Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3563

Visualisierung von dynamischen Software-Entwicklerzahlen in Arbeitsbereichen

Tanja Munz

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Daniel Weiskopf
Betreuer/in:	Dr. Michael Burch
Beginn am:	15. September 2013
Beendet am:	17. März 2014
CR-Nummer:	H.3.3, H.5.2, I.3.3, I.3.6, I.3.8

Kurzfassung

Softwaresysteme werden häufig über mehrere Jahre von vielen Entwicklern bearbeitet. In dieser Zeit können sich sowohl die Beteiligung der Entwickler als auch die Arbeitsbereiche, in denen entwickelt wird, stark verändern. In dieser Arbeit wird das Konzept der AOI Rivers für die Visualisierung von Softwareentwicklungsprozessen angepasst, indem es zu WOI Rivers erweitert wird. Mit WOI Rivers ist es möglich, die dynamischen Verhaltensweisen von Entwicklergruppen zu beobachten. Es kann sichtbar gemacht werden, wie sich die Anzahl von Entwicklern oder die Häufigkeit ihrer Beteiligung an Dateiveränderungen in verschiedenen Arbeitsbereichen über die Zeit verändert. Zusätzlich kann über Transitionen gezeigt werden, wie zwischen verschiedenen Arbeitsbereichen gewechselt wird und wann bzw. wo neue Entwickler hinzukommen oder das Projekt wieder verlassen. Da sich Entwickler zur gleichen Zeit an vielen verschiedenen Dateien bzw. verschiedenen Arbeitsbereichen in unterschiedlichen Stärken beteiligen können, ist die Höhe, die jedem Entwickler in einem Intervall zugewiesen wird, variabel und muss auf mehrere Transitionen aufgeteilt werden. Hierfür werden verschiedene Möglichkeiten untersucht und es wird eine Methode entwickelt, Transitionen zwischen gleichen Arbeitsbereichen nicht unnötig aufzuteilen, um die Anzahl an Überkreuzungen, und dadurch Visual Clutter, zu reduzieren. Die Visualisierungstechnik wurde als interaktives Visualisierungswerkzeug implementiert. In diesem können Arbeitsbereiche, sogenannte Workspaces of Interest (WOIs), in einer Hierarchiedarstellung, Entwicklergruppen aus einer Liste aller Entwickler und der darzustellende Zeitbereich für die WOI River-Visualisierung festgelegt werden. Anhand dreier Open-Source-Softwareprojekte werden Fallstudien durchgeführt, um Einsichten in die Entwicklungsprozesse dieser Projekte zu erhalten.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Softwaresysteme durchlaufen meist einen mehrjährigen Entwicklungsprozess an dem viele Entwickler beteiligt sein können. Im Laufe dieser Zeit werden verschiedene Dateien des Softwaresystems von den verschiedenen Entwicklern erzeugt, verändert oder auch wieder gelöscht. Entwicklergruppen arbeiten dabei in verschiedenen oder auch gemeinsamen Arbeitsbereichen über längere oder kürzere Entwicklungszeiträume. Dabei können sich sowohl die Entwickleranzahl, als auch die Häufigkeit, mit der Änderungen gemacht werden und die Bereiche, die bearbeitet werden, durchgehend verändern.

Um Einsichten in diesen Entwicklungsprozess zu erhalten, können Visualisierungen sehr hilfreich sein. Sie können dabei helfen, Entwicklungsstrategien zu analysieren, sowie Trends, Beziehungen, Anomalien und weitere Besonderheiten während der Entwicklung aufzuspüren. In Bezug auf das Entwicklerverhalten könnte interessieren, wie viele Entwickler über die Zeit an einem Projekt gearbeitet haben, wie sich diese Anzahl oder die Anzahl ihrer Änderungen an dem Projekt über die Zeit verändert hat und wo Schwerpunkte bei der Entwicklung lagen. Zudem könnte von Belang sein, wie viele und welche Entwickler in den einzelnen Bereichen aktiv waren oder wie viele Änderungen es dort gab. Tiefere Einblicke entstehen, wenn erkennbar ist, wie Entwickler zwischen verschiedenen Bereichen wechselten und wann bzw. zu welchen Bereichen neue Entwickler hinzukamen oder nichts mehr zu dem Projekt beigetragen haben. Es gibt noch viele weitere interessante Fragestellungen in Bezug auf den Entwicklungsprozess, dies sind jedoch die wesentlichen Aspekte, die mit Hilfe der hier vorgestellten Technik beantwortet werden können.

