung von den vorgestellten und entwickelten Konzepten zur Visualisierung von Programmiertätigkeiten.

V.3.1. Einflussfaktoren

Hauptsächlich Forscher sind Anwender dieser Software. Das System ist entwickelt worden, um zeitabhängige Daten zu visualisieren. Deshalb Einsatzmöglichkeiten sind außerhalb der Forschung gut denkbar.

Das System ist für im „single-user“-Konzept entwickelt. Deshalb können Zugriffsrechte oder parallele Datenzugriffe außeracht gelassen werden.

Entwicklungsumgebung

Die Portabilität der Programmiersprache Java war der Hauptgrund für die Wahl dieser Sprache. Die Entwicklung von Generic TV mit der Programmiersprache Java der Version 1.5 erfolgt. Die Stabilität der Software, insbesondere der Einsarz des Frameworks JfreeChart, ist mit früheren Javaversionen ist nicht ausreichend getestet worden. Ihre Anwendung ist also erst ab Version 1.5 empfohlen. Die eingesetzte Entwicklungsumgebung war Eclipse 3.2. Zur Erleichterung der Entwicklung der Benutzeroberfläche wurde das Plugin Visual Editor (VE) verwendet.

Ausgangsdaten und Framework

Die Inputdaten wurden in Abschnitt III.2 dieser Arbeit vorgestellt. Sie bestehen aus flachen Dateien. Die Lokalisierung der Daten bei Anwendung der Software ist nicht relevant. Der Benutzer wählt die darzustellenden Dateien durch ein Dateiauswahlmenü selbst. Deshalb können sie sich auf einer Anwendungsplattform oder auch auf einem Server befinden.
Der Einsatz des Frameworks JFreeChart wurde in Abschnitt III.3 begründet.

V.3.2. Abstraktionsschichten

Den Entwurf von Generic TV kann man in vier Abstraktionsschichten des Systems zusammenfassen (Siehe Abbildung 46). Dabei übernehmen bestimmte Schichten eindeutige Arbeitsphasen der Visualisierungspipeline. Der DataReader realisiert die Filterungsphase. Im DataLoader findet die Mappingsphase statt. Der DataViewer ist für die Renderingphase der Visualisierungspipeline verantwortlich.

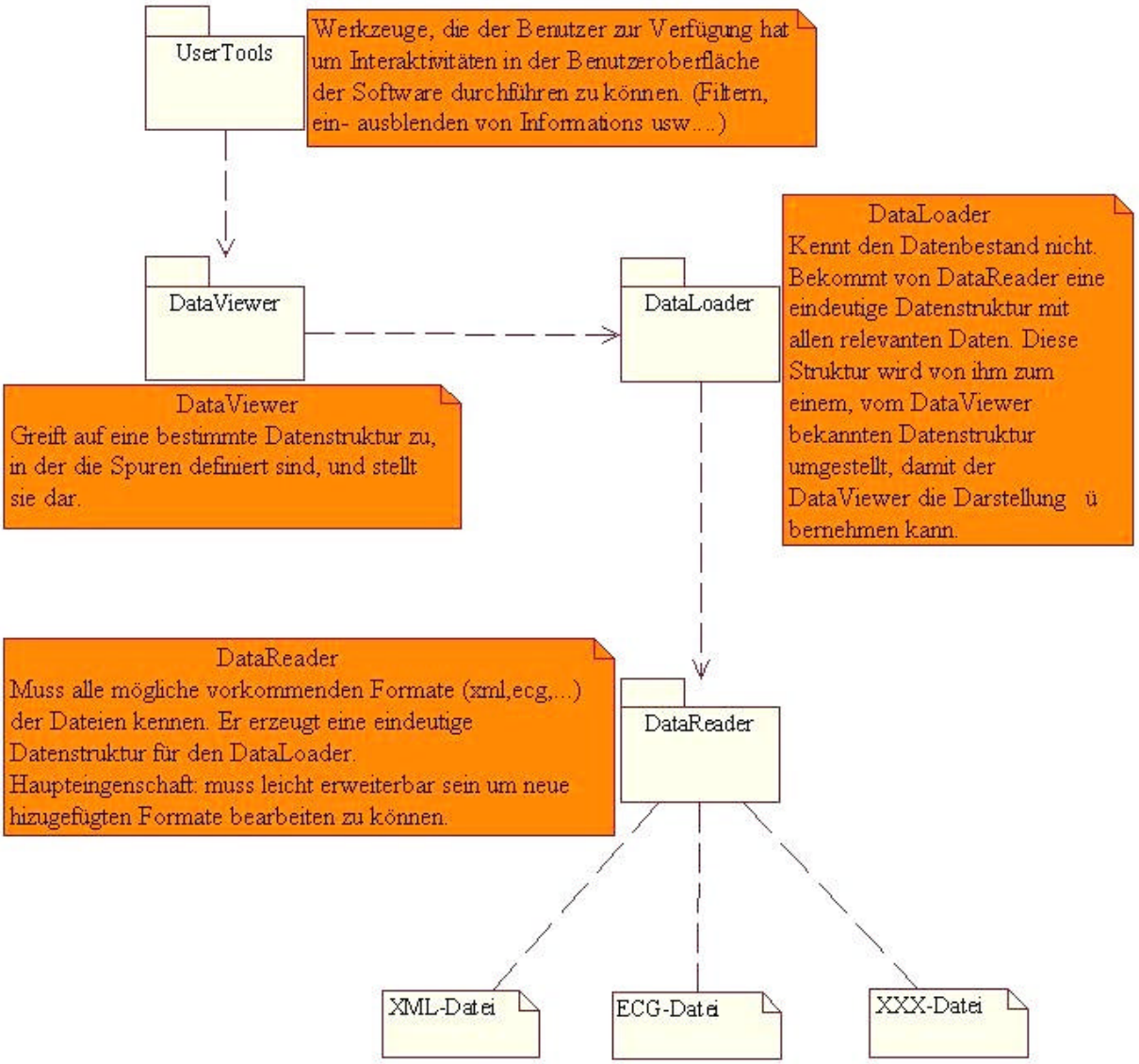


Abbildung 46 - Abstraktionsschichten von Generic TV

V.3.2.1. Abstraktionsschicht 1: DataReader

Der Zugriff auf die Inputdaten erfolgt mit dem DataReader. Die Dateiformate aller darstellbaren Inputdaten müssen ihm bekannt sein. Die Aufgabe des DataReaders ist es, die vom Benutzer zur Darstellung gewählten Dateien zu lesen. Daraus wird eine eindeutige Datenstruktur, die in diesem Abschnitt **DS1** genannt wird, zu erzeugt. Die Datenstruktur **DS1** enthält alle benötigten Datensätze[48]. Sie wird der nächsten Abstraktionsschicht (DataLoader) als QuotationList-Objekt übergeben. Der DataReader arbeitet mit drei Klassen zusammen:

1. Eine Klasse, die das Interface Quotation implementiert

[48] also Informationsobjekte und ihre Attribute

2. Eine Klasse, die die abstrakte Klasse GenTVDataGenerator[49] des DataLoaders erweitert.

3. Eine Klasse, die die abstrakte Klasse GenTVDataStore des DataLoaders erweitert.

Nach dem Einlesen der Inputdaten enthält ein GenTVDataStore-Objekt die gelesenen Datensätze. In dem Beispiel von ECG- und Atlas.ti- Inputdaten werden die GenTVDataStore-Objekte als Vector von Array implementiert. Der Einsatz des DataReaders gewährleistet einen Teil der Generizität der Software. Der DataReader entspricht der Umsetzung der ersten Phase der Visualisierungspipeline, nämlich der Filterungphase. Die Visualisierung von zeitabhängigen Daten durch Generic TV benötigt die Instanzierung von Annotation-Objekten. Diese erzeugen wiederum GenTVTask-Objekte. Siehe dazu Abbildung 47. Diese Instanzierung von Annotation kann nur erfolgen durch die Erzeugung von einem Concept-Objekts, das zu einer ConceptClass-Objekt zugeordnet ist. Zur Instanzierung der drei Objekte (Annotation, ConceptClass und Concept) sind vier bestimmte Datensätze der Inputdaten erforderlich:

1. Der Name der Konzeptklasse,

2. die Anfangszeit des Konzepts,

3. die Endzeit des Konzepts,

4. der Name des Konzepts.

Angewendet auf dem aktuellen Format der Inputdaten (Siehe Abschnitt III.2) ergab eine Rücksprache mit Stephan Salinger und Sebastian Jekutsch folgenden Abbildungen zwischen Datensätzen der Inputdaten und Elementen des im Abschnitt III.2 eingeführten Metamodells:

Aktueller Datensätzen in der Inputdaten	Element des Metamodells
type	ConceptClass.Name
Timestamp	Quotation.Start
EndTime	Quotation.End
Resourcename	Concept.Name
Eigenschaft 1	Property 1
Eigenschaft n	Property n

Tabelle 9 - Abbildung der aktuellen Datensätzen auf Metamodellelemente

Durch drei bestimmte Klassenoperationen auf die Datensätze, werden folgende Operationen nach dem Einlesen der Inputdatei, durchgeführt:

1. Zur Vervollständigung der eingelesen Datensätze überprüft der DataReader, ob alle Datensätze für die Generierung von Annotation-Objekten (Concept, Anfangs- und Endzeitpunkt des Konzepts und die Konzeptklasse) vorhanden sind. Ein konkretes Beispiel wäre:

[49] Die Klassen GenTVDataGenerator und GenTVDataStore werden im Abschnitt über den DataLoader ausführlich vorgestellt.

msdt.run.xsd 30.05.2006 16:58:49 run.

Bei diesem Datensatz werden nur der Name der Konzeptklasse, der Anfangszeitpunkt und eine Property (nämlich „**run**“) geliefert. Es fehlen also der Endzeitpunkt und der Namen des Konzepts. Sie sind aber für die Instanzierung von Annotationen erforderlich. Der DataReader führt an dieser Stelle eine Vervollständigung der Daten durch. Ein „Defaultname“ und eine „Defaultendzeit“ werden generiert und zugewiesen. Zur Defaultendzeit Anfangszeit plus eine Sekunde addiert. Das Konzept erhält den Namen „NoNameAssigned“ Annotation.

2. Eine Glättung bzw. Fehlerkorrektur der Datensätze wird ebenfalls durch den DataReader durchgeführt. Zum Beispiel werden die Anfangs- und Endzeitpunkte auf „korrekte“ Formatierung überprüft. Jede erweitende Klasse der GenTVDataGenerator-Klasse muss ein Datumsformat implementieren und ihn in einen vom System bekannten Datumsformat umzuwandeln. Das System kann nur dieses Datumsformat behandeln:

   JJ.MM.JJJJ HH:MM:SS.

   Die Anfangs- und Endzeitpunkte werden auf diese Formatierung überprüft und ggf. korrigiert.

3. Die Bestimmung charakteristischer Eigenschaften der Datensätze wird durch den DataReader realisiert in dem die abstrakte Methode *createQuotationList(GenTVDataStore)* der abstrakten Klassen GenTVDataGenerator des DataLoaders implementiert wird. Die Implementierung dieser Methode realisiert, die in der Tabelle 5 definierte Abbildung zwischen Datensätzen und Metamodellelemente. Mit dieser dritten Operation werden die eingelesene Datensätze eindeutig klassifiziert. Dabei werden die Concept-, ConceptClass und ggf. Property- und Value-Objekte durch eine konzeptuelle Analyse der gelieferten „Kopfzeile“ der Inputdatei erzeugt. Zum Beispiel kann eine Kopfzeile der Inputdatei folgendermaßen formatiert werden:

   type(tab[50])timestamp(tab)endtime(tab)resourcename(tab)duration(tab)changed.

   In dieser Kopfzeile werden die vier „Muss-“Datensätzen ermittel. Daraus werden die ConceptClass- und Concept- Objekte (mit Anfangs- und Endzeitpunkten) erzeugt. Die restlichen Datensätze der Kopfzeile (falls vorhanden) definieren die Properties bzw. Eigenschaften des Concept-Objekts. Eine eingelesener Datensatz zum Beispiel könnte folgendermaßen lauten:

   msdt.fileactive.xsd(tab)30.05.2006 17:02:52(tab)30.05.2006 17:03:23(tab)A.java(tab)31000(tab)2.

   Daraus erzeugt die GentVDataGenerator-Klasse folgende Objekte:

   a. Ein Concept-Objekt namens **A.java** mit dem Anfangszeitpunkt **30.05.2006 17:02:52** und dem Endzeitpunkt **30.05.2006 17:03:23**.

[50] Tabulator

b. Dem ConceptClass-Objekt **msdt.fileactive.xsd** wird ein Concept-Objekt zugewiesen.

c. Das Concept-Objekt besitzt zwei Property-Objekte: **Duration** und **Changed**. Deren zwei Ausprägungen werden durch die zwei Value- Objekten gleichen Namens und die numerischen Werte **31000** und **2** dargestellt.

Der DataReader ist ein Garant der Generizität von Generic TV. Er realisiert die Implementierung und Erweiterung von wichtigen Interfaces bzw. abstrakten Klassen des DataLoaders. Der Entwurf und seine Implementierung war eine sehr kritische Phase der Entwicklung von Generic TV.

V.3.2.2. Abstraktionsschicht 2: DataLoader

Die Mappingsphase der Visualisierungspipeline ist eine Daten-zu-Geometrie-Abbildung. Der Einsatz von Entwurfmustern[51], wie der Schablonen- und der Fabrikmuster, war sehr hilfsreich für den Entwurf des DataLoaders. Je nach dem, ob der eingelesenen Datensatz einen numerischen, String- oder Booleschen Wert enthält, werden Value-Objekte erzeugt. Ein Beispiel wäre die Erweiterung eines Klassenwertes, wie NumericValue oder BoolValue. Dadurch wird ihr Typ eindeutig spezifiziert. Ein weiteres Beispiel ist die Implementierung der abstrakten Klasse GenTVDataGenerator. In dieser abstrakten Klasse ist ein Konstruktor implementiert. Dieser führt den Aufruf der abstrakten Methode *createQuotationList(GenTVDataStore)* durch. Diese Methode wird von der jeweiligen Erweiterung der Klasse GenTVDataGenerator (z.B. GenTVECGDataGenerator) implementiert. Auf Grundlage der Formate der Inputdateien werden eindeutigen Quotation-Objekte erzeugt.

Das Datenmapping

Das Datenmapping in Generic TV erfolgt durch den Einsatz von drei Akteuren:

1. Die Systemkonfigurationsdatei **GenTV.ini**[52] enthält systemrelevante Informationen. Der Benutzer kann durch das Editieren dieser Datei, das System nach seinen Wünschen konfigurieren. Die für die Mappingsphase relevanten Einträge der GenTV.ini sind:

   - Definition der Datentypen von den Property: Durch Editieren der GenTV.ini muss Benutzer die Datentypen aller vorkommenden Property dem System bekannt machen. Dabei muss immer die Property und der dazugehörige Datentyp eingetragen werden. Wie zum Beispiel:

     Property A = Number, Property B = Boolean, Property C = String.

     Die hierbei definierten Datentypen werden durch Erweiterung der abstrakten Klasse *Value* (z.B. NumericValue oder BoolValue) implementiert.

   - Definition des Aussehens der darzustellende Datensätze: Der Benutzer oder (in seltenen Fällen) das System[53] können durch Einträge in der GenTV.ini das

[51] Siehe eine Zusammenfassung des Klassendiagramms des DataLoaders in der Abbildung 47.
[52] Die Bedienungseinleitung des Systems enthält eine genauere Beschreibung der Datei
[53] durch die Verwendung von Farb- und Formtools

Aussehen der Informationsobjekte konfigurieren. Hierfür müssen für alle Informationsobjekte folgende Zeile in der GenTV.ini-Datei eingefügt werden:

Konzeptnamen|Konzeptklasse={IntRed|IntGreen|IntBlue};Füllung;Aform.

