PETER GASSNER
MAD HGKZ 3. SEMESTER 2007

PROF. JÜRGEN SPÄTH
HOCHSCHULE FÜR GESTALTUNG UND KUNST ZÜRICH

PROF. DR. HARALD GALL, SANDRO BOCCUZZO
INSTITUT FÜR INFORMATIK, UNIVERSITÄT ZÜRICH

SOFTWARE VISUALISIERUNG

“

Tausend Bücher à je fünfhundert Seiten mit visuellen Mitteln zugänglich machen

AUFGABENSTELLUNG

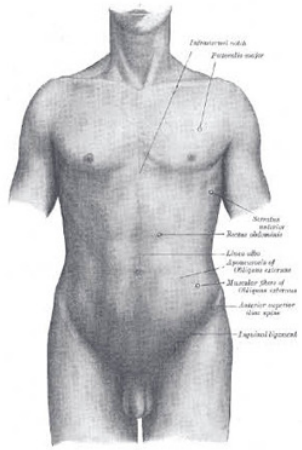
Heutige Computersoftware besteht aus Millionen von Zeilen Programmcode. Es ist daher praktisch unmöglich alle Zusammenhänge in der Software im Überblick zu behalten um etwa Fehler zu entdecken und zu reparieren oder um Personalressourcen geschickt an den richtigen Stellen einzusetzen.

Eine geschickte visuelle Repräsentation des Programmcodes kann helfen, Zusammenhänge qualitativ erfassbar zu machen.

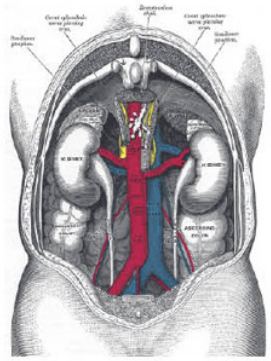
INHALTSVERZEICHNIS

1 SOFTWARE ARCHITEKTUR

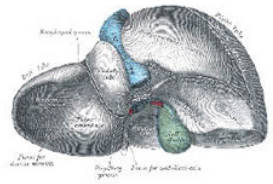
SYSTEM



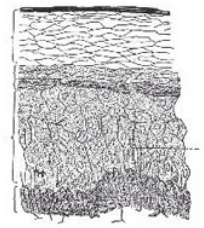
BESTANDTEILE



SUBSYSTEM



MODUL



INSTANZ EINER KLASSE

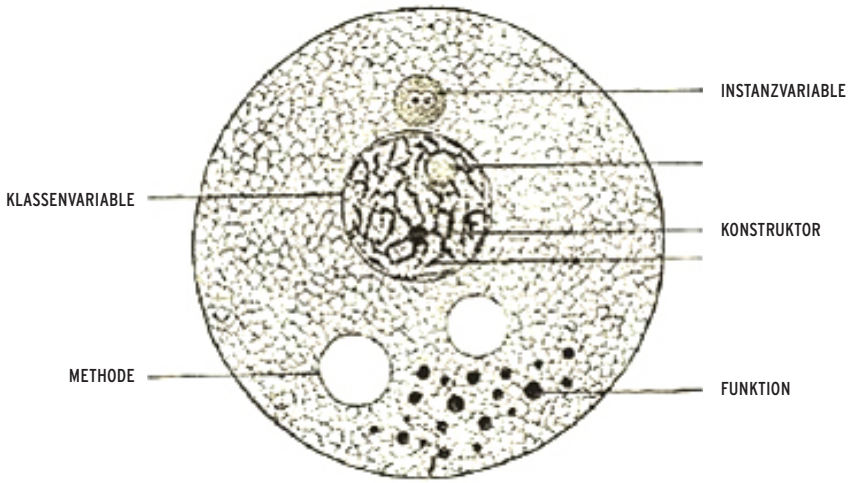
Die Frage nach dem Aufbau von Software haben wir anhand eines Beispiels aus der Biologie versucht zu veranschaulichen. Wir sind in der Diskussion darauf gestossen, dass gewisse Parallelen zwischen Soft- und Wetware bestehen die uns für das Verständnis von Software eine so gute Hilfestellung waren, dass wir diesen Ansatz weiterverfolgt haben.

SYSTEM MENSCH

Wir gehen vom Menschen als System aus. Das System enthält Subsysteme (Programme) wie etwa das «Blutreinigungsprogramm» Leber.

Die Leber ist aus verschiedensten Geweben (Modulen) zusammengesetzt deren Zusammenarbeit essentiell ist: einige produzieren benötigte Stoffe, andere sorgen für die Strukturierung, manche kontrollieren die Aktivität. Diese Gewebe kommunizieren jedoch nur über genau festgelegte Schnittstellen (Interfaces) miteinander. Es können aber auch andere Subsysteme wie der Magen der Leber mitteilen, mehr Gallensaft zu produzieren (Application Programming Interfaces).

Die einzelnen Gewebe bestehen aus Zellen der jeweils gleichen Art, etwa Muskelzellen. Diese Zellen können als Instanzen der Klasse «Muskelzelle» angesehen werden.



ZELLE: INSTANZ EINER KLASSE

Eine Zelle kann als Instanz einer Klasse angeschaut werden. In der DNA (Source-Code-Beschreibung der Klasse) ist festgehalten, wie eine Instanz dieser Klasse auszusehen hat. Wird ein Objekt basierend auf dieser Klasse instanziiert, erhält es sowohl allgemeine, für alle Instanzen dieser Klasse gültige Klassenvariablen, aber auch Instanzvariablen, die nur für dieses Objekt gelten.

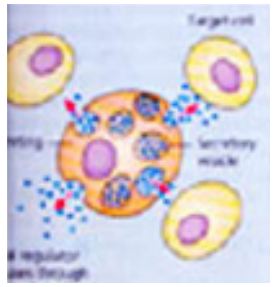
Solche Instanzen kommen meistens in einem Zellverband vor, können aber an ganz verschiedenen Orten im Körper auftreten, je nach dem, wo sie instanziiert werden.

