Informationsvisualisierung Zusammenfassung

Einfache Diagramme:
Tortendiagramm: Stellt relative Anzahl dar, Teile sind schwer zu vergleichen
Balkendiagramm: Entweder gestapelt (Balken addieren sich) oder nebeneinander
Zeitleisten: Zeitl. Verlauf anhand von Linie(n) dargestellt
Sparklines: Typographische kleine Linien

Metaphern:
Es sollten Metaphern benutzt werden, Worte deren übertragene Aussage gemeint ist.
z.B. Papierkorb, Baum, stark vereinfachte Piktogramme
Bei Piktogrammen wird Menge am besten durch Anzahl, also mehrere Piktogramme dargestellt.

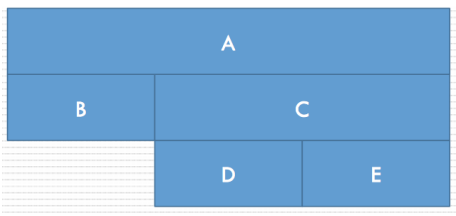
Infografiken:
Bestehen aus Grafiken, Texten und grafischen Diagrammen
Manuell speziell für diesen Fall erstellt
Einfach zu lesen

Wahrnehmung:
Der Mensch kann sich visuelle Informationen leichter merken als andere.
Das Sichtfeld des Menschen ist auf einen Fokuspunkt beschränkt.
Beim Erstellen muss an Farbblindheit gedacht werden.
Helligkeitskontraste besser erkennbar als Farbkontraste
Farbcodierung max 6-12 Farben
Regelmäßige Bewegungen werden ausgeblendet, unerwartete Bewegungen wecken Aufmerksamkeit
Der Mensch neigt dazu Gruppen wahrzunehmen (Naheliegende Punkte, gleiche Form, etc.)
Dreidimensionales Wahrnehmen ist kompliziert

Darstellung hierarchischer Daten:
Node-Link:
- Flexibles Layout, leicht lesbar, einfach verständlich
- Kanten brauchen Platz, schwer zu beschriften, je größer, desto schwieriger

Indented-Outline-Plot:
- Leicht verständlich, leicht zu beschriften, täglich benutzt (Dateiexplorer etc.)
- Innere Kanten brauchen Platz, unflexibel

Icicle Plot:
- Einfach zu beschriften, auch radial möglich, Fläche gut nutzbar
- Weniger intuitiv, Kinder auf Vatergröße beschränkt



TreeMap:
- Fläche der Knoten benutzbar, füllt beliebigen Platz, innere Knoten brauchen keinen Platz, da in Vaterknoten
- Weniger intuitiv, schwer zu beschriften, hierarchische Struktur schwer zu erkennen

Graphvisualisierung

Node-Link:

Codierungen:

- Linienstärke/-farbe: Kantengewichtung
- Pfeilspitzen: Kantenrichtung
- Verschachtelte umgebende Boxen: Hierarchie
- Beschriftung: Knotennamen

Ästhetische Merkmale:

- Kantenkreuzungen minimieren
- Fläche minimieren
- Kantenlänge minimieren
- Knickstellen minimieren, Kanten nicht biegen, nur knicken
- Symmetrie maximieren
- Cluster zusammen anordnen

Wahl des Layouts:

- Hierarchisch, Orthogonal, Radial, Force

Matrizen:

- Layout Problem: NP-Vollständig bei Suche nach optimaler Darstellung

Interaktionstechniken

Information Seeking Mantra:

Overview: Patterns, trends
Zoom: Subset der Daten
Filter: Subset basierend auf Werten
Detail on Demand: Details zur Anzeige auswählen
Relate: Werte vergleichen
History: Aktionen und Sichten im Verlauf verfolgen
Extract: Markieren und extrahieren von Daten

Select: Markieren/Auswählen von Punkten
Explore: Auswahl des Sichtbereichs z.B. per Schwenken
Reconfigure: Neuanordnung der Elemente
Encode: Wechsel zu einer anderen Darstellung
Abstract: Mehr oder weniger Detail anzeigen
Filter: Daten einschränken
Connect: Verbinden von Datenpunkten
Focus und Context: Details im Fokus angezeigt

Multivariante Daten

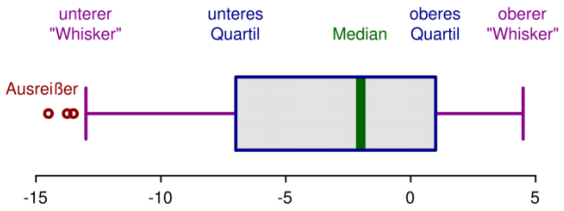
(mehrere Variablen pro Objekt)