Wie zum Beispiel:

Welcome|msdt.partactive.xsd={25|45|133};Empty;RoundRectangle2D

Formal beschrieben bedeutet diese Zeile: „Alle Konzepte (**Welcome**), der Konzeptklasse (**msdt.partactive.xsd**), werden mit den Integerwerten **25, 45, 133** der java.awt Farbe[54] dargestellt. Dabei sollen sie mit der geometrischen Form **RoundRectangle2D** repräsentiert werden. DieRoundRectangle2D muss leer (**Empty**) sein".

2. Die Klasse **GenTVAppManager** ist die Schnittstelle bzw. das Fassade-Objekt eines Fassaden-Muster zwischen dem DataLoader und dem DataViewer. In dem Fall ist der DataViewer der „Client" des Fassaden-Musterkonzepts. Der GenTVAppManager verwaltet alle relevanten Informationen der GenTV.ini für die Darstellung der Inputdaten.

3. Die Klasse **GenTVTask**[55] bildet eine der Schnittstellen zum Framework JfreeChart durch Erweiterung der JfreeChart-Klasse Task. Die Attribute (wie Form, Farbe und Füllung) der GenTVTask-Objekte werden durch Einträge in die GenTV.ini gesetzt. Der DataViewer übernimmt die Darstellung auf dem Bildschirm. Er gibt die Attribute aus, welche in den GenTVTask-Objekten gesetzt wurden.

Die Generizität

Die Abbildung 47 zeigt ein zusammengefasstes Klassendiagramm des DataLoaders mit seinen Schnittstellen zum DataReader und zum DataViewer. Es ist eine Erweiterung des in Abschnitt III.2 vorgestellten Metamodells. Die Generizität der Generic TV wird durch zwei Klassen und ein Interface garantiert:

1. Die Arbeitsweise und Implementierung der Klasse **GenTVDataStore** wurde im Abschnitt über den DataReader erläutet. Die GenTVDataStore-Objekte müssen in ihren Attributen die vier erforderlichen Datensätze zur Erzeugnung von Annotations-Objekten enhalten. Im dem Fall ECG- und Atlas.ti- Inputdaten sind sie als java.util.Vector von Array implementiert worden. Dabei wird jeder Eintrag der Kopfzeile der Inputdatei in einem Feld des Arrays gespeichert. Z.B. Beispiel für eine Kopfzeile mit folgenden Einträgen:

   type(tab)timestamp(tab)endtime(tab)resourcename(tab)duration(tab)changed

   Zu diesem Beispiel wird ein Array der Länge 6 erzeugt. Das erste Feld enthält en Eintrag „**type**", das zweite Feld „**timestamp**" (Anfangszeitpunkt) und so weiter. Das erste Array-Objekt des Vectors GenTVDataStore enthält die „Kopfzeile" selbst. Die weiteren Arrays enthalten Ausprägungen der Kopfzeile. Das bedeutet zum Beispiel, dass Arrays folgendermaßen initialisiert worden sind:

[54] Instanzzierung einer java.awt.Color(25,45,133)- Objekt
[55] Siehe Abbildung 47

String [] ls_einArray = {msdt.FileActive.xsd, 30.05.2006 17:02:52, 30.05.2006 17:03:58, A.java, 45000, 2}.

Für die Kopfzeileeinträge, die keine Ausprägung besitzen übernimmt die Filterungphase, also der DataReader, die Eintragung des Werts „null“ bzw. eines Defaultwerts im Arrayfeld.

2. **Das Interface Quotation** durch die Implementierung des Interfaces werden, die Concept-, Focus- und Value- Objekten erzeugen. Die Quotation- Objekte werden in der Klasse GenTVDataGenerator erzeugt.

3. Die Klasse **GenTVDataGenerator** erzeugt aus den Informationen des GenTVDataStores (Konzeptklasse, Konzept, Anfangs- und Endzeitpunkte, Property etc.) Quotation-Objekte. Diese werden in einer Instanz der Klasse QuotationList verwaltet. Aus dem QuotationList-Objekt werden durch eine Instanz der Klasse GenTVAnnotationGenerator aller Annotation-Objekte und dadurch alle GenTVTask-Objekte erzeugt. Die GenTVTask- Objekte stellen die Informationsobjekte[56] einer Visualisierung dar. Die durch Generic TV dargestellten Programmiertätigkeiten sind Instanz der Klasse GenTVTask.

[56] im Abschnitt V.1 beschrieben

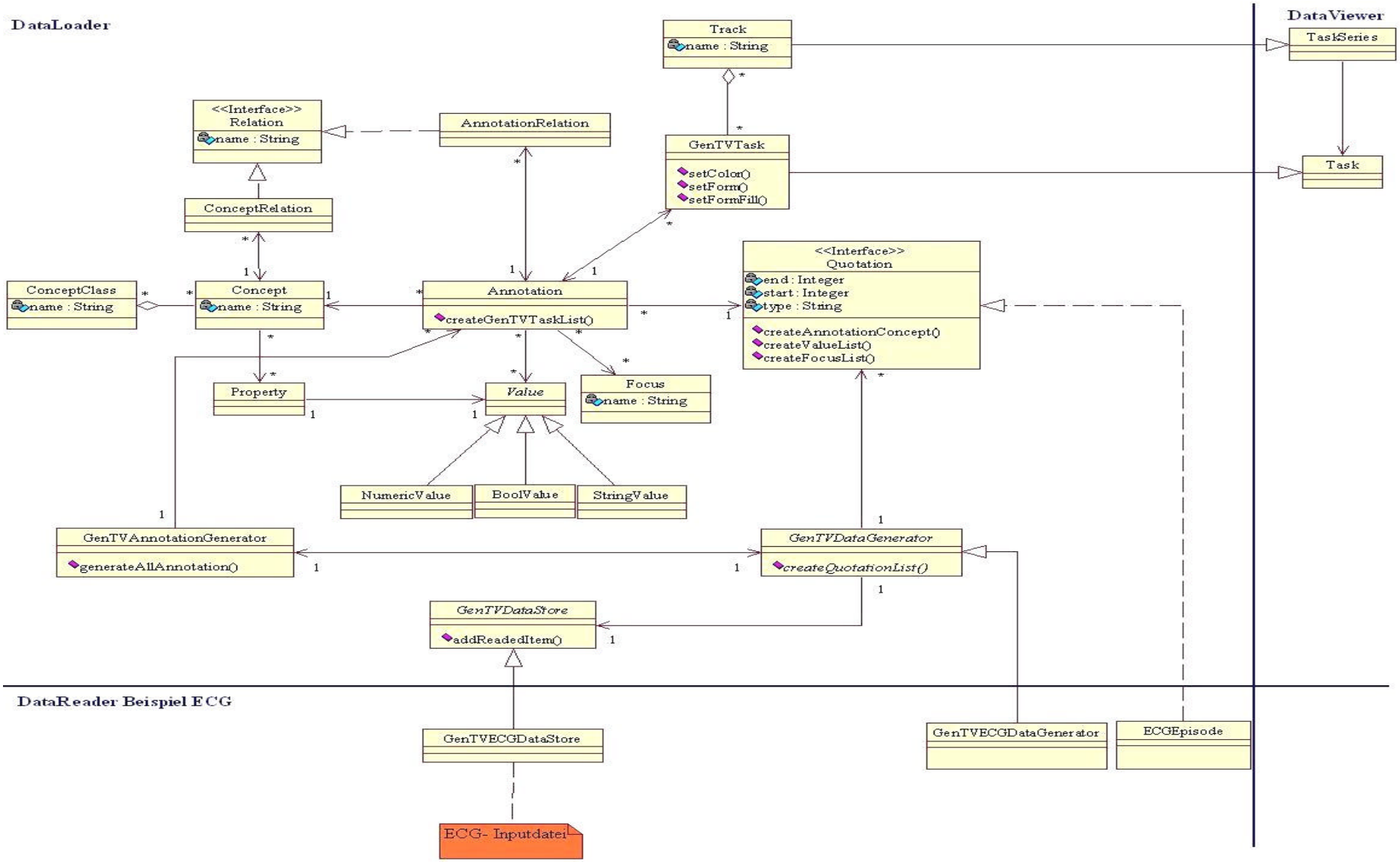


Abbildung 47 - Zusammenfassung des Klassendiagramms des DataLoaders mit Schnittstellen zum DataReader und DataViewer

V.3.2.3. Abstraktionsschicht 3: DataViewer

Der DataViewer ist die Umsetzung der Renderingsphase der Visualisierungspipeline in Generic TV. Er stellt die gefilterten und „gemappten“ Inputdaten grafisch dar. Bei dieser Darstllung rüstet er sich auf dem Framework JFreeChart. Der DataViewer hat keinen direkten Zugriff auf die Inputdaten. Sie sind schon durch den DataReader und den DataLoader so strukturiert worden, dass der DataViewer nur eine eindeutige Datenstruktur bekommt. Die hier darzustellende Datenstruktur sind Instanzen der Klasse Track. Die Klassen Track und GenTVTask stellen weitere Schnittstellen zum Framework JFreeChart, in dem sie jeweils seine Klasse TaskSeries und Task erweitern[57]. Im Abschnitt V.1.1 wurde der Entschluss, die Programmiertätigkeiten in InternalFrames darzustellen, begründet. Jedes InternalFrame wird im Hauptfenster der Anwendung dargestellt. Nur innerhalb des Hauptfensters kann es bewegt werden. Die InternalFrames sind Instanzen der Klasse W_MasterInternalFrame. Das Hauptanwendungsfenster ist eine Instanz der Klasse W_MasterFrame. Diese zwei Klassen gehören zu den wichtigsten des DataViewers. Die Renderingphase findet in der Klasse GenTVGanttRenderer statt. Sie ist eine Erweiterung der Klasse GanttRenderer von JFreeChart. Wie der Name ahnen lässt[58], stellt der GenTVGanttRenderer Objekte, welche man als „Arbeiter“ oder „Handwerker“ der Darstellung bezeichnen könnte. Die „gemappte“ Information (Form, Farbe und Füllung) der GenTVTask werden in der Klasse GenTVGanttRenderer abgelesen und für die Zeichnung der Inforamtionsbobjekte angewendet. In dieser Klasse werden unter anderem die Koordinaten der darzustellenden geometrischen Figuren, die Dimensionen (Länge, Höhe), die Anzahl der Spuren oder auch ihre Breite berechnet.

Bei der Entwicklung von Generic TV tratten unterschiedliche Probleme auf. Die Erweiterungen bestimmeter Klassen von JFeeChart haben bei dem Lösen dieser Problemen geholfen. Ohne diese Erweiterungen wären z.B. folgende Anforderungen nicht erfüllt:

- Unterschiedliche Farbe innerhalb einer Spur darstellen: Die Grundfunktionalität von JFreeChart sieht die Anwendung von unterschiedlichen Farben für Task-Objekte nicht vor. Sie musste deshalb überarbeitet werden.

- Unterschiedliche geometrische Figuren innerhalb einer Spur darstellen: Die einzigen dargestellten Figuren waren Rechtecke.

- Realisierung einer quantitativen Visualisierung der Inputdaten.

- Keine Visualisierungsmöglichkeit der überlappenden Tasks.

- Keine Unterstützung der drill-down-Mechanismen des Visualisierungsprozesses

- Keine Interaktionen zwischen Anwender und Darstellung etc.

Die Erweiterung der Klasse GanttRenderer in der Klasse GenTVGanttRenderer war einen notwendiger Schritt, um die Anforderungen an Generic TV zu erfüllen. Dadurch sind entscheidende Punkte der Visualisierung zeitabhängiger abstrakter Daten in einem Gantt-Diagramm nicht „besser“, sondern „anders“ implementiert worden.

[57] Siehe die Erläuterungen im Abschnitt III.3.
[58] Die deutsche Übersetzung des englischen Worts Renderer ist Leistungsbringer.

V.3.2.4. Abstraktionsschicht 4: UserTools

Der DataViewer zeichnetdie Datensätze in die **BOF** der Software. Danach kann der zahlreiche Aktivitäten in der BOF von Generic TV durchführen. Diese „Aktivitäten“ werden als Benutzerinteraktionen bezeichnet. Durch die Benutzerinteraktionen sind unterschiedliche Ansichten der Inputdaten zu möglich. Mittels eines Dialogsystems, kann der Anwender Daten manipulieren sowie Ansichten selbst definieren. Eine bessere Analyse der Daten ist so möglich. Die Erhöhung der Analysefähigkeit ist ein wesentlicher Pluspunkt für Generic TV.

Die Realisierung von Benutzerinteraktionen wird in Generic TV durch die Abstraktionsschicht UserTools ermöglicht. Sie ist die Benutzer Sschnittstelle. UserTools stellt alle Dialogsysteme für die interaktive Nutzung von Generic TV zur Verfügung. Dazu gehört unter anderem die Umsetzung der Drill-down-Mechanismen des Visualisierungsprozesses. Unter drill-down versteht man, dass der Benutzer auf Daten (Roh- oder aufbereitete Daten) durch Bildinteraktion zugreift. Mit Generic TV sind keine Veränderungen der Inputdaten möglich. Gemeint ist damit, dass Generic TV keine Editierfunktionalität für Inputdaten unterstützt. Stattdessen stellt UserTools zahlreiche Werkzeuge zur Datenmanipulation in Generic TV zur Verfügung. Zu den wichtigsten Werkzeugen zählen:

- Das FilterTool ermöglicht, Datensätze benutzerdefiniert gefiltert darzustellen.
- Das TracerTool[59] gestattet dem Benutzer, neue Spuren zu generieren. Durch die definierten Kriterien können neue Spuren hinzugefügt sowie dargestellte Datensätze kopiert werden.
- Mit dem ColorTool kann der Anwender die Farbe der dargestellten Informationsobjekte setzen.
- Das SortTool befähigt den Benutzer, die Reihenfolgen der Spuren in der Darstellung festzusetzen.
- Durch das QuantitativVisTool können für quantitative Visualisierungen notwendige Parameter (Farbe der Booleschen Werte und Höhe der Numerischen Werte)[60]) eingegeben werden.

Diese Werkzeuge sind durch die sogenannten Dialogtechniken entwickelt worden. Sie stellen die Funktionalität der Benutzungsschnittstelle von Generic TV dar. Eine Dialogtechnik ermöglicht den Austausch zwischen Computer und Benutzer. Sie hat folgende Merkmale:

1. Unterschiedliche Eingabegeräte (Mause, Tastatur etc.) müssen verwendet werden.
2. Die Eingabedaten sind grafischen Daten und/oder beziehen sich auf die grafische Ausgabe.

Die „International Organisation for Standardization“ ISO hat Normen und Standards für die Gestaltung von **BOF** entwickelt. Die wichtigste Norm ist die DIN ISO 9241 [ISO_I]. Sie bestimmt die ergonomischen Anforderungen für Bürotätigkeiten an Bildschirmarbeitsplätzen. In ihnen werden z.B. Anforderungen an die Farbdarstellung, die Informationsdarstellung und

[59] analog dem FilterTool
[60] Siehe Abschnitt V.1.3

die Benutzerführung gestellt, also unter anderem an die Grundsätze für die Dialoggestaltung[SH06]. Diesen Leitfaden muss jede moderne Software erfüllen. Es folgen Beispiele für die Umsetzung einiger Normen durch Generic TV:

- **Aufgabenangemessenheit**: Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen. Zum Beispiel sollte ein Dialog dem Benutzer nur Informationen anzeigen, die zur Erledigung seine Arbeit braucht.

  In Generic TV setzt der Anwender das QuantitativVisTool ein. Damit können Inputdaten quantitativ visualisiert werden. Der Benutzer ist in der Lage, die Farbeabbildung der Datenwerte und die Höhe der geometrischen Figuren für die Größeabbildung der Datenwerte selbst zu definieren.