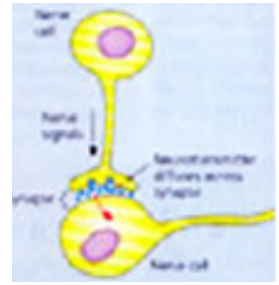
Die Instanz wird auch beeinflusst durch ihre direkte Umgebung (etwa bei der Fleckenverteilung einer Kuh zu beobachten): obwohl sie auf der gleichen Klasse aufbaut, sieht sie anders aus als ihre Nachbarzelle, sie hat andere Instanzvariablen.

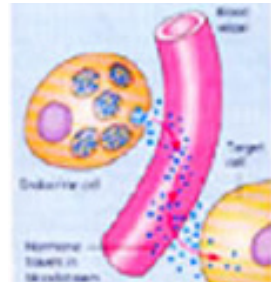
Eine Zelle hat Funktionen, also Programmabläufe, die ohne Argumente ausgeführt werden. Das wären zum Beispiel Enzyme, die Eiweisse abbauen. Eine Zelle hat auch Methoden die mit Argumenten aufgerufen werden können. So kann eine Zelle z.B. über Hormone benachrichtigt werden, den Stoffwechsel anzukurbeln.

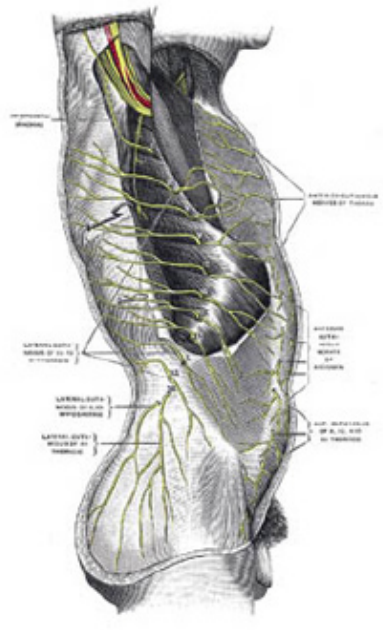
KOMMUNIKATION

Anhand dreier einfacher Beispiele kann auch die Kommunikation, die zwischen Software-Modulen und -Klassen stattfindet, im biologischen System betrachtet werden. Es gibt selbstverständlich noch mehr Kommunikationsmöglichkeiten und viele Feinheiten, es geht hier eher um ein allgemeines Verständnis dafür, wie kommuniziert werden kann.









INTER-ZELLULÄRE KOMMUNIKATION

Zellen, die nah bei einander liegen, können direkt miteinander kommunizieren. Meistens findet eine solche Kommunikation zwischen Zellen der gleichen Art statt.

Ähnliches passiert bei einem Software-Programm: Instanzen von Klassen gleicher Art können gut miteinander kommunizieren, da sie viel Code miteinander teilen und sich daher besser «kennen».

NERVENZELLEN: SCHNELL UND PUNKTUELL

Die Kommunikation über Nervenzellen geht von einem Ursprungsort an einen genau definierten Zielort. Die Geschwindigkeit ist sehr hoch. Es müssen jedoch beide Kommunikationspartner vom jeweils anderen Bescheid wissen. Über eine Nervenzelle kann nur in eine Richtung kommuniziert werden, es gibt also meistens eine Einweg-Kommunikation.

HORMONE: LANGSAM ABER BREIT GEFÄCHERT

Hormone werden über die Blutadern transportiert und gelangen praktisch an jeden Ort im Körper. Die Kommunikationsgeschwindigkeit ist jedoch nicht sehr schnell. Angesprochen werden alle Zellen, die einen Rezeptor für ein bestimmtes Hormon haben. Der Vorteil liegt darin, dass man die Zielgruppe gar nicht genau kennen muss, es genügt zu wissen, dass die, die sich angesprochen fühlen, reagieren werden.

2 ORGANIC INFORMATION DESIGN

DIE ARBEIT VON FRY KANN UNTER FOLGENDER URL ABGERUFEN WERDEN:

http://acg.media.mit.edu/people/fry/thesis/

JEGLICHE REFERENZEN KÖNNEN IN DIESER ARBEIT NACHGESCHLAGEN WERDEN.

Ben Fry vom Media Lab am MIT hat sich in seiner Master-Arbeit mit der Visualisierung von riesigen Datenmengen beschäftigt. Er beschreibt darin auch einige Grundsätze, die ich hier zusammen mitJeremy Stucki zusammengefasst habe.

QUALITATIVE REPRÄSENTATIONEN

Es ist wichtig, die qualitativen Aspekte eines Datensets zu verstehen. Die menschliche Wahrnehmung ist nicht in der Lage, hunderte tausende individuelle Mengen simultan zu verarbeiten. Es wäre auch von kleinem nutzen, da nur ein kleiner Teil der Information überhaupt nützlich ist.

ANSATZ FÜR ORGANISCHE SYSTEME

Betrachtet man Maschinen als Tiere kann man einige interessante Aspekte beobachten: Eine Maschine, die sich von einem entfernt scheint einem nicht zu mögen oder angst vor einem zu haben. Kommt eine Maschine mit hoher Geschwindigkeit auf einen zu, erscheint sie agressiv. Elemente, die sich einander annähern, scheinen sich zu «mögen».