Es gibt bereits viele Techniken, die sich mit der grafischen Darstellung eines Softwareentwicklungsprozesses beschäftigen. Hierbei sind statische Verfahren wie die Evolution Storylines [OM10] und Code Flows [TA08] zu nennen. Aber auch dynamische Verfahren wie Gource [Cau10] oder code_swarm [OM09] veranschaulichen, wie Entwickler ein Softwareprojekt über die Zeit verändern.

Die hier vorgestellte Technik ist statisch und basiert auf den AOI Rivers [BKW13]. Diese stellen eine Erweiterung der ThemeRiver [HHWN02] dar, welche – angelehnt an das Konzept der Sankey-Diagramme [Tuf83] – um Transitionen sowie Zu- und Abflüsse erweitert wurden. Mit Hilfe der AOI Rivers kann das zeitliche Verhalten von Augenbewegungen verschiedener Teilnehmer einer Eye-Tracking-Studie dargestellt und analysiert werden. Hierfür werden Bereiche auf dem Bildschirm als Areas of Interest (AOIs) festgelegt, in welche die Augenbewegungen und Fixationspunkte der Teilnehmer während des Experiments fallen können bzw. zwischen denen sie wechseln können.

Im Folgenden wird die Technik der AOI Rivers für die Visualisierung des Softwareentwicklungsprozesses angepasst. Statt Areas of Interest werden Arbeitsbereiche der Projekthierarchie, sog. Workspaces

of Interest (WOIs) betrachtet. Innerhalb dieser WOIs bearbeiten Entwickler die Dateien des Softwaresystems.

Informationen zum Entwicklungsprozess können aus Versionskontrollsystemen gewonnen werden. Mit ihrer Hilfe können Entwickler ihre Änderungen an dem Softwaresystem speichern, indem sie ihre veränderten Dateien in ein Repository „einchecken“ (bzw. „committen“). Dabei werden unter anderem Informationen bezüglich des Änderungsdatums, des Entwicklernamens und der bearbeiteten Dateien protokolliert, aus denen die Systementwicklung rekonstruiert werden kann. Diese Daten können verwendet werden, um Informationen zur Verhaltensweise von Entwicklern zu erhalten und diese als WOI River darzustellen und interaktiv zu untersuchen.

Während bei Eye-Tracking-Experimenten die Teilnehmer jederzeit auf maximal eine Position des Bildschirms schauen können, bearbeiten Entwickler meist mehrere Dateien gleichzeitig in Bezug auf ein festgelegtes Zeitintervall. Folglich werden bei AOI Rivers Teilnehmer in jedem Zeitschritt genau einem AOI zugeordnet und die Transitionen eines Teilnehmers verlaufen zwischen maximal zwei AOIs. Bei den WOI Rivers muss die Höhe der Entwickler in jedem Zeitschritt bzw. Intervall auf verschiedene WOIs aufgeteilt werden und die Transitionen können zwischen all diesen WOIs verlaufen. Außerdem tragen Entwickler in verschiedenen Zeitbereichen und im Vergleich mit anderen Entwicklern unterschiedlich viel zu einem Projekt bei. Hier stellt sich die Frage, wie dies in der Erweiterung der AOI Rivers sinnvoll umzusetzen ist.