  Die Abbildung 48 zeigt unterschiedliche Sichten des Werkzeugs. Die selektierte Konzeptklasse (Punkt A in oberem Bild der Abbildung 48) zeigt **nur** Property an. (Punkt B), die in Konzepten der selektierten Konzeptklasse vorkommen. Dadurch ergibt sich, dass der Benutzer die Konzeptklasse „msdt.partactive.xsd“ im oberen Bild gewählt hat. Das System hat dazu nur „duration“ als Property aufgelistet. Im unteren Bild war die Wahl des Benutzers „msdt.fileactive.xsd“ als Konzeptklasse. Das System hat „duration“ und „changed“ als Property in B aufgelistet. Das passiert, weil die Konzeptklasse „msdt.fileactive.xsd“ zwei Eigenschaften bzw. Properties besitzt. Das System garantiert dadurch, dass nur „brauchbare“[61] angezeigt werden. Also nicht alle Properties, unabhängig von der selektierten Konzeptklasse, auflistet werden. Je nach der selektierten Property zeigt das QuantitativVisTool dem Benutzer, welche Abbildungen der Datenwerte auf visuelle Attribute möglich sind. Im oberen Bild hat der Benutzer die Property „duration“ gewählt. Der Datentyp dieser Property ist „Number“. Bei der Abbildung von numerischen Werten wird die Höhe der geometrischen Figuren eingestellt. Durch das QuantitativVisTool wird über ein Benutzermenü die Höhe der geometrischen Figuren geregelt. Im unteren Bild ist die Property „Changed“ gewählt worden. Es ist ein Boolescher Datentyp. Das Menü von Generic TV ermöglicht dem Benutzer die Farben für „True-“ und „False-“ Boolesche Werte auszuwählen. (Die Abbildung von Booleschen Werten erfolgt durch Farbe).

[61] notwendige

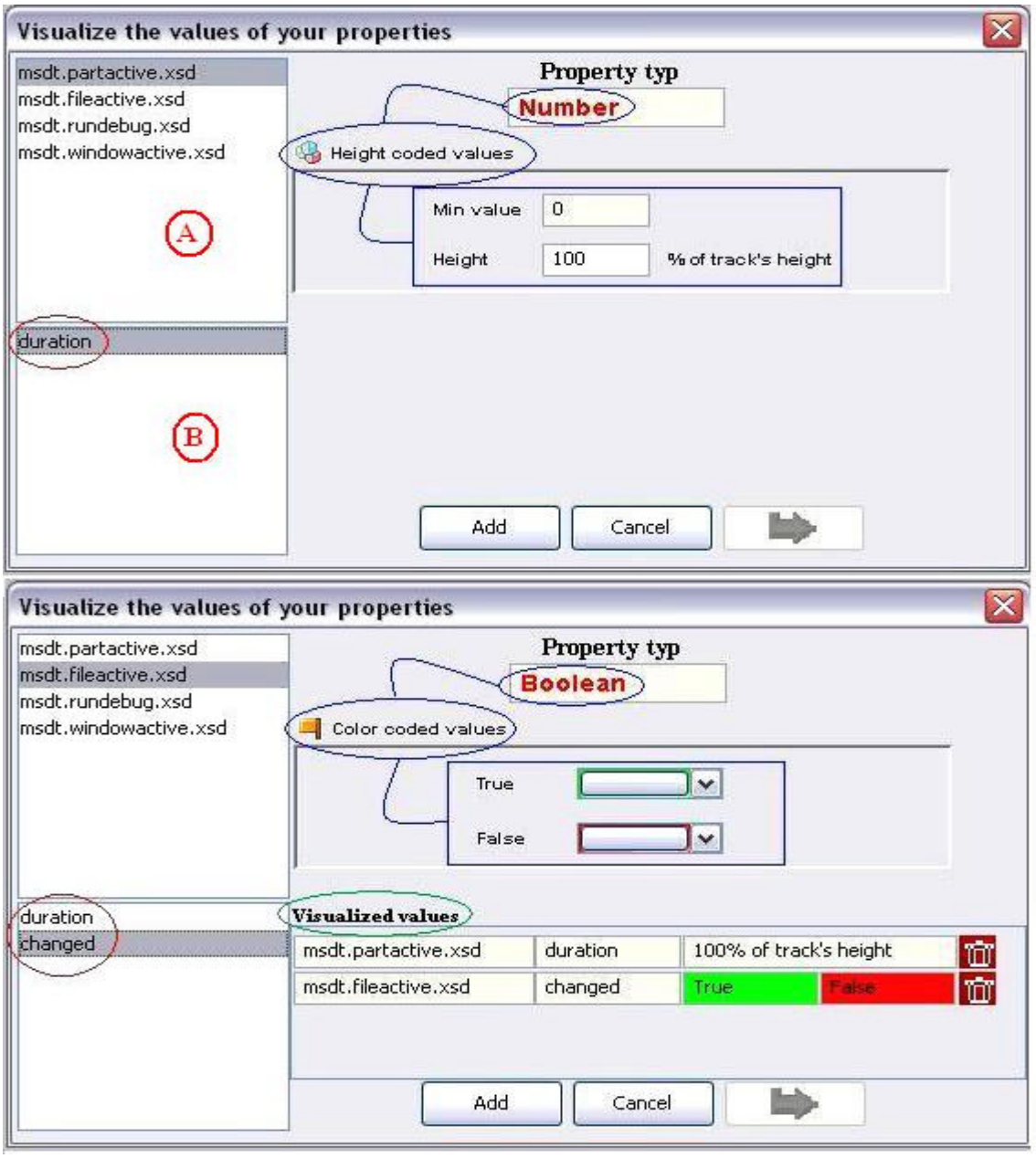


Abbildung 48 - Beispiel der Umsetzung der Aufgabeangemessenheit in Generic TV

- **Selbsterklärungsfähigkeit**: Ein Dialog ist selbstbeschreibend, wenn jeder einzelne Dialogschritt verständlich ist bzw. auf Anfrage erklärt wird.

  Im Beispiel von Generic TV kann der Benutzer mit dem SortTool die Reihenfolge der angezeigten Spuren definieren. Beim Öffnen des Tools wird die aktuelle Reihenfolge angezeigt (Punkt A). Mit Hilfe des „drag and drop"-Prinzips können neue Spurreihenfolgen entstehen (Punkt B)[62].

[62] Siehe Abbildung 49

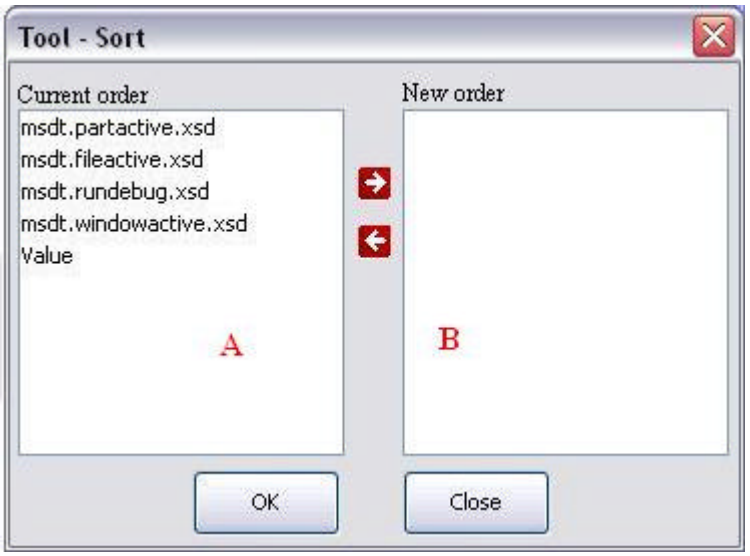


Abbildung 49 - Beispiel der Umsetzung der Selbsterklärungsfähigkeit in Generic TV

- **Fehlertoleranz**: Ein Dialog ist fehlertolerant, wenn das Arbeitsergebnis, trotz fehlerhafter Eingaben, mit keinem oder minimalem Korrekturaufwand erreicht werden kann. Es gibt nach fehlerhaften Eingaben keinen undefinierten Systemzustand. Das FilterTool von Generic TV kann als Beispiel angeführt werden. Es gibt dem Benutzer die Möglichkeit, die angezeigten Inputdaten mit definierten Kriterien zu Filtern. Dabei werden alle Datensätze (GenTVTasks), die die Filterkriterien **nicht** erfüllen, aus der Darstellung entfernt. Die Eingabe der Filterkriterien besteht aus 4 Schritten:

  1. Spur auswählen,
  2. Property auswählen,
  3. Operator auswählen,
  4. Value bzw. Ausprägung für die Property auswählen.

  Nachdem die Spur ausgewählt worden ist, folgt die weitere Auswahl nach dem Prinzip der Aufgabenangemessenheit. Es werden in den nächsten Schritten nur Datensätzen angezeigt, die sich aus den vorherigen Selektionen ergeben.

  <u>Beispiel</u>:

  Spurauswahl ➔ nur vorhandene Property aufgelistet ➔Propertyauswahl ➔ nur mögliche Operator and Ausprägungen aufgelistet

  Das obere Bild der Abbildung 50 zeigt ein durch das Verfahren entstandenes Filterkriterium (Puntk A). Dieses Filterkriterium entspricht einem Benutzerbefehl an Generic TV. Formal lautet es: *„Entferne aus der Spur msdt.partactive.xsd alle dargestellten Informationsobjekte, dessen Namen Welcome sind.“* Bevor die Filterkriterien angesetzt werden, führt Generic TV eine syntaktische Überprüfung der Kriterien durch. Bei einer fehlerhaften Eingabe eines Filterkriteriums generiert Generic TV eine Fehlermeldung. Der Benutzer wird gewarnt, dass sein Befehl nicht durchgeführt werden kann. Im unteren Teil der Abbildung 50 (Punkt B) ist das

Filterkriterium nicht vollständig eingegeben worden. Der Name fehlt. Dadurch ist das Filtern der Daten nicht möglich. Generic TV informiert den Benutzer, dass sein Filterkriterium die syntaktische Überprüfung **nicht** bestanden hat. Dass sie dadurch ungültig ist.

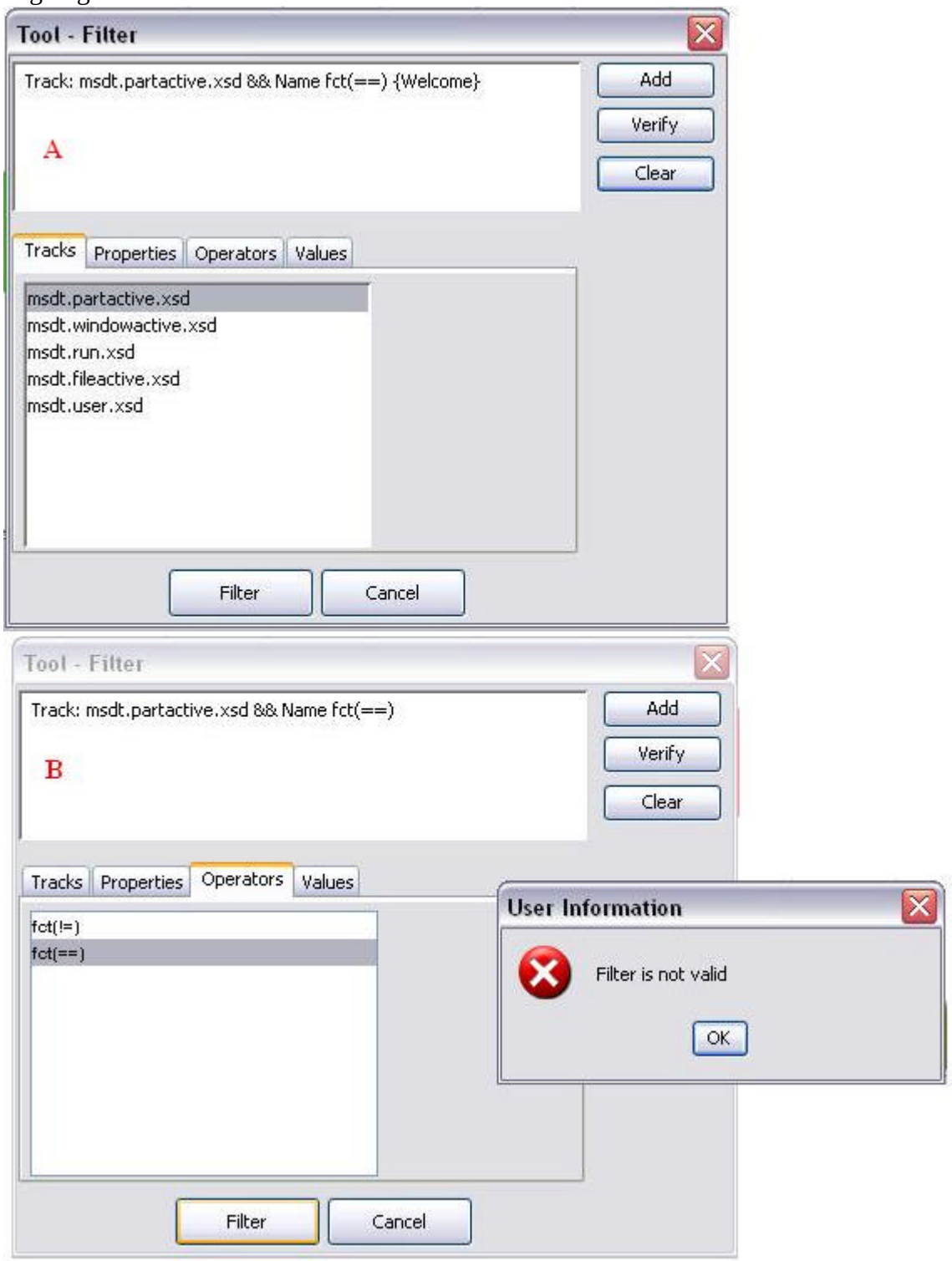


Abbildung 50 - Beispiel der Umsetzung der Fehlertoleranz in Generic TV

- **Steuerbarkeit**: Ein Dialog ist steuerbar, wenn der Benutzer Richtung und Geschwindigkeit des Dialogablaufs beeinflussen kann.

In Generic TV sind die Filter- und TracerTools auf eine enge Zusammenarbeit mit dem Benutzer angewiesen. Dass trifft besonders für die Auswahl der Spuren, der Property und ähnlichem zu. Die Steuerbarkeit des Dialogs gewährleistet die Javax.swing-Komponente **JTabbedPane**. Die JtabbedPane-Komponente ist definiert als eine Komponente, die verschiedene Komponenten auf Reitern anordnen. Die Geschwindigkeits- und Richtungskontrolle der Erzeugung von Filterkriterien wird so dem Benutzer überlassen.

Alle Dialoge von Generic TV halten sich streng an die sogenannte Vier-W-Regel [SH06]. Diese Regel hilft dem Benutzer den Überblick bei seiner Arbeit nicht zu verlieren. Die vier Fragen dieser Regel lauten:

- **W**o bin ich?

  Alle Tools von Generic TV besitzen einen Namen. Sobald das Tool geöffnet wird, wird dieser angezeigt

- **W**ie kam ich hierher?

  Alle Tools von Generic TV sind als modaler Dialog implementiert worden. Das heißt, dass sobald sie geöffnet sind[63], kann der Benutzer sich nur innerhalb des Dialogs bewegen. Dadurch weiß der Benutzer immer, dass das letzte Iconklick ihn „hierher“ brachte.

- **W**as kann ich hier tun?

  Alle Tools von Generic TV besitzen einen Hilfebutton. Die angezeigte Hilfe ist stets dialogbezogen.

- **W**ohin kann ich von hier aus?

  Auf Grund der Charakteristik des modalen Dialogs, kann der Benutzer beim Verlassen des aktuellen Dialogs nur wieder im Hauptanwendungsfenster landen.

[63] durch Mauseklick auf einem Icon der Haupanwendungsfenster

VI. Fazit

In Generic TV ist ein Werkzeug entwickelt worden, das in der sehr komplexen Welt der Visualisierung eingesetzt werden kann. Mit dem Werkzeug kann man zeitabhängige Daten visualisieren. Gegebenenfalls kann die Visualisierung der Daten durch die Benutzeraktivitäten manipuliert werden.

Der Einsatz von Generic TV ist für die explorative Analyse von abstrakten Daten eine klare Unterstützung. Ein Bild sagt mehr als tausend Worte, lehrt der Volksmund. Aber kann ein Bild nicht auch mehr als tausend Worte lügen? Visualisierungswerkzeuge können heute bei der Beantwortung solcher Fragen helfen. Es sind aber explizite Anforderungen an das Werkzeug vonnöten. Die Schwerpunkte der Visualisierung stellen die eingesetzten Darstellungstechniken, die Wiedergabe des Zeitbezugs sowie die Umsetzung einer quantitativen Darstellung der Datensätze dar.