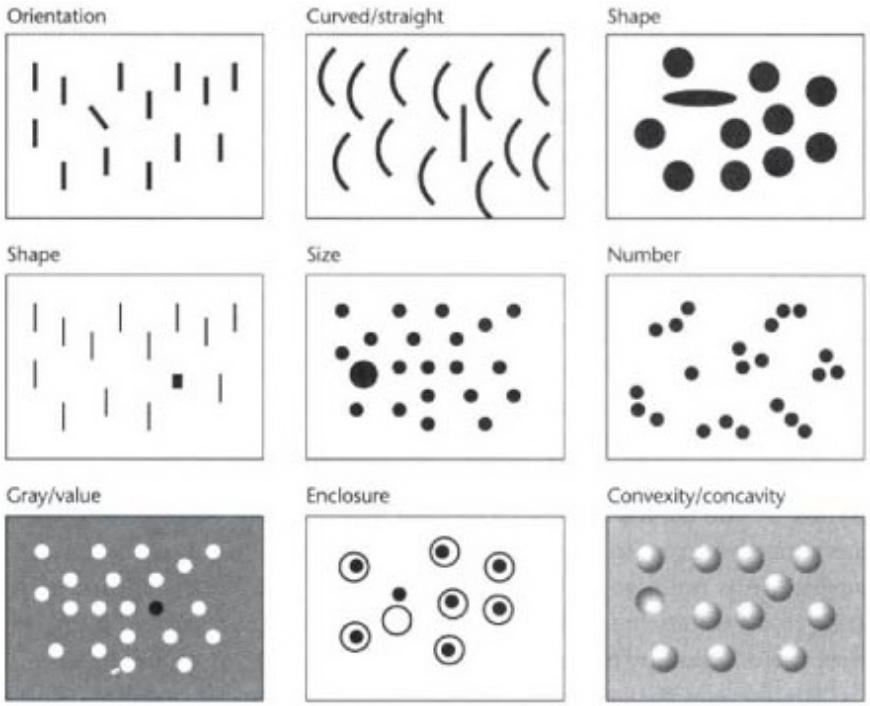
Diese psychologischen Metaphern eignen sich sehr gut zur Konstruktion einer organischen Visualisierung: ein Betrachter kann solche Systeme besser lesen.

DIE VISUALISIERUNG ALS WERKZEUG

Eine organische Informationsvisualisierung bietet einem Betrachter die Möglichkeit, aktiv an der Dekonstruktion des Datensets teilzunehmen. Die Komplexität der Daten kann durch die Echtzeitinteraktion sowie durch Modifizierung der Regeln des Datenbestandes «angefasst» und niedergebrochen werden.

INTERAKTIVE UMGEBUNGEN

Interaktion ist ein essentieller Teil einer Visualisierung, speziell, wenn es sich um die Repräsentation von riesigen Datenstruktu-



«This information is conveyed without any active viewing, meaning that it is ‹pre-attentive.› The term is assigned to objects that are processed faster than 10 milliseconds; as compared to non-preattentive features requiring 40 milliseconds or more.

Triesman, 1988 via Ware, 2000

ren handelt. Das gezielte Ein- und Ausblenden von Informationen ist nur durch Interaktionstechniken möglich. Computer können einen also 1. darin unterstützen, Berechnungen auszuführen und 2. die Resultate interaktiv zu erkunden.

Ein begrenzender Faktor für die Nützlichkeit solcher Repräsentationen ist, dass es sich möglicherweise nur für Dinge eignet, die als einzelne Instanz auf dem Bildschirm angezeigt werden können. Es gibt jedoch viele Daten, die nicht durch einfache Beziehungen zueinander dargestellt werden können.

VISUELLE DARSTELLUNGEN

Eine grosse Zahl von Informationsvisualisierungen scheinen die Wichtigkeit einer klaren und eleganten Erscheinung zu übersehen. Der Weg aus dieser Situation ist, den Gestaltungs- und Programmierprozess in einer geeigneten Umgebung zu unternehmen.

Nicht nur der visuelle Aspekt ist wichtig, sondern auch, wie sich das System verhält. So sind Änderungen, die ohne einen Übergang passieren, sehr schwer nachzuvollziehen.

EIGENSCHAFTEN VON ORGANISCHEN SYSTEMEN

Aus der Betrachtung von lebenden Organismen werden für diese Arbeit neun Eigenschaften abgeleitet, mit denen ein organisches System seine Eigenschaften gegenüber der Umwelt kommunizieren kann.

Es handelt sich dabei um einfache Regeln, mit denen einzelne Individuen miteinander interagieren sollen. Obwohl es sich dabei um sehr einfache Regeln handelt, entsteht aus der Kombination von mehreren solchen Regeln ein ausgeklügeltes, sich selbst organisierendes System, das sich an die sich verändernden Eigenschaften seiner Umgebung anpassen kann.

STRUKTUR

Durch die Ansammlung von vielen Elementen entstehen bedeutungsvolle Strukturen. Mittels Wahrnehmung ihrer Umgebung können Elemente sich autonom mit verwandten Elementen gruppieren.

ERSCHEINUNG

Über sein Erscheinungsbild kann ein Element seinen internen Zustand (oder dessen Veränderung) ausdrücken (z. B. über seine Grösse).

METABOLISMUS

Über ein Regelwerk wird bestimmt, wie verschiedene Daten «verdaut» werden, d. h. welche Auswirkungen sie auf das System haben, ob sie als Bauelemente dienen (z. B. Wörter eines Texts) oder Attribute der Elemente verändern (z. B. wie häufig diese Wörter jeweils vorkommen). Sie können auch wie Enzyme beeinflussen, wie andere Daten das System beeinflussen.

WACHSTUM

Ein System wächst entweder in der Grösse oder in der Anzahl Elemente. Während aktive Teile wachsen, verkümmern andere. Dieser Prozess ist langsam und nicht explizit und sorgt für die Anpassung des Systems an Veränderung. Eine Balance von Wachstum und Verkümmern ist essentiell, da der Darstellungsraum (physisch sowie auch kognitiv) begrenzt ist.

HOMÖOSTASE

Es werden Regeln benötigt, die verhindern, dass Kräfte, die auf Elemente einwirken ein gewisses Mass überschreiten und dass das System «explodiert».

REAKTIONSFÄHIGKEIT

Es gibt drei Arten von Regeln, wie Elemente auf Stimuli reagieren können:

Kompositionsregeln Elemente erkennen, wenn z. B. ein anderes Element zu nahe ist und können entweder ausweichen oder das andere Element zum ausweichen bringen.

Datenregeln Neue Daten (d. h. oft Hinzukommen neuer Elemente) haben Auswirkungen auf existierende Elemente, wie z. B. neue Minimal-/Maximalwerte oder Verschiebung des Durchschnitts.

Interaktionsregeln Elemente reagieren auf Benutzereingaben wie Mausklicks, z. B. ordnen sie sich neu an.