1.2 Ziele

Ziel dieser Arbeit ist, ein Visualisierungswerkzeug zur Analyse dynamischer Entwicklerzahlen in Workspaces of Interest (WOIs) eines Softwaresystems zu erstellen. Hierzu soll das Konzept der AOI Rivers entsprechend umfunktioniert und angepasst werden.

Als Eingabe sollen Daten eines Softwaresystems mit dynamischen Entwicklerzahlen verwendet werden, indem sie aus einem Open-Source-Softwaresystem extrahiert und lokal gespeichert werden. Die Systemstruktur des Softwareprojekts soll als interaktiv manipulierbare Hierarchiedarstellung angezeigt werden, in der WOIs interaktiv festgelegt und verändert werden können. Außerdem sollen Entwicklergruppen definiert werden können, die für die Visualisierung verwendet werden. In Abhängigkeit der festgelegten WOIs und Entwicklergruppen soll der entsprechende WOI River als interaktive Visualisierung angezeigt werden. Dabei soll eine Reduzierung der Überkreuzungen in den WOI Rivers sowie ein hoher Farbkontrast benachbarter Ströme angestrebt werden. Des Weiteren sollen benutzerdefinierte Ausschnitte der Daten zur späteren Analyse abgespeichert werden können.

Anhand mehrerer Fallstudien zu verschiedenen Open-Source-Softwaresystemen soll die implementierte Visualisierungstechnik evaluiert werden.

1.3 Gliederung

Die restliche Arbeit ist wie folgt gegliedert:

Kapitel 2 – Verwandte Arbeiten: Dieses Kapitel gibt einen Überblick über einige bereits existierende Visualisierungstechniken in den Bereichen Hierarchievisualisierung, Visualisierung dynamischer quantitativer Daten und Softwarevisualisierung.

Kapitel 3 – Visualisierungstechnik: Die Visualisierungstechnik wird genauer vorgestellt. Hierbei wird erst auf das Datenmodell, danach auf die Hierarchie- und WOI River-Visualisierung eingegangen. Bei letzterer stehen die Erzeugung der Transitionsmatrizen und die Konstruktion der einzelnen Komponenten im Vordergrund.

Kapitel 4 – Implementierung: In diesem Kapitel wird auf die Implementierung eingegangen. Hierzu wird erklärt, wie aus einem Versionskontrollsystem eines Softwareprojekts die relevanten Daten für die Anwendung erhalten werden können und wie aus diesen Daten die entsprechende Visualisierung des WOI Rivers erzeugt werden kann. Dabei werden Funktionalitäten und Interaktionsmöglichkeiten der Anwendung beschrieben.

Kapitel 5 – Fallstudien: Es wird versucht, Einsichten in die Entwicklungsprozesse dreier Open-Source-Softwaresysteme zu erhalten und die Anwendung bzw. deren Visualisierungstechnik wird analysiert. Der erste untersuchte Datensatz repräsentiert den Entwicklungsprozess der Programmiersprache Python. Das zweite untersuchte Projekt ist die Bibliothek libvpx für die Videocodecs VP8/VP9. Außerdem wird der Linux-Kernel als recht großes Projekt betrachtet und es wird geprüft, ob die Visualisierungstechnik auch für größere Datensätze skalierbar ist.

Kapitel 6 – Zusammenfassung und Ausblick: Die Arbeit wird zusammengefasst und es werden Verbesserungs- und Erweiterungsmöglichkeiten für die Anwendung und die Visualisierungstechnik geliefert.

2 Verwandte Arbeiten

In dem hier entwickelten Visualisierungswerkzeug spielen Komponenten aus verschiedenen Bereichen der Informationsvisualisierung eine wichtige Rolle. Allgemein beschäftigt sich die Visualisierung mit *Softwarevisualisierung*. Genauer betrachtet ist die Visualisierung aus zwei Bestandteilen aufgebaut: Zum einen gibt es das Softwaresystem, das durch eine *Hierarchievisualisierung* dargestellt wird; diese Hierarchie enthält alle Dateien, die während des Entwicklungsprozesses erstellt, verändert oder gelöscht wurden. Zum anderen wird in der Hauptvisualisierung ein zeitlicher Verlauf der Entwicklerbeteiligung in Arbeitsbereichen abgebildet. Dies wird durch eine *Visualisierung zeitveränderlicher Daten* in Flussdarstellung realisiert.