Unterschiedliche Sichten der Inputdaten sollten eine explorative Analyse unterstützen. Die Möglichkeit Generic-TV-Darstellungen zu manipulieren, machen diese unterschiedlichen Sichten der Inputdaten möglich. Gegenstand dieser Diplomarbeit war die Erstellung eines Werkzeugs, das die Visualisierung von Programmieraktivitäten „geeignet“ darstellt. Durch die generischen Abschnitte von Generic TV können nicht nur Programmiertätigkeiten „geeignet“ dargestellt, sondern alle zeitabhängigen Daten. Dafür müssen sie vorher eine Bedingung respektieren, nämlich eine vorgegebene Formatierung.

Die Herausforderung einer implementierungsorientierten Arbeit, in der Kundenanforderungen grafisch umgesetzt werden müssen, empfand ich als besonders reizvoll. Die Einarbeitung in das Einsatzsgebiet des Visualisierungstools machte mir den Beginn des Projekts noch interessanter.

Eine gute Einarbeitung in die zu untersuchenden Forschungsgebiete hat die Bestimmung der Anforderungen an die Software vereinfacht. Nur durch eine genaue Analyse der Anforderungen war es möglich, ein detailliertes Pflichtenheft zu erstellen. Das enge Kunde-Dienstleister-Verhältnis garantierte Spannung. Diese wirkte inspierend und produktiv während des gesamten Projekts. Wie es oft in der Praxis der Fall ist, waren manche Gespräche zwischen Auftraggeber und Auftragnehmer eine beeindruckende Sammlung von Missverständnissen.

Der Zeitdruck, von Erarbeitung der Kernkriterien des Pflichtenhefts bis zur Fertigstellung der Software, war enorm. Dieses Projekt war nur durch den Einsatz eines geeigneten Frameworks möglich. Die Suche danach stellte eine erwähnenswerte Schwierigkeit dar.

VI.1. Kritik: Probleme, nötigte Verbesserungen

Probleme

- **Framework**

Nach der Einarbeitung in die Forschungsgebiete, die Generic TV unterstützen sollte, war die Suche eines Frameworks die erste große Schwierigkeit des Projekts. Ohne eine genaue Problembeschreibung, ist die Suche von wiederverwendbaren Komponenten nur sehr schwer zu realisieren. Aus den gewonnenen Projekterfahrungen kann ich folgenden Leitfaden verfassen: Benenne klare Ziele![64] Erkunde mögliche Wege zum Erreichen des Ziels![65] Wähle die Mittel zur Umsetzung![66]

- **Analyse und Entwurf**

  Der Entwurf der Generic TV war keine leichte Aufgabe. Die Verwirklichung der generischen Schnittstelle hat mir die meisten Sorgen verursacht. Sie hatte einen softwaretechnisch hohen Anspruch. Bei der Erstellung des ersten Metamodells der generischen Schnittstelle war meine Betreuer Stephan Salinger eine große Hilfe. Dieses Metamodell stellte das Fundament für den weiteren Ausbau des Projekts dar. Das Ergebnis ist zufriedenstellend. Die Generizität der Software war garantiert. Alle abstrakten Daten, die einem bestimmten Format entsprechen, können durch Generic TV visualisiert werden. Falls neu formatierte Inputdaten visualisiert werden müssen sind nur Erweiterungen der Software für das Einlesen und die Fertigstellung der Daten notwendig. Die Erweiterung von nur drei Klassen der Software ist erforderlich. Die Entwürfe der Werkzeuge für die Benutzeraktivitäten haben auch gewisse Schwierigkeiten verursacht. Es waren nicht nur konzeptionelle[67], sondern auch grafische Herausforderungen. Denn im Blickpunkt stand allzeit die gute Handhabung für den Anwender.

- **Die Implementierung von** Generic TV

  Die Implementierung war aufgrund des Umfangs mühsam. Die Software besteht aus vier Packages[68]. Diese wiederum enthalten insgesamt 54 Klassen.

  Die Umsetzung mancher Visualisierungsprozesse, wie z.B. der drill-down-Mechanismus der Visualisierungspipeline, war durch den Einsatz von Entwurfmustern sowie den gesamten guten Entwurf der Software möglich.

  Das größte Problem der Implementierung war die Vermeidung der sogenannten „Überlappungen[69]“ von geometrischen Figuren innerhalb einer Spur. Die Idee eine Spur, bei der Überlappungen vorhanden sind, in „Subspuren“ zu teilen, stammt von Stephan Salinger. Dadurch können einzeln dargestellte Informationsobjekte nicht überlagernd, sondern untereinander dargestellt werden. Die Schwierigkeit liegt in der Ermittlung von Überlappungen vor der Renderingphase. Also bevor die geometrischen Figuren auf dem Bildschirm ausgegeben werden. Für jedes darzustellende Informationsobjekt stellten sich folgende Fragen:

  a. Kollidiert dieses Informationsobjekt IO mit einem anderem?
  b. Wenn, ja zu welcher Spur gehört es?

[64] „Was muss man wissen?“
[65] „Wie man es tun?“
[66] „Womit möchte man es tun?“
[67] Umsetzen der Funktionen, wie z.B. „Farbe setzen“ oder „Spuren löschen“.
[68] vier Abstraktionsschichten
[69] Das Problem wurde im Abschnitt V.I.1 eingeführt

c. Wie viele Kollisionen gibt es in dieser Spur?[70]
d. Berechnet DrawPosition von IO und allen anderen Elementen diese Spur?

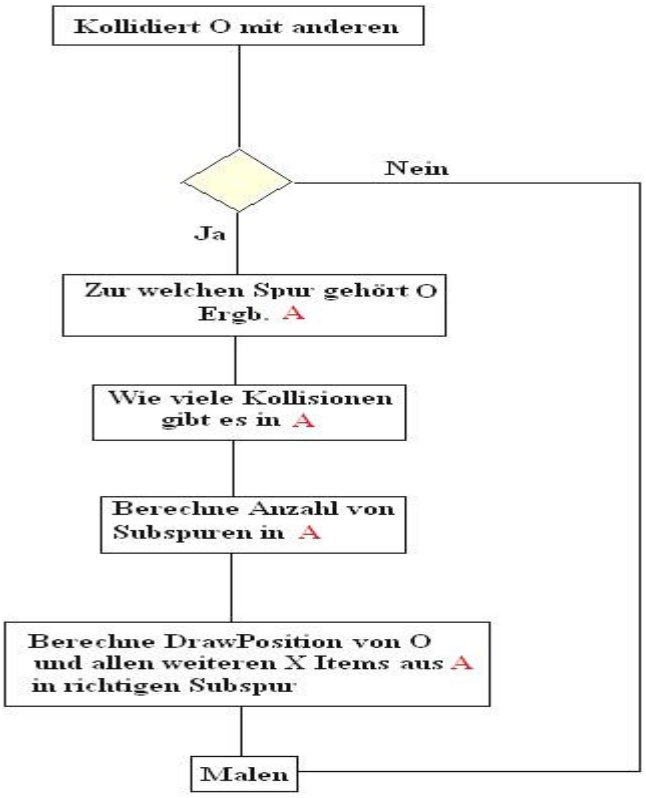


Abbildung 51- Idee zur Entwicklung des Algorithmen zur Lösung das Überlappungsproblem

Benötigte Verbesserungen

Eventuelle Erweiterungen bzw. Verbesserungen könnten Generic TV noch leistungsfähiger machen. Hier können diese Erweiterung in zwei unterschiedliche Klassen geordnet werden:

- **Defizite**

1. Die aktuelle Darstellung der Legende ist nicht zufriedenstellend. Die große Anzahl von Informationsobjekten in der Darstellung erzwingt eine große Anzahl von Legendenelementen. Informationsobjekte und Legendenelemente stehen i.d.R. in einem Spannugsverhältnis, welches daduch charakterisiert ist, dass Platzmangel für die Darstellung herrscht.

   Die momentane Lösung des Problems ist die Benutzung einer dynamischen Legende. Der Benutzer der Anwendung kann sie auf Wunsch ein- bzw. auszublenden. Eine zukunftige Lösung wäre z.B. die Legende in einem weiteren InternalFrame darzustellen. Problem dabei ist beim Ausdrücken der Darstellung. Der Druckprozess sollte eine evtl. dargestellte Legende suchen und mitdrucken. Ein weiterer derzeitiger Mangel ist, dass die Legende nicht tabellarisch dargestellt wird. Dass führt zu einem schlechteren Layout der Darstellung.

[70] Daraus ergibt sich Anzahl der Subspuren.

2. Die aktuelle Funktionalität des Filtertools besitzt einen zu verbessernden Schwachpunkt. Die dargestellten Datensätze können zwar aus der Darstellung entfernt werden, aber diese Aktionen sind nicht wieder rückgängig zu machen. Es besteht zurzeit keine Möglichkeit, gefilterte Datensätze wieder in die Darstellung einzufügen. Dass das Filtern einen Datenverlust verursacht ist hiermit erkannt. Eine Lösung des Problems wird als dringend eingestuft.

- **Weiterentwicklung**

Die Weiterentwicklung von „Wunschkriterien“ des Pflichtenheftes wäre auf jeden Fall ein positiver Beitrag für die Software. Folgende Erweiterungen könnte man z.B. erwähnen:

1. Ein Werkzeug, welches zwei oder mehrere Spuren zusammenfügt (mergen), wäre eine reizvolle Bereicherung. Diese Erweiterung würde die Zahl unterschiedlicher Ansichten der Inputdaten erhöhen.

2. Die dauerhafte Einblendung von Attributen der Informationsobjekte wäre mit Sicherheit ein hilfreiches Feature für explorative Analysen. Die kritischen Punkte bei der Entwicklung wären die große Menge der darzustellenden Informationsobjekte und der resultierende Platzmangel.

3. Eine besonders wichtige Verbesserung wäre die grafische Repräsentation von Relationen[71]. Diese Relationen (Annotation- und ConceptRelation) bilden Beziehungen zwischen bestimmten Datensätzen der Inputdaten ab. Ihre Visualisierung wäre eine große Hilfe für die Analyse der Daten.

Die Abbildung 52 zeigt eine Vorstellung der Darstellung von Programmiertätigkeiten mit folgenden Erweiterungen:

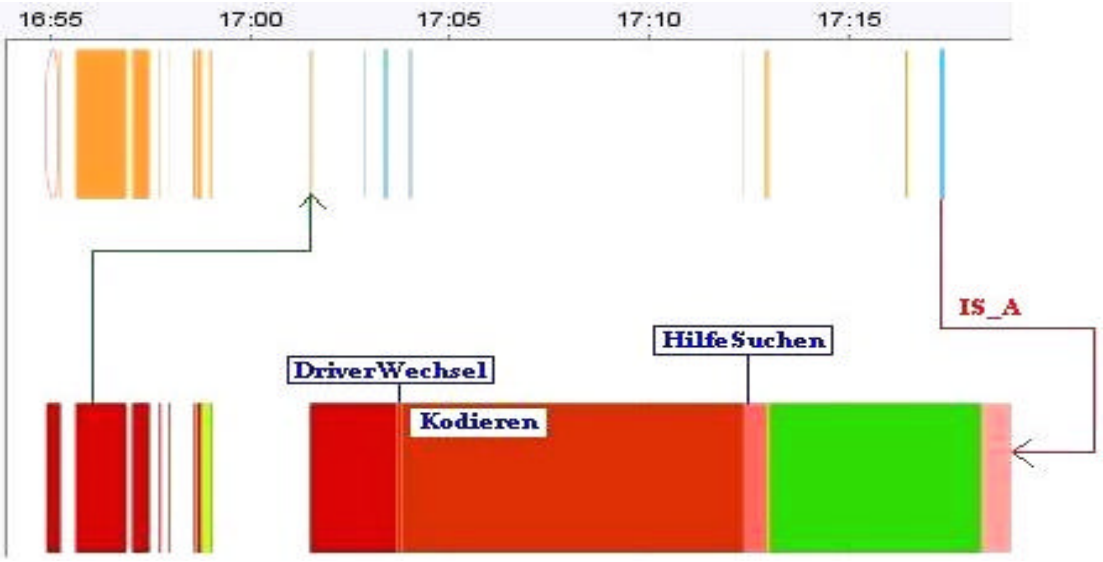


Abbildung 52 - Mögliche Erweiterungen der Darstellung von Programmiertätigkeiten und ihre Eigenschaften durch Generic TV

[71] Siehe Metamodell im Abschnitt III.2

- Dauerhaften Einblendung von Attributen: Dafür wäre den Einsatz des Konzepts Excentric Labeling [EL_I] sehr hilfreich. Die Idee ist es, wie im Abbildung 52 gemalt die Attributen außerhalb des dargestellten Informationsobjekts als Label darzustellen. Es gäbe aber ein Hauptproblem dabei, da in dem Fall nur den Platz zwischen zweier Spuren nutzbar wäre. Es besteht aber keine Garantie, dass genau dieser Platz ausreichend für die Darstellung der Labels wäre.

- Darstellung von Relationen: Ein Pfeil zwischen zwei oder mehreren Informationsobjekten, bei denen eine Beziehung besteht, wäre eine Möglichkeit die Relationen darzustellen. Diese Art und Weise aber sie darzustellen, würde mit einer ansteigenden Anzahl der darzustellenden Annotationen nur Wirrwarr in der Darstellung Erzeugen, bei der zwischen sich kreuzenden Pfeilen nicht mehr zu erkenne wäre.

Literaturliste

[A..]

- [AW04] Wolfgang Aigner: Visualization of time-oriented data 2004

[B..]

- [BH98] Helmut Balzert: Softwaretechnik
- [BH05] Heide Balzert: Lehrbuch der Objektmodellierung. Analyse und Entwurf mit der UML 2

[C..]

- [CMS99] Stuart Card/Jock Mackinley/Ben Shneiderman (Hg.): Readings in Information Visualization. Using Vision to Think. San Francisco: Kaufmann 1999
- [CW00] Information Visualization : Perception for Design by Colin Ware, San Francisco 2000
- [CB87] Mc Cormick Bruce H. (1987): Visualization in Scientific Computing. Computer Grafics
- [CD_I] URL: http://www.advsofteng.com Stand vom 06 Februar 2007

[D..]

- [DR99] Rolf Däßler: Informationsvisualisierung Stand, Kritik und Perspektiven 1999
- [DD00] Duden, Bibliografisches Institut & F.A. Brockhaus AG, Mannheim, 2000

[E..]

- [EA94] Endres, A.: Bedienoberflächen, Vorlesungsskriptum, WS 1994/95, TU München, Fakultät für Informatik, 1994
- [EC05] : URL: http://www.electrocodeogram.org Stand vom 06 Februar 2007
- [EL_I]: URL: http://www.cs.umd.edu/hcil/excentric/ Stand vom 13 Februar 2007

[G..]

- [GS67] The Discovery of Grounded Theory: Strategies for Qualitative Research.
- [GK96] Graw, K.-U., Die Auswahl von Daten zur Visualisierung, Preprints aus dem Fachbereich Informatik, Universität Rostock, 1996
- [GHJ01] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software. Addison-Wesley, Reading, Mass., 2001.
- [GM_I]: URL: http://www.gtd-manager.de Stand vom 06 Februar 2007

[H..]

- [HS_I]: URL: http://www.anecon.com/downloads/Qualitaetssicherung_Sneed.pdf Stand vom 06 Februar 2007

[I..]

- [ISO_I]: URL: http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=21922&ICS1=13&ICS2=180&ICS3 Stand vom 05 Februar 2007

[J..]

- [JS05] Jekutsch Sebastian, Micro-process of software development 2005
- [JS06] Jekutsch Sebastian, Promotionsvorhaben: Mikroprozessanalysen von Fehlerverhaltensmustern beim Programmieren
- [JF88] Ralph E. Johnson and Brian Foote. Designing reusable classes. Journal of Object-Oriented Programming, 1988
- [JFC_I] URL: http://www.jfree.org/jfreechart Stand vom 06 Februar 2007
- [JFCU_I] URL: http://www.jfree.org/jfreechart/users.html Stand vom 06 Februar 2007

[L..]