ADAPTION

Das System passt sich langsam den Daten an, die es repräsentiert. Z. B. wenn die gleichen Daten oft vorkommen, «gewöhnt» sich das System daran und eine Wiederholung hat nicht mehr den selben Effekt.

BEWEGUNG

Bewegung ist eines der wichtigsten Mittel für Elemente, Veränderung auszudrücken. Sei es, um ein neues Verhältnis der Elementkomposition darzustellen oder um auf sich aufmerksam zu machen. Auch minimale Bewegung ist dank den Eigenschaften des menschlichen Wahrnehmungssystem sofort und gut erkennbar.

REPRODUKTION

Reproduktion ist im strengen Sinne nicht möglich, da die Elemente durch neue Daten erschaffen werden und sich nicht selbständig fortpflanzen. Jedoch können neue Elemente Eigenschaften von schon bestehenden übernehmen und so im Verhältnis mit ihnen entstehen anstatt für sich alleine.

FAZIT

OIVs sind effektive qualitative Repräsentationen. Sie vermitteln einen guten Überblick in kurzer Zeit und können Tendenzen in einem dynamischen Datenset gut aufzeigen. Hingegen sind sie weniger effektiv für die quantitative Analyse von Daten.

Sie eignen sich gut für erforschbare Visualisierungen, also interaktive Darstellungen.

Sie demonstrieren visuell ihren Inhalt anstatt spezifisch darauf einzugehen. Der Betrachter vergleicht mit vorhergegangenen Zuständen und Bildern und kann so ohne explizite intellektuelle Anstrengung Veränderungen und Charakteristiken erkennen. Dies ist jedoch eher zur Erkennung von grösseren Trends und Zusammenhängen als zur Erforschung von spezifischen und präzisen Details geeignet.

3 RECHERCHE LANDKARTEN

In der Diskussion sind wir auf die Landkarten-Metapher gestossen. Wir haben mittels Google-Earth festgestellt, dass die Navigation vom Weltall bis zur HGKZ ziemlich einfach ist, da man sich an verschiedenen Referenzpunkten in der Landschaft orientieren kann. So sind wir über den italienischen Stiefel zu den Umrissen der Schweiz, dem Zürisee, dem Hauptbahnhof und schlussendlich zur HGKZ gelangt. Da wir uns in der Schweiz auskennen, haben wir das Selbe für ein Gebiet in Kasachstan versucht und stellten auch dort fest, dass die vielen landschaftlichen Merkmale uns die Navigation ermöglichten.

Weiter stellten wir fest, dass, je nach dem was man sucht, das Satellitenbild hilfreicher sein kann als die gezeichnete Karte, da dort sehr viele (man könnte meinen unnötige) Details vorkommen die dem Betrachter helfen, sich zu orientieren.

Es gibt aber einen markanten Unterschied zwischen Geografie und Software: Beständigkeit. Die Landschaft verändert sich verhältnismässig wenig, man begegnet nicht alle paar Wochen einer neuen Situation. Software hingegen kann sich markant verändern und es ist daher schwierig, sie in einem kartografischen System abzubilden. Trotzdem finde ich geografische Merkmale ein mächtiges Werkzeug zur Navigation in einem solch riesigen System und habe darum versucht, Möglichkeiten zu finden, diese darzustellen.

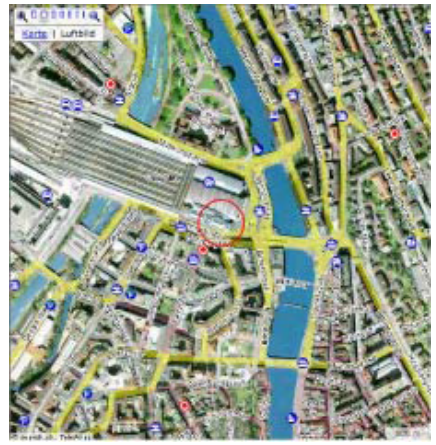
EINZIGARTIGKEIT

Wie das Beispiel der Landkarten zeigt, helfen einem kleinste Details wie etwa die Form und Konstellation von Seen, sich zurecht zu finden. In meinen Modellen versuche ich dieses Hilfsmittel zu verwenden um jede Methode und jede Klasse eigenständig aussehen zu lassen. Es werden dabei sicherlich viele ähnliche Objekte entstehen, aber nur schon durch einige wenige einzigartige Objekte sollte die Navigation erleichtert werden.








DIE PRÄSENTATION VON HANS ROSLING KANN AUF DER TED-SEITE ANGESCHAUT WERDEN:

http://www.ted.com/talks/redirect?key=hans_rosling

«Why should we be interested in visualization? Because the human visual system is a pattern seeker of enormous power and subtlety. The eye and the visual cortex of the brain form a massively parallel processor that provides the highest-bandwidth channel into human cognitive centers. At higher levels of processing, perception and cognition are closely interrelated, which is the reason why the words ‹understanding› and ‹seeing› are synonymous.»

Colin Ware, 2000 via Ben Fry, 2004

ANIMATION

Ein wichtiges Werkzeug ist sicherlich die Animation, die es einem ermöglicht, Übergänge zwischen zwei Stadien zu beobachten. Offensichtlich wird diese Stärke meiner Meinung nach in der Präsentation von Hans Rosling zum Thema Drittwelt-Mythen. Ein Werkzeug wie es Rosling vorstellt ermöglicht einem die Beobachtung eines Objekts über die Zeit, es ist darum keine Frage, dass sich diese Möglichkeit auch anwenden lässt, um Softwareveränderungen zu beobachten. Und gerade hier ist eine solche Möglichkeit wichtig, da sich Software von Release zu Release stark verändert.

PRIORITÄTEN

Ich denke, für eine Visualisierung ist es sehr wichtig Prioritäten zu setzen, die einem ermöglichen in die Visualisierung einzutauchen: Zuerst orientiert man sich am offensichtlichsten Merkmal, kann aber bei genauerem Hinsehen auch weitere Details ausmachen. Auf den Karten von map.search.ch sieht man z. B. sofort die Icons für Bahnhöfe oder Bushaltestellen, kann dann aber ohne weitere Interaktion die Umgebung betrachten.