Statistische Daten:

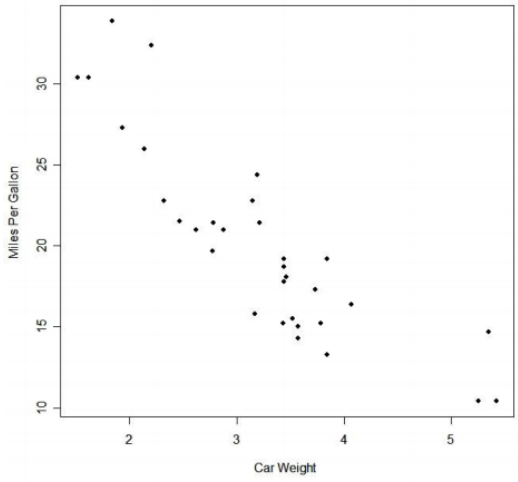
Mittelwert, Median (mittlerer Wert), Quartile (25/75% der Werte), Modus (häufigster Wert), Standardabweichung, Standardfehler

Boxplots:

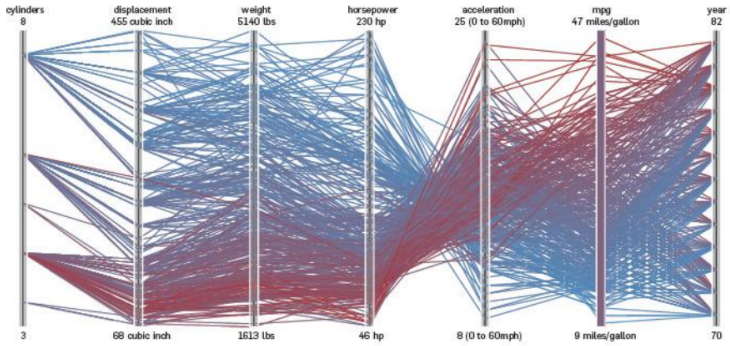
Erweiterung des Balkendiagramms



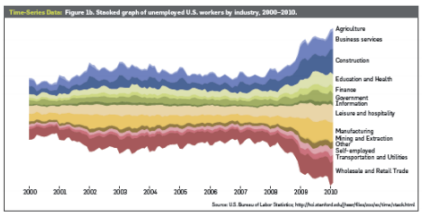
Histogramme:
Balkendiagramm Häufigkeitsverteilung
Dimensionsreduktion:
Zwei Dimensionen in einem Graphen darstellen
Clustering:
Gruppieren von Objekten, Darstellung der ähnlichen Objekte
Tabellen:
Erweiterbar für Kategorien, Zahlen und Sparklines

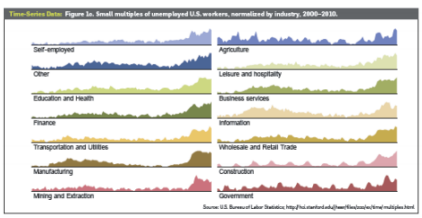


Scatterplot:
Zusammenhang zweier Variablen darstellen.
Scatterplot Matrix:
Mehrere Scatterplots zusammen darstellen.
Parallele Koordinaten:



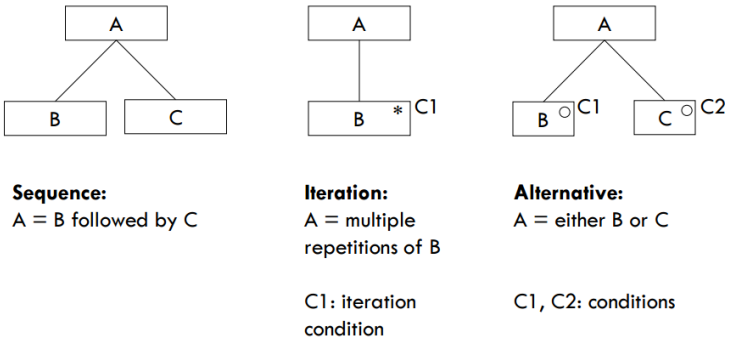
Star Plots:
Radiale Variante der parallelen Koordinaten
Zeitleisten:
Gut animierbar, Kombination mit Hierarchien, Graphen