Im Folgenden wird eine Übersicht über bereits existierende Visualisierungen in diesen Gebieten geliefert. Zuerst werden die Hierarchievisualisierung und die Visualisierung zeitveränderlicher Daten beschrieben; danach die Softwarevisualisierung als spezielles Anwendungsgebiet, in dem Visualisierungstechniken aus den anderen Bereichen verwenden werden können.

Da in den einzelnen Bereichen jeweils eine große Anzahl an Techniken existiert, werden nicht alle einzeln vorgestellt sondern vor allem Techniken, die für die hier angestrebte Visualisierungstechnik relevant scheinen.

2.1 Hierarchievisualisierung

Es gibt sehr viele verschiedene Techniken zur Visualisierung von Hierarchien. Zu den bekanntesten visuellen Metaphern der Hierarchievisualisierung gehören Node-Link-Diagramme, Layered Icicle Plots [KL83], Indented Plots und Treemaps [JS91]. Es gibt jeweils nochmals mehrere Varianten, die sich in mehreren Aspekten wie Form oder Anordnung unterscheiden und im zwei- oder dreidimensionalen Raum visualisiert werden. Eine weitere, recht neue Technik ist der Verallgemeinerte Pythagorasbaum [BBM+14]. Jürgensmann und Schulz haben eine Übersicht verschiedener Techniken erstellt [JS10]. Jede dieser Techniken besitzt gewisse Vor- und Nachteile und ist daher je nach Anwendungsbereich mehr oder weniger gut geeignet.

Im Folgenden werden die bereits genannten Techniken kurz vorgestellt und es wird auf ihre Eigenschaften eingegangen. In Abbildung 2.1 werden die verschiedenen Techniken anhand einer einfachen Hierarchie, die aus sechs Knoten besteht, veranschaulicht.

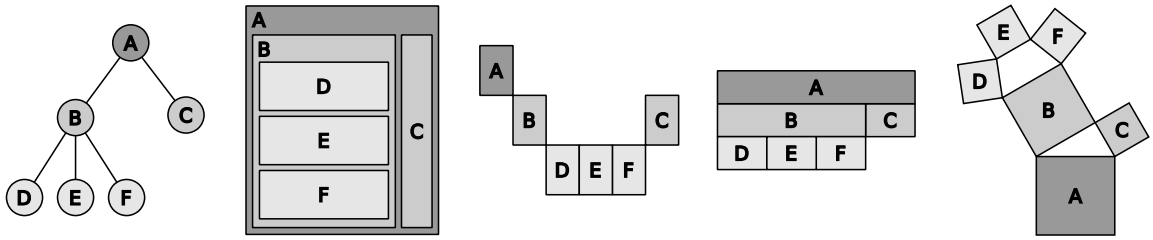


Abbildung 2.1: Veranschaulichung verschiedener Visualisierungstechniken für Hierarchien: (a) Node-Link-Diagramm; (b) Treemap; (c) Indented Plot; (d) Layered Icicle Plot und (e) Verallgemeinerter Pythagorasbaum. (Darstellungen aus [Mun13])

Node-Link-Diagramm Einen klassischen Ansatz zur Visualisierung von Hierarchien stellen Node-Link-Diagramme dar. Knoten werden hierbei häufig durch Kreise dargestellt, die über Linien miteinander verbunden sind. Oft wird der Wurzelknoten oben abgebildet und die Kinderknoten darunter; die Anordnung kann allerdings auch auf verschiedene andere Arten erfolgen. Ein Beispiel für ein Node-Link-Diagramm ist in Abbildung 2.1a zu sehen. Die Struktur von Node-Link-Diagrammen ist einfach erkennbar und einzelne Pfade können gut verfolgt werden. Der zur Verfügung stehende Platz wird allerdings nicht effizient genutzt. Außerdem ist diese Methode eher für kleinere Hierarchien geeignet.