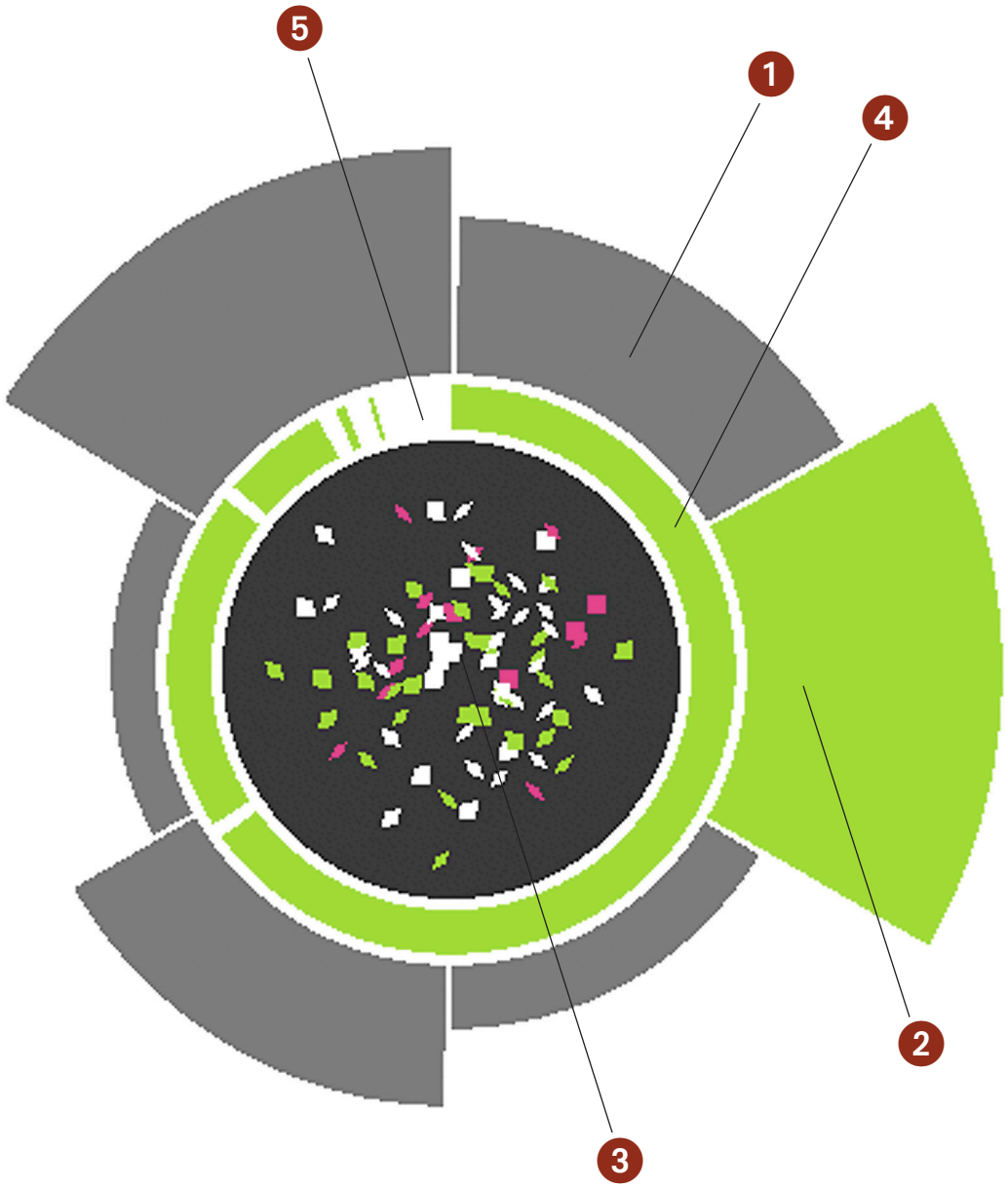
4 UMSETZUNG KONZEPTSKIZZE

Ausgehend von den Recherchen zum Thema Software Kartografie habe ich versucht, eine Grundlage zu schaffen, aus der dann ein Zeichen entwickelt werden kann.

Ich habe auf der Ebene der Methoden begonnen und die Frage gestellt, wie ich eine Methode darstellen kann, damit sie etwas über sich aussagt. Dazu habe ich die vier Parameter «outgoing calls», «incoming calls», «lines of code» und «Fehlerwahrscheinlichkeit» verwendet. Die Schiefe des Zeichens ist so gewählt, dass man stark nach aussen kommunizierende Methoden gut von Methoden mit vielen incoming calls unterscheiden kann: Ausgeglichene Funktionen sind viereckig, was man als ausgewogen lesen kann.

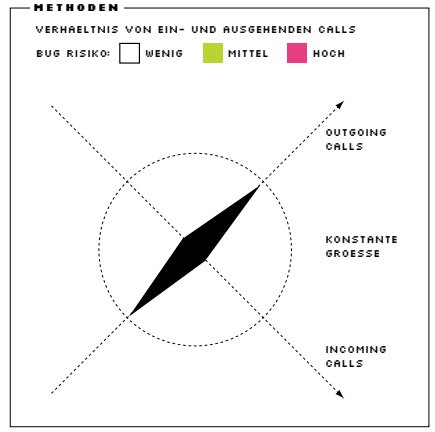
Etwas sehr wichtiges finde ich die Fehlerwahrscheinlichkeit, darum habe ich hierfür die Eigenschaft «Farbe» verwendet. Die Grösse variiere ich nicht, da die Funktionen sehr klein dargestellt werden und ein Grössenunterschied somit schlecht lesbar wird.

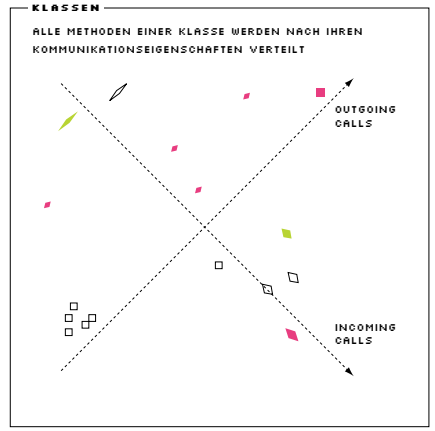
Ausgehend von den Methoden bin ich weiter zur Darstellung einer Klasse übergegangen. Hier habe ich die Parameter der Funktionsaufrufe wiederholt und die einzelnen Funktionen auf der Ebene nach ihrer Kommunikationsmenge angeordnet. Dadurch entsteht für jede Klasse eine ihr eigene Form, ein Fingerabruck.