Softwarevisualisierung

Technische Artefakte: Quelltextfragmente, Datenstrukturen, Arbeitsspeicher
Andere Artefakte: Anforderungen, Software Design, Bug Reports, Codeänderungen
Struktur: Architektur, Abhängigkeiten, Datenstrukturen
Verhalten: Ablauf, Laufzeit, Programmzustand
Evolution: Entwicklungsgeschichte, Änderung am Quelltext
PrettyPrinting, Syntax Highlighting (Gute Zeilenumbrüche etc.)
Jackson Diagrams:



Sequence:
A = B followed by C

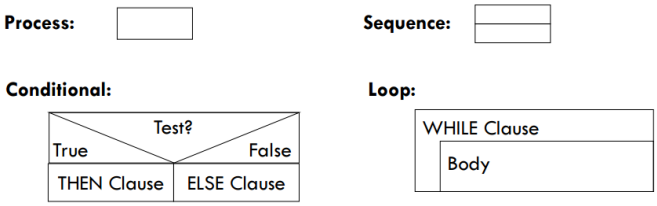
Iteration:
A = multiple repetitions of B

C1: iteration condition

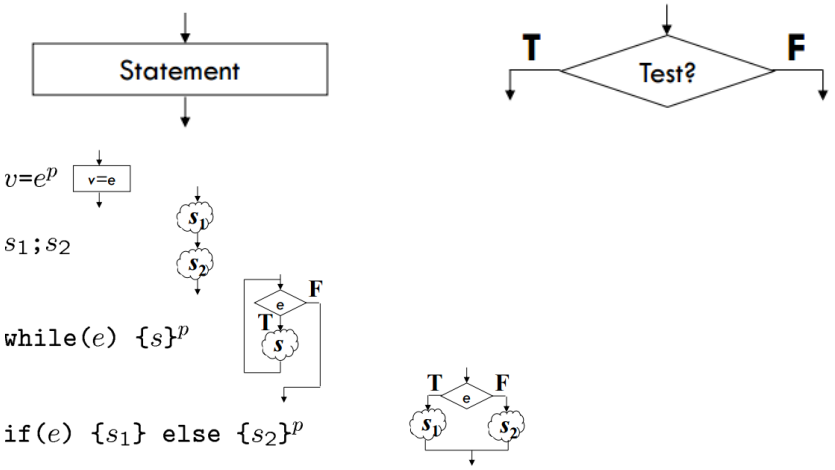
Alternative:
A = either B or C

C1, C2: conditions

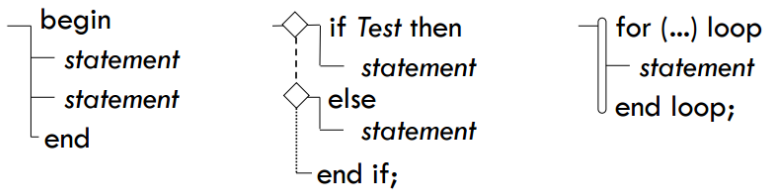
Nassi-Shneiderman Diagrams:



Control Flow Graph:

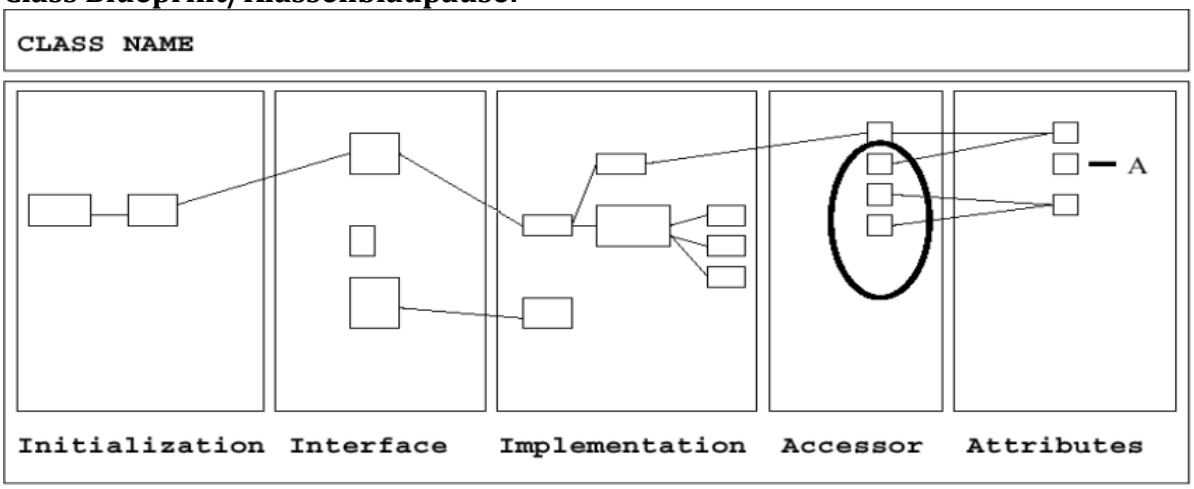


Control Structure Diagrams:



Softwarearchitekturen

Pipes and Filters: Input Stream → Filter → Output
Layered Systems: Zugriff nur von höheren zu niedrigeren Stufen
Blackboard-driven: Verschiedene Einheiten
UML, Node-Link, Matrix können automatisch erstellt werden.
UML Diagramme: Darstellung der Klassen und deren Methoden/Attribute
UML Package Diagramm: Darestellung der Packages und deren Imports einer Software
Class Blueprint/Klassenblaupause:



Dependency Structure Matrix:
Matrix, die die Interaktion der Klassen darstellt

Softwareverhalten

Dynamische Datenerfassung:
- Vor/Nach jeder Instruktion oder bestimmten Instruktionen
- paralleler Thread der den Speicher ausliest
- Debug Hardware
- Erfassung von Programmposition (schwierig im Code anzuzeigen)
- Werte von Variablen

Akkumulative Visualisierung: Durchschnittswerte (DV = Datenvisualisierung)
- Aufrufhäufigkeit eines Programmpunktes (CV = Codevisualisierung)

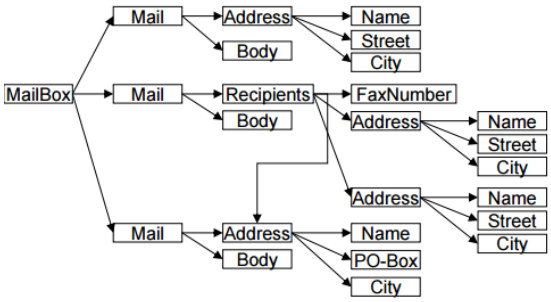
Diagramm mit Zeitachse: Wann wurde Variable geändert (DV)
- Wann wurde Anweisung ausgeführt (CV)

Animation: Folge von Diagrammen des Programmzustandes (DV)
- Jeweils ausgeführte Anweisung anzeigen (CV)

Das Aufzeichnen der Daten hat Einfluss auf das Laufzeitverhalten
Typische Architekturen:
Ad Hoc/Libraries: Ohne spezielle Hilfsmittel, Implementierung im Code
Spezielle Datentypen: Datentypen die über Visualisierung verfügen
Post-Mortem: Aufzeichnung während Laufzeit, Visualisierung danach
Interesting Events: Kennzeichnen interessanter Programmstellen, Senden von Events an Animation

Deklarativ: Trennung von Annotierung und Algorithmus, Demon überwach Zustand des Algorithmus
Semantic-Directed: Automatische Visualisierung durch visuellen Debugger

Visual Debugging:
Debugging: Erkennen, Lokalisieren und Beheben von Fehlern.
Memory Graphs:
Repräsentation des verwendeten Arbeitsspeichers
Knoten = Inhalt, Kanten = Zugriffe
Memory Slices:
Forward Memory Slice (FMS): Von diesem Knoten aus erreichbarer Teilgraph (Vorwärtskanten)
Backward Memory Slice (BMS): Rückwärtskanten
Reference Pattern:
Generalisierung Memory Graph -> Baumstrukturen
Start ist eine Menge von Objekten
Wurzel ist für jede Startmenge die zugehörige Klasse
Kinder: Alle direkt referenzierten oder referenzierenden Objekte als Klasse



Slicing & Dicing:
Static Slice: Menge aller Anweisungen, die eine Variable an einer Programmstelle beeinflussen können
Dynamic Slice: Menge aller Anweisungen, die eine Variable an einer Programmstelle tatsächlich beeinflussen
Execution Slice: Menge aller Anweisungen, die für eine bestimmte Eingabe ausgeführt werden.
Dice: Differenz zweier Slices
Benutzt um z.B. den Unteschied zweier Ausführungen zu zeigen

Softwareevolution

Produktmetriken: Modulgröße, Laufzeit, Tiefe
Prozessmetriken: Anzahl Änderungen, Bugfixes etc.
Arten: Linien (Code Farblich markieren etc.)
Klassenevolution:
Pulsar: Größe pulsiert
Neue Features vs. Refactoring
Supernova: Plötzliche Größenexplosion
White Dwarf: Wird immer kleiner
Vielleicht überflüssig
Evolutionskopplung: Zwei Objekte sind evolutionär gekoppelt, wenn sie zur selben Zeit verändert wurden.