- [LW97] Lambert Wiesing, Die Sichtbarkeit des Bildes. Geschichte und Perspektiven der formalen Ästhetik 1997
- [LNS06]Lange, S.; Nocke, T., and Schumann, H.:Visualisierungsdesign – ein systematischer Überblick. Proceedings SimVis'06, Magdeburg, Germany, März, 2006

[M..]

- [ML98] W. Schroeder, K. Martin, B. Lorensen: "The Visualization Toolkit" Prentice Hall, 1998
- [ME95] MacEachren, A.M. 1995. How Maps Work. The Guilford Press, New York, USA
- [MJ99] Meyer J.-A., Visualisierung von Informationen, Gabler Verlag, Wiesbaden 1999

[N..]

- [NM97] G. Nielson, H. Hagen, H. Müller: "Scientific Visualisation" IEEE Computer Society Press 1997
- [NZ_I]: URL http://www.netzeitung.de/internet/366911.html Stand vom 06 Februar 2007
- [NC_I]: URL http://www.visualmining.com/products/graph-gallery.shtml Stand vom 06 Februar 2007

[S..]

- [SM00] H. Schumann, W. Müller: "Visualisierung - Grundlagen und allgemeine Methoden" Springer Verlag 2000.
- [SB96] Shneiderman, Ben (1996). The eyes have it: A task by data-type taxonomy for information visualizations. In Proceedings of Visual Languages (IEEE 1996)
- [SM03] Visualization methods for time-dependent data – An overview Schumann, H., Müller, W., 2003
- [SH05] Schumann, H.: Computergrafik: Künstliche Bilder aus dem Rechner 2005
- [SH06] Schumann, H Visualisierung abstrakter Daten Vorlesung SS 06 Uni Rostock Prof. Heidrun Schumann
- [SM04] Schumann, H.; Müller, W. :Informationsvisualisierung: Methoden und Perspektiven.; IT - Information Technology 3(2004), Oldenbourg-Verlag, 2004
- [SH04] Schumann, H.: Konzepte und Methoden der wissenschaftlichen Visualisierung.; Kartografische Bausteine, Band 26, TU Dresden, Juni, 2004
- [SS06] Salinger Stephan: Grundlagen und Basisstrukturen für die Visualisierung konzeptueller Beschreibungen von Programmiermikroprozessen
- [SR01] Information Visualization, Addison Wesley, Harlow 2001
- [SF05] Schlesinger Frank: Protokollierung von Programmiertätigkeiten in der Eclipse- Umgebung ECG, Diplomarbeit 2005

[P..]

- [PL06] Prechelt Lutz: Vorlesung Softwaretechnik WS06/07(FUB)

[T..]

- [TR_I] Themeriver URL: http://www.cg.tuwien.ac.at/courses/InfoVis/HallOfFame/2004/05_ThemeRiver/#examples Stand vom 16 Februar 2007
-

[W..]

- [WD00] D.L. Well, D. Widdrig: Managing Software Requirements – A Unified Approach, Addison Wesley (2000)
- [WIK_I] URL: http://de.wikipedia.org/wiki/Ganttchart Stand vom 06 Februar 2007

- [WM_I].URL: www.worldmapper.org Stand vom 21 Februar 2007

Anhang I– Aufgabenstellung

veritas
iustitia
libertas

TWiki > SE > ThesesHome > ThesisMicroprocessVisualization
Institut für Informatik, Freie Universität Berlin

Home
Research
Teaching
Theses
People
Links
Contact

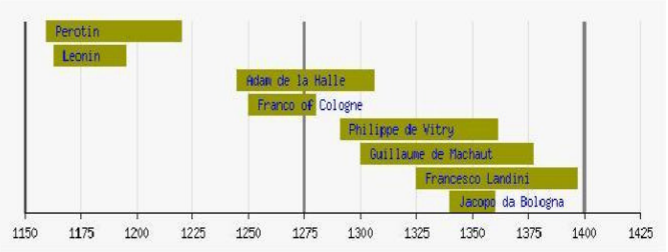
EN-to-DE
DE-to-EN
Explain word
Search

Software Engineering
Orga-Sachen
Rechnerbetrieb
Sandbox
Wiki FAQ

Impressum

Diplomarbeit

Visualisierung konzeptioneller Beschreibungen von Programmieraktivitäten



Zeitliche Visualisierung: Komponisten im Mittelalter (Quelle: Wikipedia)

Projekte

Micro-process of software development, Studying the characteristics of agile development processes?

Art

Anforderungs- und implementationsorientierte Arbeit mit der Chance, eigene (vor allem graphische) Ideen zu verwirklichen

Beschreibung

In den genannten Projekten werden Programmieraktivitäten von einzelnen oder Paar-Programmierern erfasst, indem Videosequenzen "kodiert" werden, d.h. die dort beobachtbaren Aktivitäten, Ereignisse und deren Eigenschaften werden erfasst und der Sequenz zugeordn vorhandene Datenbasis ist schon recht groß geworden, so dass erste Auswertungen der Daten möglich werden. Die zu der Kodierung bei Software bietet leider für eine explorative Suche in den Daten nach Mustern und Übereinstimmungen keine ausreichende Visualisierung.

Aufgabe

Im Rahmen dieser Diplomarbeit soll ein Softwaresystem erstellt werden, die die erfassten Kodierungen der Programmieraktivitäten geeigr visualisieren kann. Die Visualisierung orientiert sich an der Zeit, ähnlich der Zeitleisten in Geschichtsbüchern (siehe Bild). Die Darstellung so den darzustellenden Zeitraum und die darzustellenden Ereignisse, Aktivitäten und deren Eigenschaften auswählbar machen und darauffolgendend interaktiv mittels Scrollbar und weiteren Filterungen veränderbar präsentieren. Auch sollte es möglich sein, mehrere Zeitstrahle untereinander zu stellen.

Die Art und Weise der Visualisierung ist sehr abhängig von den Forschungsabsichten. Daher beinhaltet die Arbeit einen vergleichsweise hoh Zeitanteil zur Ermittlung der Anforderungen. Ein Aspekt davon ist die frühe Erstellung eines Prototyps. Desweiteren ist es Zwecks Beschleunigung der Arbeit wichtig, ein geeignetes Rahmenwerk/Bilbiothek zur Implementation der Visualisierung zu finden und einzusetze

Die Arbeit umfasst:

- Einarbeitung in den Forschungsgegenstand und Erhebung der Anforderungen
- Suche nach wiederverwendbaren Komponenten
- Entwurf und Realisierung eines schnellen Prototyps
- Änderung und Erweiterung des Prototyps aufgrund der Rückmeldung
- Dokumentation der Software, ihres Aufbaus und ihrer Bedienung
- Einsatz und Validierung des Werkzeuges im Realeinsatz
- Schriftliche Ausarbeitung und Vortrag

Durchführung

Die Diplomarbeit wird durchgeführt von NicolasNgandeu und betreut von StephanSalinger, SebastianJekutsch und LutzPrechelt.

Detailliertere Informationen sind hier zu bekommen:

- Anforderungsbeschreibung?
- Stand der Dinge, Wochenberichte, Planung
- Ideen, fremde Arbeiten, Links, etc.?

Action: Edit | Attach image or document | aEdit | More topic actions
View: Printable version | Raw text
Revisions: | r1.3 | > | r1.2 | > | r1.1 | Full page history

Anhang II Pflichtenheft

Dieser Abschnitt stellt das Pflichtenheft der Realisierung von Generic TV vor. Die vorgestellten Punkte werden nur Zielbestimmungen, Qualitätsanforderungen und Benutzeroberfläche. Die weiteren Abschnitte des Pflichtenhefts, wie z.B. die Einleitung, die Vorstellung der Daten und die Produktübersicht, die zu einem Pflichtenheft gehören, schon im anderen Kapiteln und Abschnitt der Ausarbeitung vorgestellt worden sind, werden sie in diesem Abschnitt nicht mehr erwähnt.

A - Zielbestimmungen

Definition: Der Begriff Annotation in diesem Dokument ist als ein Spurelement zu verstehen.

Anmerkungen:
genTV.ini ist eine Systemkonfigurationsdatei.

A.1 Musskriterien

Anforderungen, die unbedingt als Funktionalitäten der Software zu implementieren sind:

- **Verschiedene, aber eindeutig lesbare Formate der Inputdaten**:

  Das System kennt eindeutige Dateitypen, welche ein vom System eindeutig bekanntes Format besitzen. Es sind nur diese Dateien, die vom System verarbeitet werden können. Das Hinzufügen von neuen Dateitypen (d.h. mit einer neuen Formatierung) als Inputdaten der Software, ist ohne Erweiterungen an dem System nicht möglich.

- **Jedem darzustellenden Element ist ein eindeutiger Datentyp zuweisen**:

  Der Software sind eindeutige Datentypen bekannt. Die mögliche Visualisierung der Inputdaten durch das System basiert auf die Existenz dieser Datentypen. Aufgrund der Generizität, die von der Software garantiert werden soll, ist es unmöglich vorherzusagen, für welche Arten von Inputdaten (bzgl. die Formatierung und der Inhalt der Datei) die Software darstellen soll. Deswegen kann diese Software keine grafische Darstellung für jedes Element aus Inputdaten vorhersehen. Sondern die Software kann die grafische Darstellung nur für bestimmte Datentypen realisieren. Nach Absprache mit dem Auftraggeber sind bestimmte Datentyp definiert worden. Auf diese werden die generierten Inputdaten zugewiesen. Nur durch eine Typzuweisung (Datensatz ➔ Datentyp) wird eine Visualisierung der Inputdaten realisiert.

- **Eindeutige Datentypen darstellen**:

  Der Software bekannte Datentypen werden in der BOF durch zwei grafische Elemente dargestellt: Text und Grafiken. **Dabei werden Grafiken als Symbole, Icons und geometrische Figuren (Rechtecke, Ellipsen, Quadrate etc.) definiert**. Alle bekannten Datentypen, welche mit dem Auftraggeber erarbeitet wurden, sind folgendermaßen in der BOF dargestellt:

Datentyp	Darstellung in der BOF
Zeitintervall	Grafiken

Enum	Grafiken und (oder) Text
String	Text
Number (Integer, Float, Double etc.)	Grafiken und (oder) Text
Boolean	Grafiken und (oder) Text
Zeiger	Grafiken

- **Tabelle 1 - Definierte Datentypen und ihre Zuweisungen als grafische Elemente**

- **Zeitproportionale Darstellung von Annotationen:**

  ➔ Die Darstellung der Annotationen in der BOF wird in einem 2-dimensionalen Raum stattfinden. Dieser 2-dimensionale Raum wird in diesem Dokument **TVPanel** genannt. Die X-Achse stellt die Zeitachse dar. Sie wird vom System mit Zeitpunkten annotiert. Der Abstand zwischen Zeitpunkten auf der Zeitachse hängt von der Anzahl der darzustellenden Annotationen ab. Sie kann nicht vom Benutzer beeinflusst werden

  ➔ Das System stellt die Annotationen als geometrische Figuren dar. Eine Haupteigenschaft der Annotationen ist ihre Dauer. Sie ist durch Anfangs- und Endzeitpunkt charakterisiert. Darum sind Größe[72] und Position der geometrischen Figuren ausschließlich von der Dauer der Annotation abhängig. Das System realisiert eine Darstellung der geometrischen Figuren proportional zur Zeitachse.

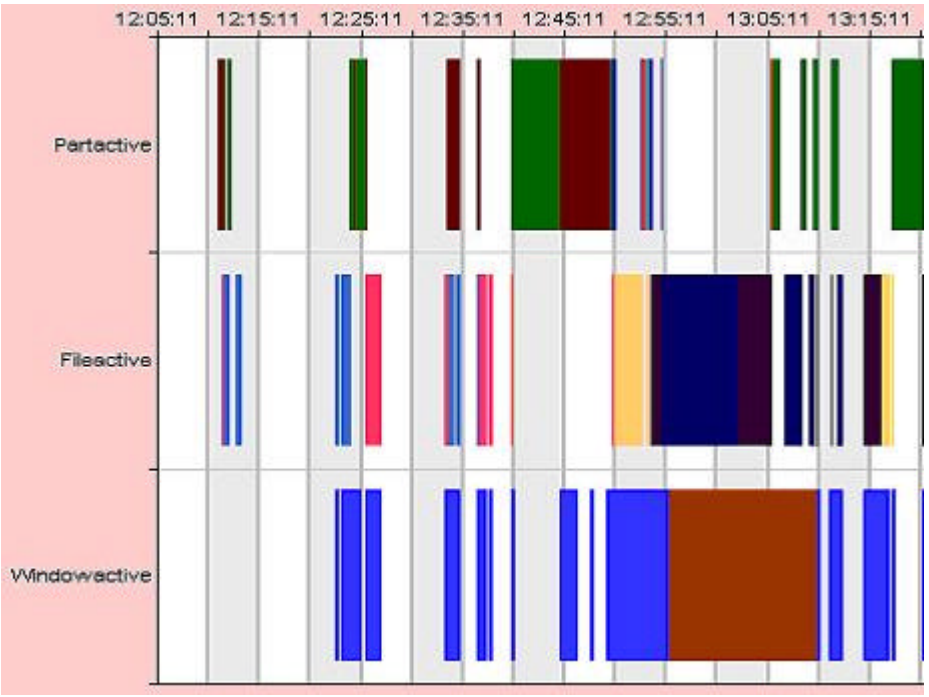


Abbildung 01 - Mögliche Sicht eine zeitproportionale Darstellung von Annotationen

- **Eindeutige Darstellung aller Annotationen:**

  Alle Annotationen werden vom System so dargestellt, dass jede Annotation eindeutig vom Benutzer erkennbar ist. Die eventuelle Schwierigkeit für den Benutzer Annotationen zu erkennen, kann durch folgende Punkte begründet werden:

[72] bzw. Länge

➔ Die Annotationen sind in Abhängigkeit zu ihren Anfangs- und Endzeitpunkten dargestellt

➔ Der Benutzer kann der Darstellung der Annotationen unterschiedliche Farben zu zuweisen.

➔ Sobald die Anfangs- oder Endzeitpunkte zweier Annotationen sich überschneiden, entstehen Unterscheidungssprobleme für den Benutzer. Deshalb werden alle Überschneidungen vom System erkennbar gemacht.

- **Auswahl und Laden der darzustellenden Dateien:**

  Der Benutzer kann über ein Dateiauswahlmenü die Inputdatei auswählen. Sie wird dadurch im System geladen und dargestellt. Nach der Derstellung der Inputdatei, kann der Benutzer beliebig oft und jederzeit neue Dateien im System laden. Durch das Laden von neuer Datei wird die aktuelle Darstellung um die neuen geladenen Datensätzen erweitert.

- **Alle darzustellenden Annotationen erkennen:** aus jedem Dateityp, muss das System in der Lagen sein, alle darzustellende Annotationen zu erkennen.

- **Farbezuweisung der darzustellenden Annotationen:**

  Bevor das System gestartet wird, muss der Benutzer in der Systemkonfigurationsdatei **GenTV.ini** eine Farbezuweisung erstellen. Zu jeder darzustellenden Annotation ist eine Farbe zuzuweisen. (z.B. alle Konzepte mit Namen „Muster" sollen Rot angezeigt werden) Falls kein Farbeintrag für eine Annotation in der **GenTV.ini** vorhanden ist, wird das System beim starten eine Defaultfarbe für die Darstellung zuweisen. Ansonsten werden die Annotationen mit der vom Benutzer vordefinierten Farbe grafisch dargestellt. Die in der **GenTV.ini** gesetzte Farben sind dynamisch (in der Laufzeit) vom Benutzer veränderbar. Diese dynamischen Veränderungen werden nur durch eine explizite Benutzeranweisung des dauerhaft in der **GenTV.ini** gespeichert.