Treemap Bei Treemaps [JS91] wird Verschachtelung zur Visualisierung der Hierarchie verwendet. Knoten werden durch Rechtecke dargestellt, wobei sich Kinderknoten innerhalb ihrer Elternknoten befinden, wie in Abbildung 2.1b zu erkennen ist. Diese Methode nutzt den Platz sehr effizient, allerdings ist die hierarchische Struktur teilweise schwer zu erkennen. Für die Anordnung der Knoten gibt es mehrere verschiedene Unterteilungsalgorithmen. Ursprünglich wurde das Slice-and-Dice-Verfahren verwendet. Andere Ansätze sind z.B. Squarified Treemaps [BHW00], bei denen näherungsweise Quadrate für die Knoten verwendet werden oder kreisförmige Treemaps (Pebbles) [Wet03] und Voronoi-Diagramme [BDL05], die keine Rechtecke zur Darstellung der Knoten verwenden.

Indented Plot Indented Plots verwenden Einrückung zur Positionierung der Kinderknoten. Alle Knoten werden horizontal nebeneinander angeordnet, wobei Kinderknoten im Vergleich zu ihren Elternknoten nach unten eingerückt werden und hierdurch in unterschiedlicher Höhe dargestellt sind. Knoten gleicher Tiefe in der Hierarchie werden in gleicher Höhe in der Visualisierung dargestellt. Ein Nachteil dieser Visualisierungstechnik ist, dass sie horizontal viel Platz benötigt. Ein Beispiel ist in 2.1c zu sehen.
Diese Methode ist aus Dateimanagern bekannt, bei denen die Knoten vertikal angeordnet werden und Unterhierarchien interaktiv ein- und aufgeklappt werden können. In [BRW10] wurden Indented Pixel Tree Plots vorgestellt, die auf diesem Verfahren basieren und auch besonders für große Hierarchien geeignet sind um einen Überblick über die hierarchische Grundstruktur zu liefern.

Layered Icicle Plot Layered Icicle Plots [KL83] entstehen durch Stapelung der Knoten, wobei Kinderknoten auf ihre Elternknoten gesetzt werden, wie in Abbildung 2.1d dargestellt. Jeder der Knoten wird als Rechteck gezeichnet, wobei sich der Wurzelknoten häufig oben befindet; Kinderknoten werden rekursiv darunter gezeichnet. Der Wurzelknoten benötigt hierdurch horizontal genauso viel Platz wie seine Kinderknoten. Eine Abwandlungen dieser Darstellung stellt beispielsweise Sunburst [SZ00] dar, dort wird der Wurzelknoten als Kreis dargestellt und Kinderknoten nach außen in kreisförmigen Schichten aufgesetzt.

Verallgemeinerter Pythagorasbaum Verallgemeinerte Pythagorasbäume [BBM+14] basieren auf der Konstruktionsweise von Pythagorasbäumen [Bos57]. Bei Pythagorasbäumen werden Quadrate über rechtwinklige Dreiecke miteinander verbunden. Dies ermöglicht jedoch nur die Darstellung binärer Hierarchien. In der Erweiterung werden beliebige konvexe Polygone als Verbindungselemente verwendet; die erweiterte Variante kann dadurch zur Darstellung beliebiger Hierarchien verwendet werden, wie in Abbildung 2.1e anhand der Beispielhierarchie zu sehen ist. Diese Technik liefert ästhetische Darstellungen und ist besonders für tiefe Hierarchien geeignet. Nachteile dieser Visualisierungstechnik sind, dass der zur Verfügung stehende Platz nicht effizient genutzt wird und Überlappungen der Knoten auftreten können.