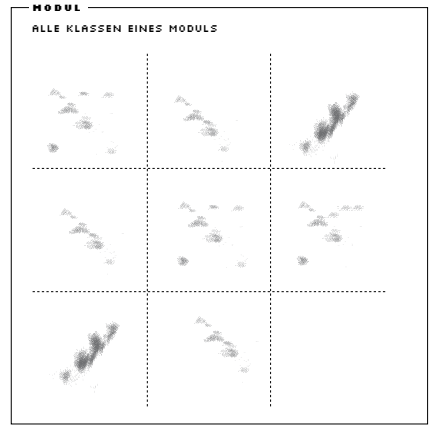


ZEICHEN FÜR EINE KLASSE

1. RELEASES: DIES IST DER ERSTE RELEASE, DANACH GEHT ES IM UHRZEIGERSINN WEITER
2. AUSGEWÄHLTER RELEASE: ALLE PARAMETER PASSEN SICH DIESER VERSION AN
3. METHODEN-FINGERABDRUCK MIT DEM JEWEILIGEN BUG-RISIKO ALS FARBMARKIERUNG
4. ENTWICKLER: DIESER ENTWICKLER HAT CA 70% DES CODES BEIGESTEUERT
5. ENTWICKLER: VIELE ENTWICKLER, DIE WENIG BEIGESTEUERT HABEN







Den entwickelten «Fingerabdruck» wollte ich in diesem zweiten Schritt in ein Zeichen einbauen. Dazu habe ich folgende Prioritäten gesetzt:

1. RELEASES

Ausgehend vom System das Subversion verwendet möchte ich jeweils zu einer Klasse das gesamte Releasespektrum zeigen. Und zwar mit der Menge des geänderten Codes sowie mit den Zeitpunkten, zu denen sich viel getan hat. Dieses Spektrum soll eingegrenzt werden können, so dass z. B. die letzten zehn Releases miteinander verglichen werden können. Dadurch, dass die Releases die äussere Form bilden erkennt man schnell, welche Releases viel geändert werden und welche wenig, welche kürzlich und welche schon lange nicht mehr.

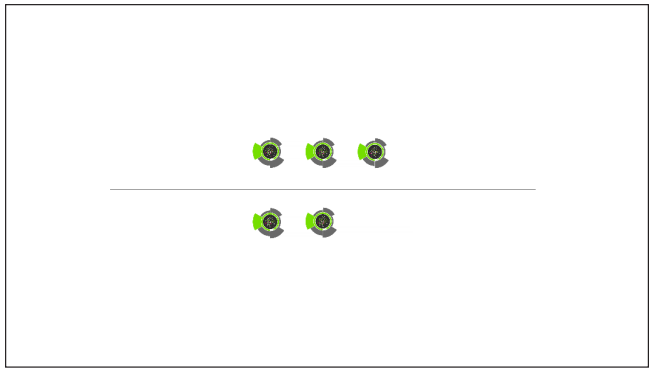
2. METHODEN

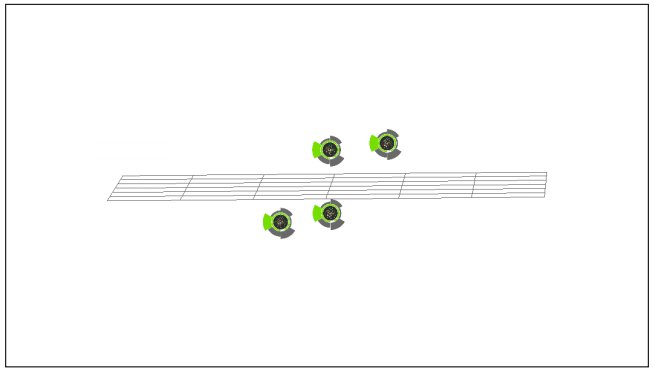
Das charakterisierende Merkmal einer Klasse sind ihre Methoden, darum soll der beschriebene «Fingerabdruck» im Zeichen ersichtlich sein, auch auf kleinster Zoomstufe.

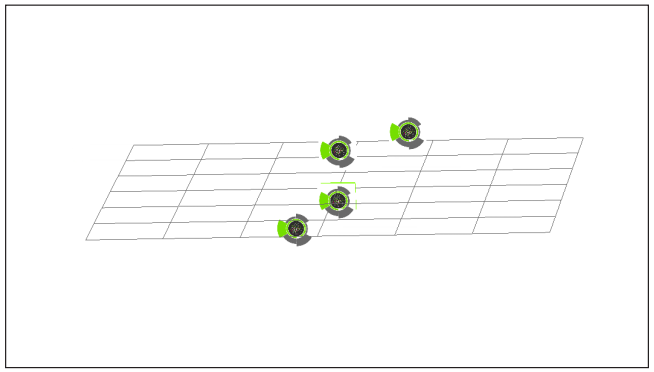
Da ich das Fehlerrisiko durch Farben anzeige kann man aus diesem Methoden-Fingerabdruck auch gleich herauslesen, wie fehleranfällig eine Klasse ist. Die Farbigkeit ist so gewählt, dass auch bei kleiner Darstellung einerseits der Fingerabdruck deutlich wird und andererseits die Fehleranfälligkeit hervorsticht.