- **Zuweisung von geometrischen Figuren zu den darzustellenden Annotationen:**

  Bevor das System gestartet wird, kann der Benutzer die geometrischen Figuren für die Darstellung von Annotationen (z.B. Konzept 1 als Rechteck, Konzept 2 als Ellipse etc.) in der **GenTV.ini** festlegen. Das System kann folgende geometrischen Figuren darstellen[73]: Rechtecke, Ellipsen, abgerundete Rechtecke. Das System setzt die geometrische Figur Rechteck als Standard für alle Annotationen

[73] als Wunschkriterium erweiterbar

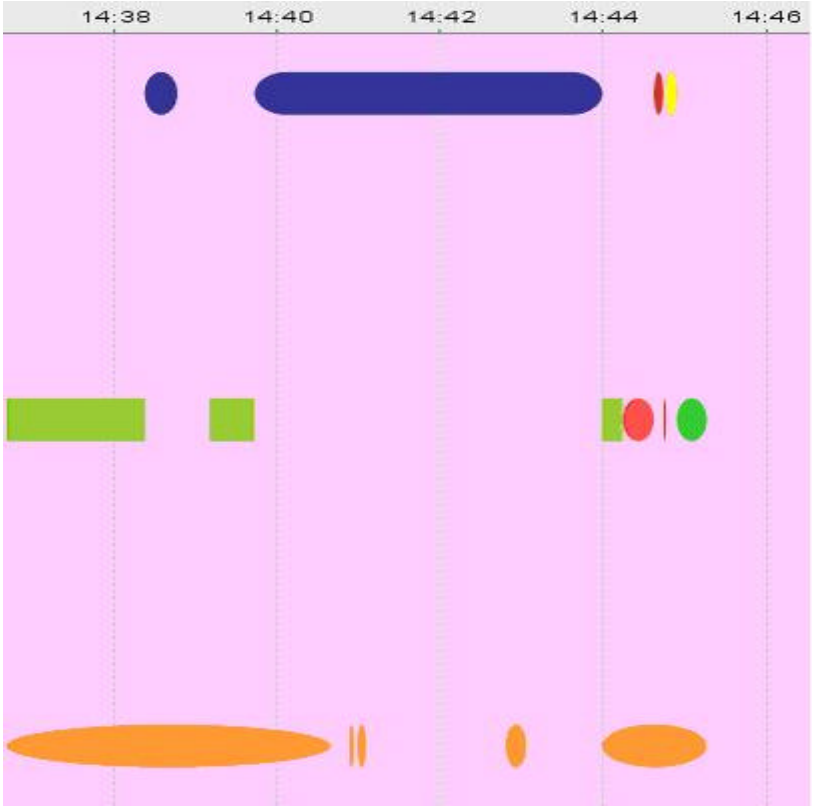


Abbildung 532 - Darstellbare geometrische Figuren

- **Zuweisung der Füllung der geometrischen Figuren:**

  Der Benutzer kann über die Systemkonfigurationsdatei **GenTV.ini** entscheiden, ob die zugewiesenen geometrischen Figuren gefüllt oder leer in der BOF angezeigt werden. Falls sie vom System voll dargestellt werden, wird der innere Raum der geometrischen Figuren, mit der vom Benutzer gesetzten Farbe gefüllt (Siehe Bild im vorherigen Punkt). Sonst (sie sollen leer vom System dargestellt werden) nimmt der innere Raum der geometrischen Figuren die Hintergrundfarbe des **TVPanels** an. Die Hintergrundfarbe des TVPanels kann dynamisch vom Benutzer über ein Kontextmenü gesetzt werden.

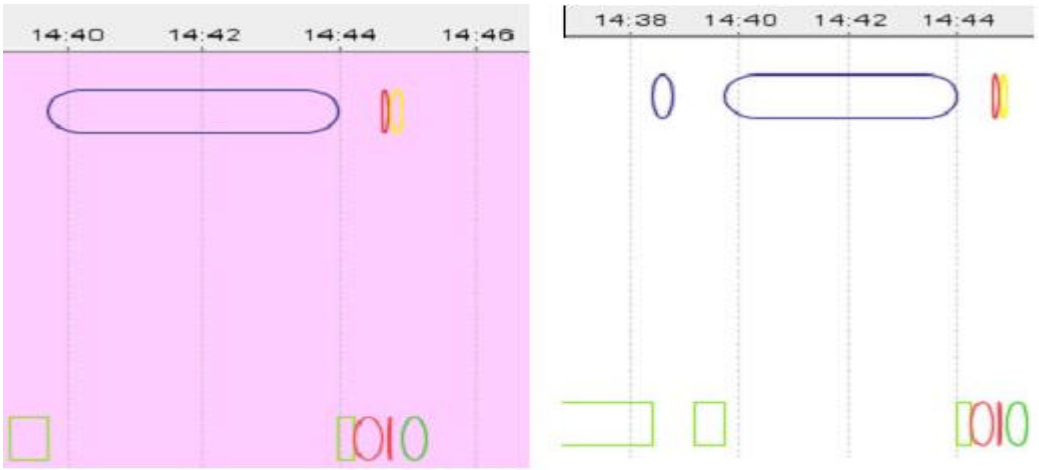


Abbildung 03 - Beispiel von leeren Füllungen

- **Quantitative Visualisierung:**

  Das System kann eine quantitative Visualisierung der Inputdaten realisieren. Noch zu definierende Merkmale repräsentieren die unterschiedlichen Werte (numerische oder Boolesche Werte) mancher Annotationen. Das System kann aus den geladenen Dateien erkennen, welcher Datensatz einen Wert besitzt. Dementsprechend wird dieser Wert in der dazugehörige Annotation darstellt.

Abbildung 04 - Mögliches Beispiel einer quantitativen Visualisierung

- **Darstellung bzw. Einblendung (auf Wunsch des Benutzers) von zusätzlichen Informationen über die dargestellten Annotationen**:

  Die dargestellten Annotationen besitzen weitere Attribute bzw. Eigenschaften (Name, Dauer, etc.). Nur durch Mausklick sind diese Attribute in der BOF darzustellbar. Das System stellt dem Benutzer als Musskriterium nur eine Möglichkeit zur Verfügung. Alle Attribute der Annotation(en) werden per Mauseklick veranschaulicht. Das System generiert ein kleines Dialogfenster mit allen Attributen der Annotation.

- **Zoomen in einem ausgewählten Bereich der Darstellung**:

  Der Benutzer soll die Möglichkeit haben, in einem mit der Maus markierten Bereich hineinzuzoomen. Das System muss in diesem Fall eine neue vergrößerte Darstellung, der momentan dargestellten Annotationen generieren.

- **Filtern der Daten der aktuellen Darstellung**:

  In der BOF wird der Benutzer über einen Buttonklick ein Filtertool starten können. Dieses Filtertool soll bestimmte Annotationen aus der Darstellung ein- bzw. ausblenden. Dies sind Annotationen, deren Attribute vom Benutzer selektierte Filterkriterien erfüllen. Um die Filterkriterien zu selektieren, muss der Benutzer folgende Schritte im Filtertool durchführen:

1. **Spur auswählen**: Nachdem das Filtertool vom Benutzer gestartet worden ist, wird das System zuerst alle dargestellten Spuren auflisten. Sie werden durch ihre Namen erkennbar sein. Der Benutzer wird eine Spur auswählen müssen.
2. **Filterkriterien auswählen**: Sobald eine Spur, vom Benutzer ausgewählt worden ist, wird das System alle Attribute der Annotationen, die sich in dieser Spur befinden auflisten. Aus dieser Auflistung wird der Benutzer ein Attribut auswählen müssen (z.B. Name oder Anfangszeit). Ab diesem Zeitpunkt wird aus diesem Attribut ein Filterkriterium des Filtertools.
3. **Wert für die Filterkriterien auswählen**: Der Benutzer bekommt vom System für das gewählte Filterkriterium eine Auflistung von allen im System vorhandenen Werten, die das gewählten Filterkriterium annehmen könnte. Der Benutzer muss dann für das Filterkriterium einen Wert auswählen.
4. **Selektion bestätigen und Filter auf die Darstellung anwenden**: Wenn die vom Benutzer gewählten Spuren, Filterkriterien und Werte durch ihn bestätigt worden sind, wird das System die Filterkriterien auf die Darstellung anwenden. Die Anwendung der Filterkriterien verändert die Darstellung. Die Annotationen, die die Filterkriterien nicht erfüllen, werden ausgeblendet.Jene, die die Filterkriterien erfüllen, werden eingeblendet.

Durch Wiederholung dieser Schritte können mehrere Spuren oder die gleiche Spur mit mehrere Filterkriterien filtern.

- **Erzeugung in der Laufzeit von neuen Spuren**:

  Der Benutzer wird über ein in der BOF vorhandenes Menü die Möglichkeit haben, aus den aktuell dargestellten Annotationen neue Spuren zu generieren. Dazu wird der Benutzer, durch von ihm definierte Filterkriterien, Annotation aus schon vorhandenen Spuren „klonen“. Diese Kopie wird neue Spur der Darstellung hinzufügt.

- **Extraktion der Konstellation (auch Design genannt) der aktuellen Darstellung in einem eindeutigen Formate:**

  Die Konstellation jeder Darstellung[74] kann durch Buttonklick jederzeit extrahiert werden. Der Benutzer wird hierfür durch ein Dateiauswahlmenü eine Datei erzeugen müssen. In dieser Datei werden alle für das System benötigten Informationen enthalten sein. Dadurch ist der Benutzer jederzeit in der Lage, wieder für die gleichen Inputdateien eine Darstellung mit seiner extrahierten Konstellation zu erstellen.

- **Scrollbare (Links/Rechts, Oben/Unten) Darstellung**:

  Falls eine vom System generierte Darstellung für die BOF zu groß ist, wird der Benutzer die Möglichkeit haben, diese von links nach rechts und von oben nach unten zu scrollen.

- **Dynamische (in der Laufzeit) Änderung der Farbe der dargestellten Annotationen**:

[74] vorhandene Spuren, geometrische Figuren der Annotation, Farben der Annotationen, gefilterte Annotationen usw.

Mit Hilfe des in der BOF enthaltenen Menüs, wird der Benutzer die Farbe von mehreren dargestellten Annotationen ändern können. Dabei werden alle Annotationen, die in dieser Farbe dargestellt sind. Ihre Erläuterungen in der Legende verändern sich mit.

- **Möglichkeit für den Benutzer, alle von ihm durchgeführten Tätigkeiten rückgängig zu machen „Undo“:**

  Das System kann durch einen Buttonklick Benutzeraktionen[75] rückgängig machen. Die Visualisierung besitzt jederzeit eine Konstellation K_x, welche durch bestimmte Eigenschaften der Darstellung charakterisiert ist[76]. Diese Konstellation wird durch jede Benutzerinteraktion veränder. Es gilt also K_{x+1}. Mit der „Undo“-Funktionalität kann der Benutzer aus der Konstellation K_{x+1}, die Konstellation K_x zurückherhalten.

- **Darstellung einer eindeutigen Legende**:

  Für jede generierte Darstellung wird das System eine dazugehörige eindeutige Legende erstellen. Diese Legende wird die Farbe jeder dargestellten Annotation erläutern. Die Namen der Annotationen werden neben der Farbe zu erkennen sein. Da sehr viele Annotationen gleichzeitig in der BOF dargestellt werden, sollen die Namen in der Legende (in die dazugehörige Farbeinformation) alphabetisch sortiert.

- **Extrahieren der aktuellen Darstellung**:

  Der Benutzer wird die Möglichkeit haben, eine von ihm generierte Darstellung[77] auf verschiedene Arten zu extrahieren:

  ➔ Ausdrucke: Einen Ausdruck der aktuellen Darstellung erhält der Benutzer durch die Druckauswahl. Verfügt der Benutzer über einen PDF-Drucker, wird auch ein Ausdruck der Darstellung bzw. die Erstellung einer Datei mit der Darstellung in diesem Format möglich sein.
  ➔ Speicherung: Durch ein Dateiauswahlmenü kann der Benutzer die aktuelle Darstellung als „*Portable Network Grafics*“ (*.png)–Datei extrahieren. Die erzeugte Datei ist eine Bilddatei. Dieses Format kann mit jedem grafischen Programm bearbeitet werden.

A.2 Wunschkriterien

Anforderungen, die nicht unbedingt umgesetzt werden müssen und eventuell erweitert werden sollten.

- **Nicht-Darstellung von bestimmten Zeitintervallen auf der Zeitachse**:

  Der Benutzer wird mit einem für die Zeitachse gedachten Filtertool Zeitintervalle (einen oder mehrere Zeitpunkte) ausblenden können. Dadurch werden die Annotationen der ausgeblendeten Zeitintervalle nicht dargestellt.

[75] Veränderungen der Farbe von Annotationen, Hinzufügen von neuen Spuren, Filtern von Spuren etc.
[76] Länge der Zeitachse, dargestellte Annotationen, Farbe dieser Annotationen, Filterkriterien usw.
[77] Annotationen, eingeblendete Informationen, Legende

- **Verschieben von Spuren (Drag und Drop)**:

  Dder Benutzer wird mit der Maus, dargestellte Spuren in der BOF verschieben können.

- **Zusammenfügen von Spuren:**

  Das System kann, aus zwei oder mehreren Spuren, eine einzige Spur generieren. Über ein BOF-Menü mit einer Auflistung aller vorhandenen Spuren des Systems, kann der Benutzer bestimmte Spur auswählen. Nach Bestätigung durch den Benutzer, wird das System aus allen Annotationen, die sich in den ausgewählten Spuren befinden, eine neue Spur erzeugen. Dabei wird der Benutzer einen Namen für die neue generierte Spur eingeben müssen.

- **Angebot von den geometrischen Figuren für die Darstellung von Annotationen erweitern:**

  Das System kann mehr als die drei geometrische Standardfiguren darstellen.

- **Darstellung von Relationen zwischen dargestellten Elementen des Datenbestandes**:

  Mittels eines Buttons in der BOF soll der Benutzer die Möglichkeit haben, sich die Relationen der darzustellenden Annotationen anzeigen zu lassen. Es existieren verschiedene Arten von Relationen zwischen den Annotationen. Deshalb muss eine farbliche Unterscheidung bei ihrer Darstellung möglich sein. Der Benutzer wird also verschiedene Farben für die unterschiedlichen Relationen über **GenTV.ini** setzen müssen. Dabei muss die Legende der Darstellung dementsprechend ergänzt werden.

- **Dauerhafte Ein- bzw. Ausblendung von Attributen in den dargestellten Annotationen:**

  Über ein BOF-Menü soll der Benutzer die Attribute der Annotationen dauerhaft ein- bzw. ausblenden können.

- **Bereitstellung eines Editors, um Kommentare zu einer Darstellung hinzuzufügen**:

  Durch Buttonklick in der BOF wird dem Anwender ein Editor zur Verfügung gestellt. Bestimmte Darstellungen können so persönliche Kommentare hinzugefügt werden. Aus diesen Kommentaren wird eine Text-Datei erzeugt. Diese muss genauso heißen, wie die im System geladene Datei. Der Speicherungsort der Datei wird auch nicht dem Benutzer überlassen. Die Vergabe soll systemgesteuert sein. Beim Starten des Systems wird das System erkennen, ob in bestimmten Dateien (Daten)[78] Kommentare gespeichert sind. Auf Benutzerwunsch können vorhandene Kommentare im Editor geladen und eventuell bearbeitet werden.

A.3 Abgrenzungskriterien

[78] Sie sind durch ihre Namen eindeutig.

Merkmale, welche die Software nicht aufzuweisen braucht.

- **Automatisierte Erkennung von Mustern**:

  Das System muss nicht in der Lage sein, eventuelle Muster in den dargestellten Annotationen zu erkennen. Dies bleibt die Aufgabe des Benutzers.

- **Editieren (Veränderungen) des Datenbestandes**:

  Der vorhandene Datenbestand muss nicht vom System verändert werden können. Nur eine Visualisierung des Datenbestandes muss realisiert werden.