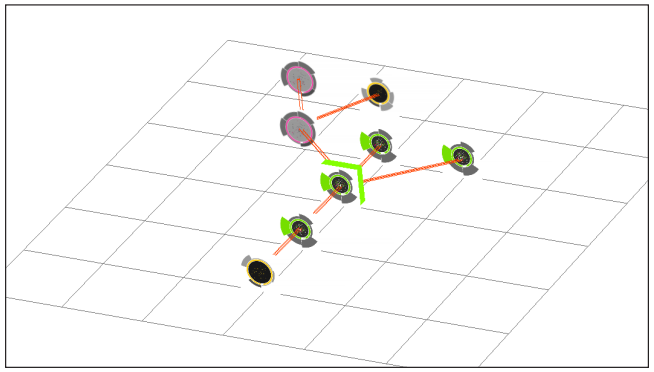
3. ENTWICKLER

Die Anzahl Entwickler sowie deren Codemenge soll ersichtlich sein. Diese sind auf den ersten Blick nicht am wichtigsten, darum erhalten sie einen kleinen Ring. Auch wird dieser Ring ziemlich weiss, wenn es sehr viele Entwickler hat. Das macht jedoch nichts, da es hier mehr darum geht eine Tendenz aufzuzeigen: wie viele Entwickler steuern in welchem Verhältnis Code bei.





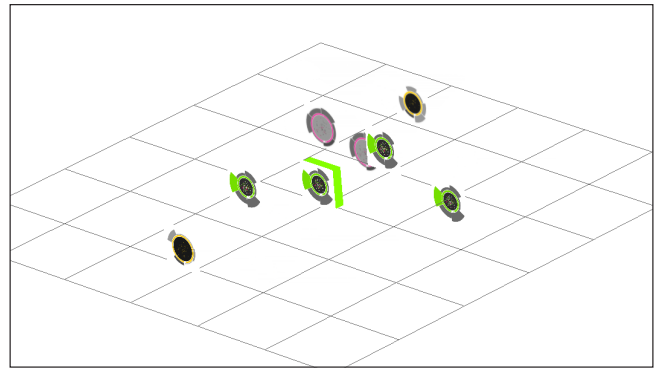


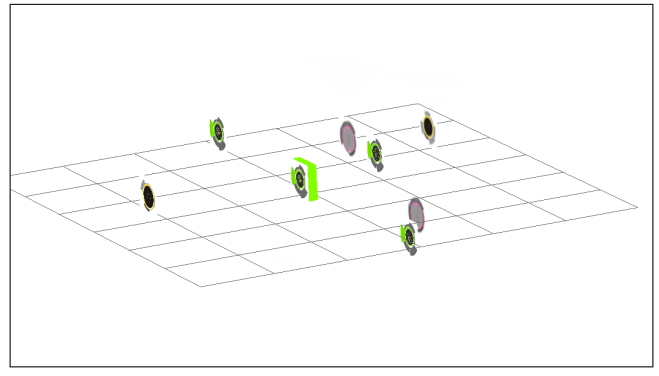


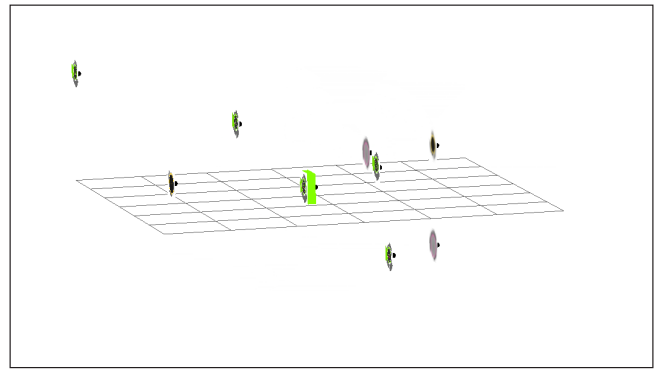
Als Ausgangslage dient die Übersicht über alle Module. In der Abbildung ist ein Modul dargestellt. Dieses Modul enthält fünf Klassen, die alphabetisch sortiert sind. Die Reihenfolge bleibt so über mehrere Versionen hinweg relativ ähnlich, da sich die Klassennamen im Gegensatz zum Quellcode nicht stark verändern.

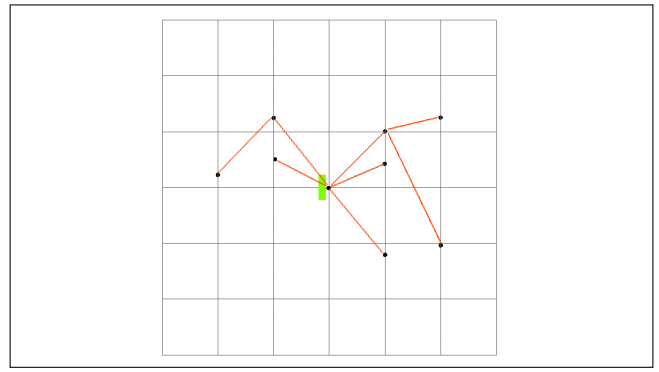
Es kann auf eine isometrische Ansicht umgeschaltet werden, die einem ermöglicht, die Zusammenhänge zwischen einzelnen Klassen zu untersuchen. Dadurch, dass der Übergang von der einen Ansicht zur anderen animiert ist, soll dem Benutzer geholfen werden, den Zusammenhang nicht zu verlieren.

Durch die isometrische Ansicht wird intuitiv lesbar, wie sich die einzelnen Klassen zueinander verhalten.