B . Qualitätsanforderungen

Produktqualität	Sehr gut	Gut	Normal	Nicht relevant
Funktionalität				
Richtigkeit		X		
Sicherheit				X
Zuverlässigkeit				
Wiederherstellbarkeit				X
Benutzbarkeit				
Verständlichkeit		X		
Erlernbarkeit			X	
Bedienbarkeit		X		
Effizienz				
Zeitverhalten		X		
Verbrauchsverhalten			X	
Änderbarkeit				
Modifizierbarkeit	X			
Analysierbarkeit	X			
Stabilität			X	
Prüfbarkeit			X	
Wartbarkeit	X			

Tabelle 02 - Qualitätsanforderungen an das System

C. Benutzeroberfläche

In diesem Abschnitt werden grundlegende Anforderungen an die Benutzeroberfläche festgelegt:

- Die Benutzeroberfläche soll alle definierten Produktfunktionen unterstützen.
- Die entsprechenden Produktdaten müssen, wie in Punkt Zielbestimmung, als definierte Formen dargestellt werden.
- Alle Dialoge der Anwendung sind fensterbasiert zu entwerfen.
- Das Fenster-Gestaltungsregelwerk ist zu beachten.
- Die Benutzeroberfläche soll möglichst intuitiv gestaltet sein. Hierdurch können nennenswerte Einarbeitungszeiten vermieden werden.
- Die Benutzeroberfläche ist auf Mausbedienung auszulegen. Ohne Maus ist die Systembedienung ist unmöglich.
- Eine Erweiterung mancher Punkte der Navigation mit Tastatur-Kurzbefehlen soll aber jederzeit möglich sein.
- Das System den BOF-Inhalt soll nach jeder Benutzerinteraktion aktualisieren. Vorausgesetzt, dass diese Interaktivität eine Veränderung der Darstellung verursacht hat.
- Die meisten Funktionalitäten[79] werden mit Icons der Toolbar realisiert (Siehe Abbildung 57).
- Über das Informationspanel können vom System manche Auflistungen (z.B. gefilterte Annotationen) angezeigt werden.

[79] z.B. Dateiladen, Filtertool aufrufen, Farbe der Annotationen ändern, den dargestellten Zeitraum ändern, Drucken, „Undo“ etc.

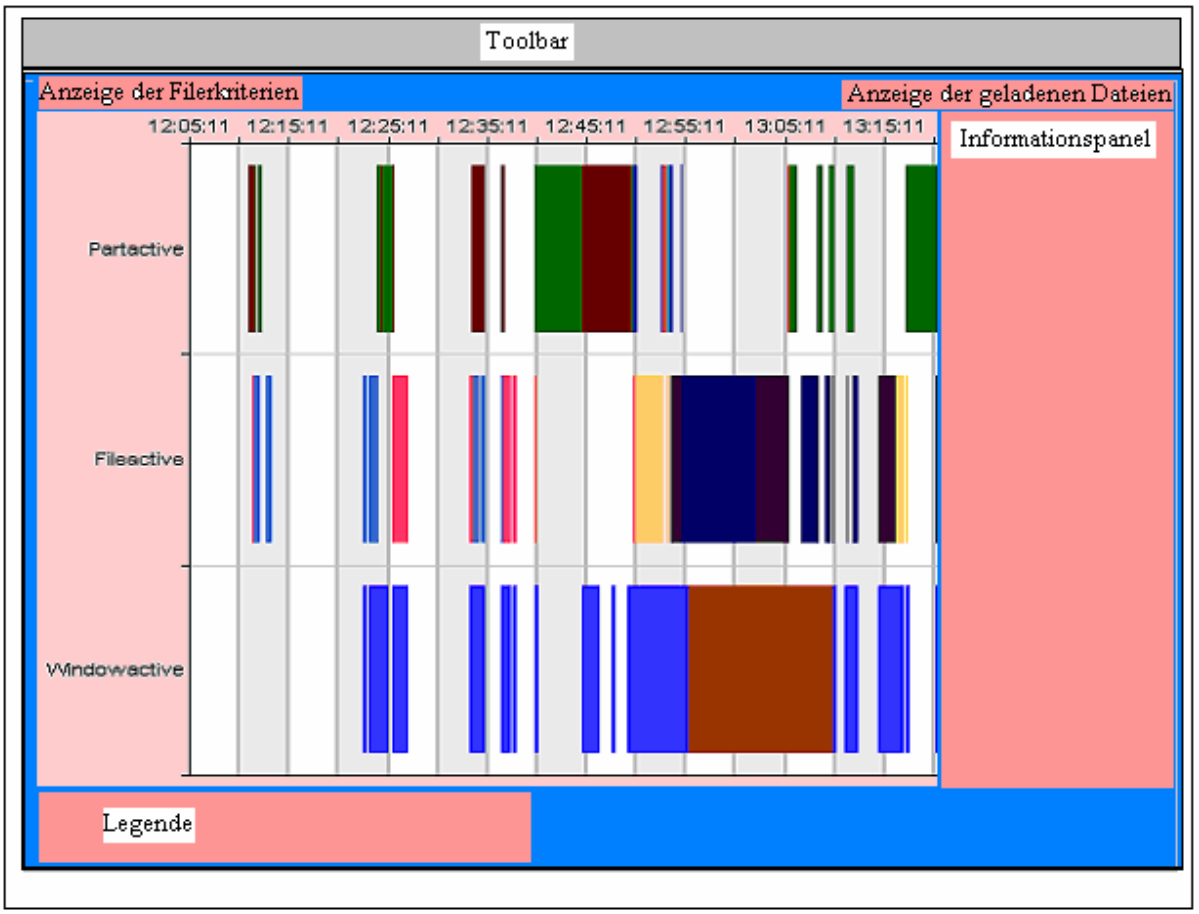


Abbildung 01 - Mögliches Aussehen der Benutzeroberfläche von Generic TV

Anhang III Analyse der grafischen Elemente

In diesem Abschnitt werden grafische Elemente analysiert. Mit der Zielsetzung, eine Auswahl für ihre Anwendung Generic TV zu finden. Das Dokument wurde am Anfang des Projekts erstellt.

Der zentrale Punkt dieser Analyse ist durch die Zusammenarbeit zwei Menge charakterisiert. Die Menge A (Ausgabemenge), die aus grafischen Elementen besteht. Die zweite Menge ist die Menge E (Eingabemenge), die aus Objekten besteht. Die Objekten der Menge E stellen dar darzustellende Objekte dar. Die Menge E ist also eine Sammlung von Objekten, die grafische dargestellt werden. Es wird in dieser Analyse versucht eine Auswahl von grafischen Elementen durchzuführen. Gesucht sind grafische A Elementen, die am geeignesten sind um E Elemente darzustellen.

Gewunscht sind Abbildungen zwischen E-Elementen und A-Elementen zu erzeugen. Solche Abbildungen wären gleich zu stellen, wie die Auswahl ein

Die Menge A

Die Ausgabemenge A besteht aus grafischen Elementen:

A = {X / X ist ein grafisches Element}

Diese Elemente bilden die Elemente einer Eingabemenge E ab. A enthält die grafischen Elemente, die das Visualisierungstool benutzt. Daraus ergeben sicher möglicherweise folgende Kernfragen:

1. **Welche grafischen Elemente werden gebraucht?**
2. **Wie werden x-beliebige Elemente der Menge E grafisch dargestellt?**
3. **Womit werden die x-beliebigen Elemente der Menge E grafisch dargestellt?**

A Zur ersten Kernfrage

Um die Ausgabemenge A zu erzeugen, muss die Eingabemenge E bekannt sein. Man verschafft sich einen Überblick von allen in E vorkommenden Elementen. Dieser Überblick ist deshalb wichtig, weil nur so eine Auswahl von grafischen Elementen (für die Menge A) realisiert werden kann. Aus dem (in Abschnitt III.2 vorgestellten) Metamodell ergeben sich die Elemente der Menge E. Generic TV muss Annotation-Objekte[80] darstellen können. Aus dieser Notwendigkeit folgt die Frage:

Die erste Kernfrage bezieht sich darauf, auf welche Rohdaten zurückgegriffen wird. Zur Beantwortung ist eine kurze Vorstellung des Begriffs „Grafisches Element“ nötig.

[80] Concept-, Focus-, Value- und AnnotationRelation-Objekte

A1 Grafische Elemente

Grafische Elemente können Text und/oder Grafiken sein.

A1.1 Text

Wenn die unten aufgeführten Regeln befolgt werden, ist die Textform sehr informativ.

- **Auf gute Lesbarkeit achten**
  - einfache Schrifttypen wählen
  - Lesegewohnheit beachten.
    - Immer von links nach rechts schreiben.
    - Groß- und Kleinbuchstaben verwenden.
- **Die vier „Verständlichmacher“ beachten**
  - Einfachheit
    - Geläufige Wörter verwenden.
    - Kurze Sätze bilden
  - Gliederung / Ordnung
    - Überschriften verwenden
    - Optische Blöcke bilden
  - Kürze
    - Aussagen auf das Wesentliche beschränken
  - Zusätzliches Stimulanz
    - Farben einsetzen
    - Beispiele geben

A1.2 Grafiken

Grafiken sind Figuren im herkömmlichen Sinne (Rechtecke, Ellipsen, Quadrate, etc.) oder zusammengesetzte Figuren (Symbole, Icons, Diagramme usw.). Grafiken als Gestaltungselemente können sehr hilfsreich sein. Die Redewendung, dass ein Bild mehr als tausend Worte sagt, trifft oft zu.

Es gibt aber nicht nur Vorteile bei der Benutzung von Grafiken. Für die Brockhaus-Enzyklopädie ist z.B. ein Symbol, *„ein sinnlicher Träger einer Bedeutung oder*

Meinung“[81]. Die Interpretierung von Grafiken kann missverständlich sein. Deshalb muss die Auswahl der Grafiken bei einer Visualisierung sehr gut überlegt werden.

Das Buch „Symbole-Statistk-Computer-Design. Otto Neurath's Bildpädagogik im Computerzeitalter“ (Wien, 1991) von Karl H. Müller enthält definierte Regeln zur Symbolgestaltung:

- *Zeichen müssen für sich selbst sprechen*
- *Zeichen sind farbunabhängig zu halten*
- *Die Anwendung aller Farben des gesamten Farbspektrums ist für die Zeichendarstellung erlaubt*
- *Zeichen sollen im Wesentlichen perspektivenlos gestaltet sein*
- *Zeichen müssen kombinierbar sein*
- *Zeichen sollen, obwohl auf das Wesentliche reduziert, trotzdem einen dauerhaften Eindruck beim Betrachter hinterlassen*

Diese Faustregeln sind circa 75 Jahre alt. Trotz der beeindruckenden technischen Entwicklung, sind sie immer noch aktuell. Zum Beispiel werden die Grafiken zur Bedienung von modernen Betriebsystemen dementsprechend gestaltet.

B Zur Kernfrage 2

Die zweite Kernfrage zielt auf eine allgemeine Begrenzung von Objekten der Eingabemenge E. Um eine Antwort zu finden, ist eine Klassierung der E-Objekte notwendig.

❖ Umsetzung von Eingabemenge-Objekten zur Datentypen

In Generic TV muss der generische Input für das Visualisierungstool garantiert sein.

Die Elemente aus der Eingabemenge E sind die Objekte, welche Generic TV visuell dargestellt. Ein sinnvoller Ansatz wäre eine wohldefinierte Klassierung der E-Elemente. Sie stellt in Generic TV einen Teilaspekt zur Sicherstellung der Generezität dar.

Die Auswahl der grafischen Elemente für die Visualisierung wird nicht durch die E-Objekte bestimmt. Die Generierung der Inputdaten wird sich mit großer Wahrscheinlichkeit in Zukunft erweitern. Dadurch könnten neue E-Objekte entstehen. Deshalb ist von der direkten Abbildung von E-Objekten zu grafischen Elementen abzuraten.

Die Verwendung von Datentypen zur der Klassierung der E-Elemente, wäre ein gangbarer Weg. Mit den aktuellen Inputdaten, kann man schon folgende Datentypen auflisten:

- Zeitintervall,

[81] *Brockhaus Die Enzyklopädie*, Ausgabe 20 (1996-1999)

- Enum,
- String,
- Number (Integer, Float, Double),
- Boolean,
- Zeiger.

C Zur dritten Kernfrage

Die dritte Kernfrage beschäftgit sich damit, welche grafischen Elemente für welche Datentypen geeignet sind.

❖ Auswahl der grafischen Elemente zur Darstellung der E-Objekte

Nachdem die verwendeten Datentypen ermittelt wurden, ist die Beantwortung von Kernfrage drei möglich.

Es folgt eine Diskussion über die geeignetsten grafischen Mittel zur Darstellung der Datentypen.

◆ Datentyp Zeitintervall

Die E-Objekte, welche durch den Datentyp Zeitintervall repräsentiert werden, haben gemein, dass sie jeweils einen Anfangs- und Endzeitpunkt. Dieser Datentyp ist ein unverzichtbarer Punkt für die Visualisierung. Die Darstellung des Datentyps Zeitintervall muss abhängig zu einer Zeitachse realisiert werden. Deshalb müssen die gewählten grafischen (zeitabhängigen) Elemente variabel darstellbar sein. Je größer die Differenz zwischen Anfangs- und Endzeitpunkt eines E-Objekts, desto größer ihre Repräsentation auf der Zeitachse.

Der Datentyp Zeitintervall kann durch Text, Grafik oder Text und Grafik ausgegeben werden.

Repräsentation durch Text

Die Textdarstellung in Generic TV des Datentyps Zeitintervall würde zu viele Regeln, die in Abschnitt A.1.1 vorgestellt wurden, verletzen. Beispiele für diese Regelverletzungen wären:

- Eine korrekte Skalierung der Schrift wäre über lange Zeiträume würde die Regel der guten Lesbarkeit zuwiderlaufen
- Eine farbliche Stimulanz ist in der Textdarstellung schwer zu realisieren.

Repräsentation durch Grafik

Die Grafikdarstellung in Generic TV des Datentyps Zeitintervall ist gut geeignet. Es besteht aber Grafikdarstellung den Nachteil, dass der Aufbau der Grafik stark die Einhaltung der Neurathsche Regeln beeinflusst. Das bedeutet, dass im allgemein gilt: Je komplizierter der Aufbau der Grafik, desto weniger Anschaulichkeit in der Darstellung. Aus diesem Grund

werden in Generic TV einfache Figuren (Rechtecke, Quadrate, Ellipsen und ähnliches) bevorzugt.

Durch Grafiken können im Allgemeinen skalierte Informationen sehr gut wiedergegeben werden. Zwei sehr aussagekräftige Beispiele dafür sind in den Abbildungen 58 und 59 zu sehen.

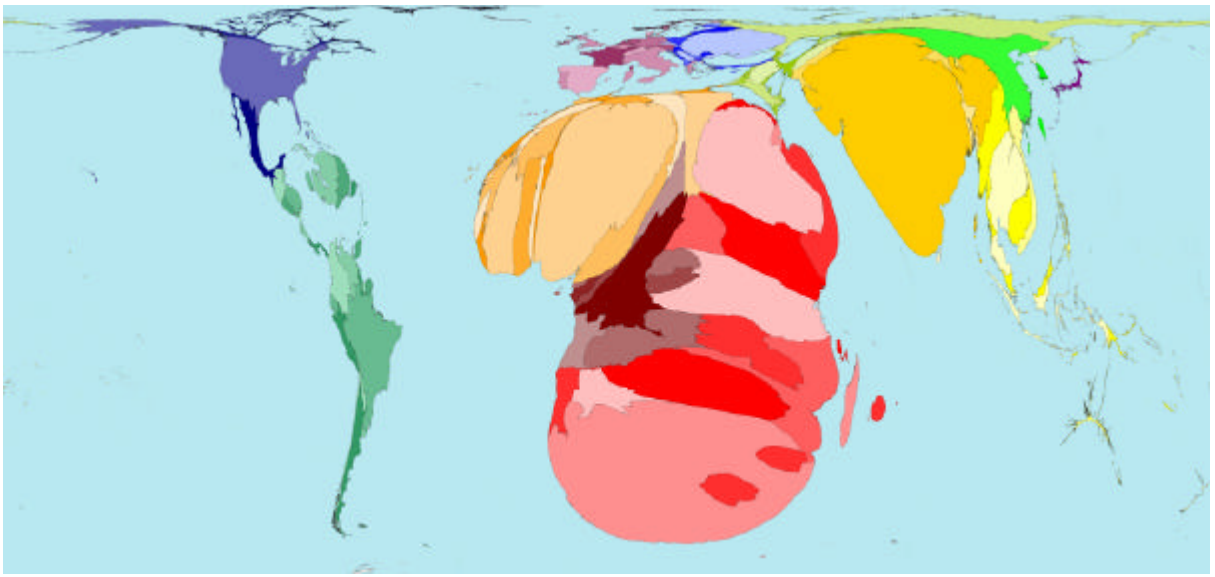


Abbildung 02 – Weltkarte Ausbreitung des HI-Viruses im Jahr 2003[WM_I]

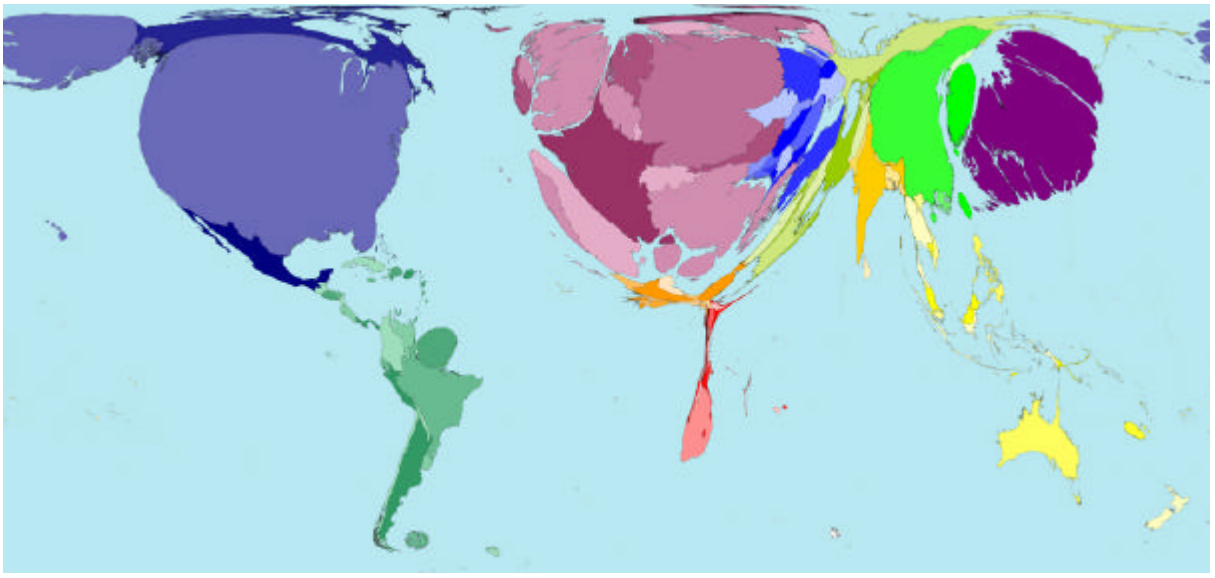


Abbildung 03 – Weltkarte der Ausgaben fürs Gesundheitswesen im Jahr 2001 [WM_I]

Bei der Darstellung des Datentyps Zeitintervall spielt die Skalliertbarkeit der angewendeten grafischen Elemente (Text oder Grafik) eine entscheidende Rolle. Nur durch das grafische Element Grafik ist die effektvolle Wiedergabe des Datentyps Zeitintervall möglich.

Aus diesem Grund eine Entscheidung für Diagramm scheint selbstverständlich zu sein. Die grafischen Symbole eines Diagramms (Balken, Kreis, Stern, Kästchen etc.) bieten viele Möglichkeiten für die Darstellung von Zeitintervallen. Außerdem ist die Darstellung von Diagrammen parameterabhängig (z.B. Zahl bei Balkendiagrammen). Alle Datentypen

(Zeitintervall, Enum, Boolean usw.) haben an die grafischen Elemente bestimmte Anforderungen (z.B. Farbe setzen, Interaktionsmöglichkeiten etc.). Diese Anforderungen erzwingen die Überprüfung der ausgewählten grafischen Elemente. Die grafischen Elemente müssen bei der Darstellung des Datentyps Zeitintervall unter anderem folgende Punkte erfüllen:

- Sie müssen verschiedene Farbe besitzen.
- Die Objekte von Typ Zeitintervall könnten weitere Eigenschaften (wie z.B. Name, Wert, etc.) besitzen. Die grafische Darstellung dieser Informationen muss realisierbar sein.
- Der Datentyp Zeitintervall soll numerische oder Boolesche Werte repräsentieren können.

Der Einsatz von Diagrammen für die Darstellung des Datentyps Zeitintervall scheint sehr gut geeignet.

B.2 – Enum

Enum ist der Datentyp für Aufzählungen. Für die Darstellung von Aufzählungen könnte man die grafischen Elemente Text und Grafiken verwenden. Eine Kombination von Text und Grafiken wäre denkbar. Die grafische Darstellung von Enum sollte kontextabhängig sein.

B.3 – String

Der Datentyp String ist sehr wichtig. Die Realisierung eines Visualisierungstools ohne Beschriftung ist nicht denkbar. Die natürliche Wahl sollte deshalb auf das grafische Element Text fallen. Die Regeln in Abschnitt A.1.1 sind die Grundlage für die Entscheidung.

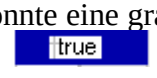
B.4 – Number

Der Datentyp Number repräsentiert die numerischen Werte der Inputdaten. Es gibt verschiedene Möglichkeiten der grafischen Umsetzung. Es ist z.B. möglich, Datenwerte durch Größen- oder Farbveränderung des grafischen Elements auszugeben. Bildet man z.B in Generic TV einen Datenwert auf die Größe eines grafischen Elements ab, ergibt sich folgende Analogie: Je größer der Datenwert, desto höher der grafische Darstellung. Das Variieren der Höhe des grafischen Elements Grafik ist ausagekräftiger als das Höhenvarieren beim grafischen Element Text. Aus diesem Grund ist die Auswahl von Grafiken zu Darstellung des Datentyp Number am besten geeignet.

B.5 – Boolean

Die Darstellung des Datentyps Boolean wird als quantitative Darstellung betrachtet, Es werden Boolesche Werte der Inputdaten wiedergegeben.

Die Wahl des geeigneten grafischen Elements ist kontextabhängig.

- Die Kombination der grafischen Elemente Text und Grafik könnten die Booleschen Werte aussagestark abbilden. Die Kreation eines Symbols könnte eine grapfische Umsetzung darstellen. Das Ergebnis könnte dann folgendermaßen 

aussehen

- Das grafische Element Grafik als alleiniger Akteur kann auch eine gute Alternative sein. Eine Möglichkeit wäre, ein grafisches Symbol mehrfarbig zu gestalten. Dieses Symbol z.B. könnte den Booleschen Wert TRUE reproduzieren. Diese Darstellung z.B. könnte den Booleschen Wert FALSE symbolisieren.

Zur besseren Handhabbarkeit in Generic TV ist eine Parametrisierung der Darstellung gut denkbar.

B.6 – Zeiger

Zum jetzigen Zeitpunkt ist die Anwendung des Datentyps Zeiger nicht geklärt. Weitere Rücksprache mit dem Kunden ist nötig.

Fazit zur Auswahl von grafischen Elementen.

Eindeutige Datentypen wurden definiert. Danach wurde überprüft, welche grafischen Elemente adequat für die Visualsierung der Datentypen wären. In der nächsten Phase werden die Elemente der Eingabemenge E den eingegrenzten Datentypen zugewiesen.

C - Zuweisung auf Datentypen

Das Metamodell bringt eine bestimmte Anzahl von Elementen der Eingabemenge E hervor. Die möglichen Zuweisungen von Datentypen auf diese Elemente[82] wären:

Datentyp	Element
Zeitintervall	**Annotation**
String	**Concept** **Focus** **Value**
Enum	**Concept** **Focus** **Value**
Number	**Concept** **Focus** **Value**
Boolean	**Concept** **Focus** **Value**
Zeiger	**AnnotationRelation**

Tabelle 01 - Abbildung von darstellbaren Datensätzen auf Datentypen

[82] Das Element Track erfordert keine Zuweisung. Es entsteht durch Zuweisungen bzw. Darstellungen der anderen Elemente.

Anhang IV – Testdokumentation

Testcode	Name	Beschreibung	Aufgangszustand	Erwartetes Ergebnis
Einlesen der Inputdaten (EI). Dies sind Tests der Klasse ReadDataInputTest				
EI01	testReadOnlyKnownFile	Überprüft, ob bekannte Dateitypen im System ladbar sind.	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei: 1. Die Datei D1 ist nicht in GenTV.ini eingetragen. 2. Die Datei D2 ist in GenTV.ini eingetragen	Zu 1: System warnt den Benutzer, dass Datei D1 nicht bekannt ist und nicht eingelesen werden kann. Dabei wird kein InternalFrame mit der Darstellung generiert. Zu 2: System lädt ohne Problem die Datei D2
EI02	testReadAllRowsFromFile	Überprüft, ob gesamten Inhalt der ausgewählten Datei vom System eingelesen wird.	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei.	Alle Zeile der ausgewählten Datei werden vom System eingelesen
EI03	testCheckReadedData	Überprüft, ob das Format der Inputdatei korrekt ist.	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei: 1. Die Datei D1 hat ein bekanntes aber korruptes Format 2. Die Datei D2 hat ein korrektes Format	Zu 1: Bei der Datei D1 warnt das System den Benutzer, dass die Datei nicht gelesen werden konnte und generiert keine InternalFrame mit der Darstellung. Zu 2: Die Datei D2 wird vom System ohne Probleme eingelesen.
EI04	testPropertyTypAvailable	Überprüft, ob zu allen vorkommenden Property einen Datentyp in GenTV.ini zugewiesen worden ist.	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei: 1. Zu Property P1 ist kein Datentyp in GenTV.ini zugewiesen worden. 2. Zu allen Propertys sind Datentypen zugewiesen worden.	Zu 1: Der Benutzer wird gewarnt, dass der Datentyp der Property P1 nicht in GenTV.ini eingetragen worden ist und dann wird das System „gekillt“. Zu 2: Das System lädt die Datei ohne Problem
Daten verarbeiten und zur Darstellung vorbereiten (DV)				
DV01	testCreateAllConceptClass	Überprüft, ob aus den Einträgen der Spalte „Type“ einer Inputdatei Konzeptklasse erzeugt werden	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei.	In der Darstellung sind nur die Konzeptklasse vorhanden, die in der Spalte „Type“ der Inputdatei vorkommen
DV02	testCreateAl	Überprüft, ob aus den	Generic TV ist gestartet	In der Darstellung

	lConceptFromAllConceptClass	Einträgen der Spalte „ResourceName“ einer Inputdatei Konzept erzeugt werden. Für jede Zeile einer Inputdatei gehören die jeweiligen Konzepten zu den Konzeptklassen der Spalte „Type“.	und der Benutzer wählt über Dateiauswahlmenü eine Datei.	sind alle Konzept vorhanden, die in der Spalte „ResourceName“ der Inputdatei vorkommen. Außerdem sind sie zu der Konzeptklasse zugeordnet, die in derselben Zeile der Inputdaten in der Spalte „Type“ vorhanden waren.
DV03	testPropertyFromConcept	Überprüft, ob alle Property richtig eingelesen und zugeordnet werden.	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei.	In der Darstellung sind alle in der Inputdatei definierten Propertys vorhanden und zu den richtigen Konzepten zugeordnet.
DV04	testValueFromProperty	Überprüft, ob alle Ausprägungen der Property eingelesen richtig zugewiesen werden.	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei.	In der Darstellung sind alle in der Inputdatei definierten Ausprägungen von Propertys vorhanden und zu den richtigen Propertys zugeordnet.
DV05	testCreateAllAnnotation	Überprüft, ob alle aus den Inputdaten erzeugbaren Annotationen erzeugt werden.	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei.	Alle möglichen Annotationen sind aus den Inputdaten erkannt und erzeugt worden.
DV06	testAllParameterFromAnnotation	Überprüft, ob jede Annotation alle ihre Attribute besitzen: Konzept, Value, Focus	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei.	Alle Annotation sind ordnungsgemäß erzeugt worden
Default- Darstellung der Inputdaten (DD)				
DD01	testCreateAllDefaultTrack	Überprüft, ob alle Default- Spuren (aus jeder Konzeptklasse eine Spur) dargestellt werden	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei. Auf die Frage, ob eine Konstellation geladen werden soll, „nein“ beantworten.	Alle Spure sind ordnungsgemäß dargestellt worden
DD02	testAllAnnotationDisplayed	Überprüft, ob alle genierten Annotationen auch dargestellt werden.	Generic TV ist gestartet und der Benutzer wählt über Dateiauswahlmenü eine Datei.	Alle Annotation sind ordnungsgemäß dargestellt worden

Benutzerinteraktivitäten DeleteTool (BD)				
BD01	testDeleteTrack	Überprüft, ob eine aus dem DeleteTool gewählte Spure aus der Darstellung entfernt wird	Eine Inputdatei ist dargestellt und Benutzer löscht durch das BenutzerTool eine Spure	Gelöschte Spur wird aus der Darstellung entfernt.
BD02	testDeleteAllTrack	Überprüft, ob das DeleteTool alle vorhandene Spure anzeigt in dem es ermöglicht sie alle zu löschen. Dadurch kann man verifizieren, dass die Tools von Generic TV immer alle vorhandene Spuren anzeigen	Eine Inputdatei ist dargestellt und Benutzer löscht durch das BenutzerTool alle Spure	Alle Spure sind aus der Darstellung entfernt.
Benutzeraktivitäten ColorTool (BC)				
BC01	testSetColorWihtRegExpression	Überprüft, ob bestimmte dargestellte Items durch Eingabe einer regulären Expression im ColorTool gefärbt werden können.	Eine Inputdatei ist dargestellt und Benutzer setzt über das ColorTool eine Farbe an bestimmte dargestellte Konzepte. Dafür muss er eine Konzeptklasse A auswählen und eine reguläre Expression eintippen.	Alle dargestellten Items, deren Namen sich aus der eingegebenen regulären Expression ergeben und, die zur Konzeptklasse A gehört werden gefärbt.
BC02	testSetColorToConcept	Überprüft, ob die Farbe von im ColorTool ausgewählten Konzepte gesetzt werden können	Eine Inputdatei ist dargestellt und Benutzer wählt über das ColorTool ein Konzept und weist ihn eine Farbe zu.	Alle Vorkommend des ausgewählten Konzepts werden mit der ausgewählten Farbe neu gefärbt.
Benutzeraktivitäten FilterTool (BF)				
BF01	testFilterDisplay	Überprüft, ob die Darstellung mit vom Benutzer generierten Filterkriterien gefiltert werden kann.	Eine Inputdatei ist dargestellt und Benutzer generiert über FilterTool Filterkriterien und filtert die aktuelle Darstellung	Alle dargestellten Items, die die Filterkriterien erfüllen werden von der Darstellung entfernt.

Benutzeraktivität Tracer (BT)				
BT01	testCreateNewTrack	Überprüft, ob der Benutzer mit der Anwendung des TracerTools neue Spur generieren kann.	Eine Inputdatei ist dargestellt und Benutzer generiert über TracerTool Filterkriterien und filtert die aktuelle Darstellung.	Alle dargestellten Items, die die Filterkriterien erfüllen werden geklont und bilden die Items einer neuen Spur. Dabei werden die ursprünglichen Items aus der Darstellung nicht entfernt.
Benutzeraktivität FormTool (BFT)				
BFT01	testSetNewForm	Überprüft, ob der Benutzer mit der Anwendung des FormTools neue geometrischen Figuren zu bestimmten Konzepte zuweisen kann.	Eine Inputdatei ist dargestellt und Benutzer weist ein Konzept über das FormTool eine neue geometrische Figur zu.	Alle Vorkommend des ausgewählten Konzepts werden mit der neuen geometrischen Figur dargestellt.