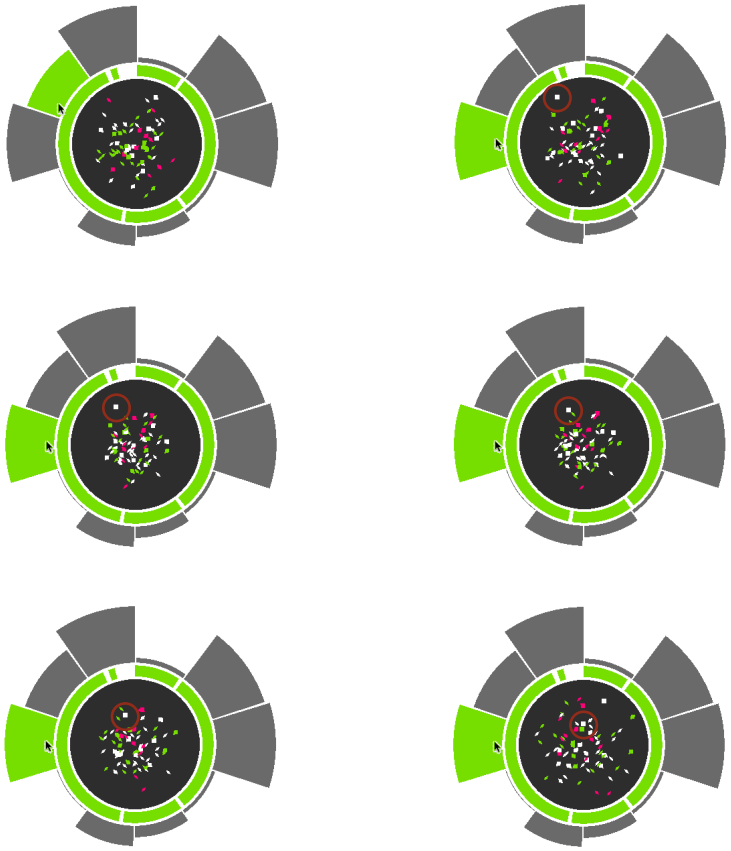


Das System kann auch von der Seite betrachtet werden. Hier werden die Zusammenhänge zwischen Methoden gezeigt.

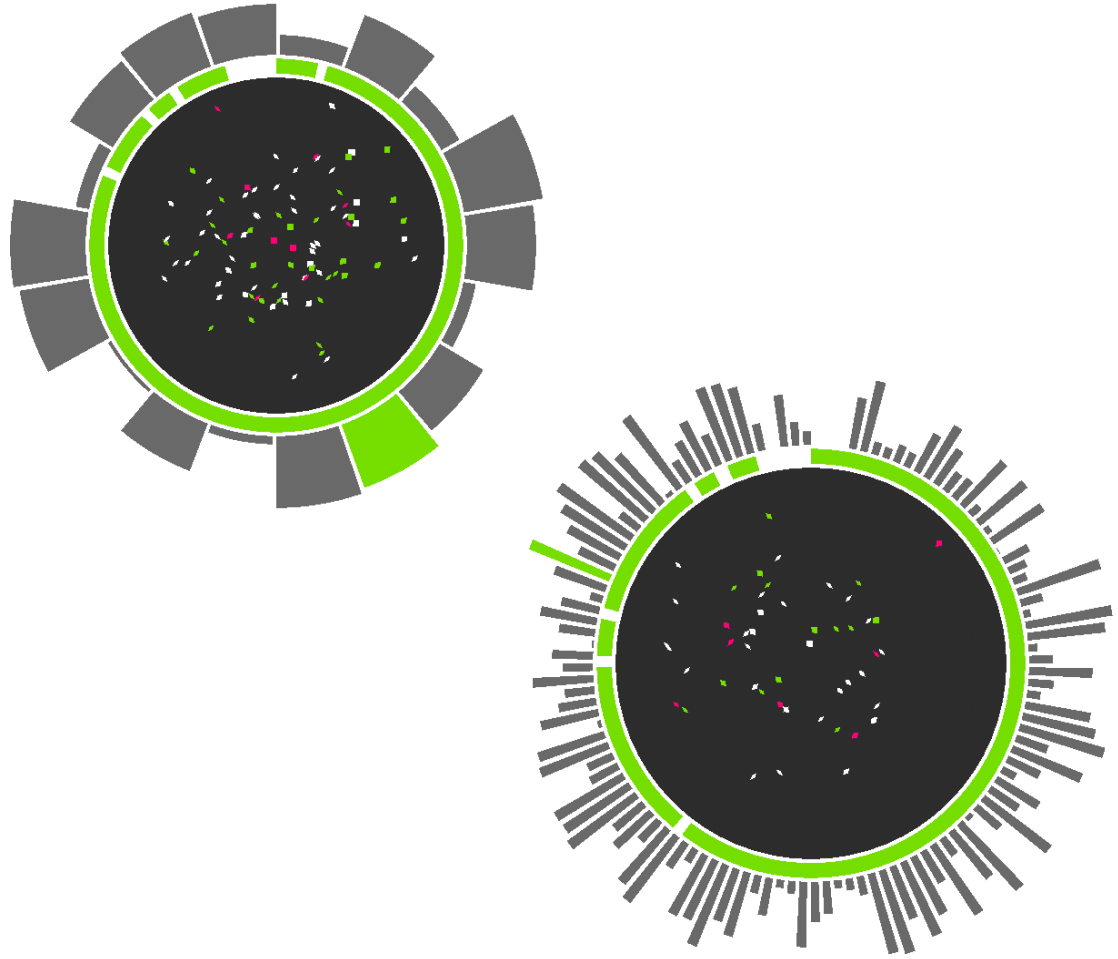
In der Mitte befindet sich die aktuell ausgewählte Methode.

Links von ihr befinden sich Methoden, welche sie aufrufen (incoming calls). Diese aufrufenden Methoden sind in der Höhe nach ihrem Einfluss angeordnet. Je höher, desto einflussreicher ist die aufrufende Methode und desto sorgfältiger muss an der aktuell ausgewählten Methode gearbeitet werden. Denn da diese aufgerufen wird hängen alle Methoden links von ihr von ihrer Funktionalität ab.

Rechts der ausgewählten Methode befinden sich von dieser aufgerufene Methoden. Diese sind in der Höhe nach ihrem «Benutztheitsgrad» angeordnet, d.h. von je mehr Methoden sie aufgerufen werden, desto höher.



Eine Klasse kann während verschiedenen Releases ihrer Entstehung betrachtet werden. Wenn man zwischen zwei Releases umschaltet bewegen sich die Methoden in einer Animation an ihren richtigen Platz. Dadurch kann verfolgt werden, wie sich eine Methode über die Zeit verändert hat.



Hier sind zwei Klassen mit achtzehn und hundert Releases dargestellt. Es ist hier wichtig, einen Filter anwenden zu können, um nur die Releases zu zeigen, die einen interessieren. Ein solcher Filter könnte z. B. «die letzten 10 Releases» oder «alle Hauptreleases im letzten Jahr sein».