2.2 Visualisierung zeitveränderlicher Daten

Für die Visualisierung zeitveränderlicher Daten gibt es sehr viele verschiedene Methoden. In [AMM+07] kann eine Übersicht verschiedener Techniken gefunden werden. Dort werden verschiedene Kategorien vorgestellt, in welche die unterschiedlichen Techniken eingeordnet werden können. Visualisierungen zeitveränderlicher Daten werden allgemein dazu verwendet, die Entwicklung von Daten über die Zeit zu untersuchen, Trends und Muster zu erkennen und um Einsichten sowie ein tieferes Verständnis für die Daten zu erhalten.

Die in dieser Arbeit verwendete Visualisierungstechnik basiert auf den AOI Rivers [BKW13]; diese Visualisierungstechnik wurde durch Verwendung der Metapher eines Flusses umgesetzt.

Im Folgenden werden nur die Techniken beschrieben, die für diese Arbeit relevant sind; hierbei wird auf ThemeRiver [HHWN02] und AOI Rivers [BKW13] eingegangen. Weitere Techniken, die in diesem Bereich genannt werden sollten, sind Streamgraphs [BW08], CloudLines [KBK11] und StoryFlows [LWW+13]. Jede dieser Techniken verwendet eine Zeitachse durch die eine zeitliche Veränderung der Daten dargestellt wird.

ThemeRiver ThemeRiver [HHWN02] ist eine bekannte Technik zur Darstellung zeitveränderlicher quantitativer Daten. Ursprünglich wurde diese Visualisierungstechnik für die Visualisierung der Veränderung von Themen in Dokumenten über die Zeit verwendet. Es gibt eine Zeitachse, die von links nach rechts verläuft und in Zeitintervalle aufgeteilt ist. Die einzelnen Themen werden durch Teilflüsse in unterschiedlichen Farben dargestellt. In jedem Zeitintervall wird die Häufigkeit der Themen durch die Flusshöhe des jeweiligen Teilflusses dargestellt. Die einzelnen Werte werden über kontinuierliche Kurven miteinander verbunden um abrupte Änderungen zu vermeiden. Zudem werden

alle Teilflüsse aus ästhetischen Gründen symmetrisch um die Mittelachse aufeinander gestapelt. Es ist erkennbar, wie die Wichtigkeit, also die Höhe der einzelnen Teilflüsse, wächst, unverändert bleibt oder abnimmt. Außerdem kann das Verhalten der einzelnen Teilflüsse im Vergleich zueinander betrachtet werden: Entweder beeinflussen sich mehrere Teilflüsse gegenseitig, weisen ein ähnliches Verhalten auf oder stehen in keiner Weise in Beziehung zueinander.

AOI Rivers Die AOI Rivers-Visualisierung [BKW13] bietet die Grundlage für die in dieser Arbeit verwendete Visualisierungstechnik. Sie basiert auf der Grundidee der eben beschriebenen ThemeRivers [HHWN02] und wird für die Visualisierung zeitveränderlicher Daten aus Eye-Tracking-Experimenten eingesetzt.

Als Erweiterung wurde die ThemeRiver-Visualisierung zur Anzeige weiterer Details angepasst, indem sie nach dem Konzept der Sankey-Diagramme [Tuf83] um Transitionen sowie Zu- und Abflüsse erweitert wurde. Mit Hilfe der AOI Rivers ist es möglich, das zeitliche Verhalten von Teilnehmern eines Eye-Tracking-Experiments zu visualisieren und zu untersuchen. Dabei werden auf dem Bildschirm sog. Areas of Interest (AOIs) festgelegt und es wird ermittelt, in welche dieser Bereiche die Augenbewegungen und Fixationspunkte der Teilnehmer fallen. Zunächst erfolgt die Darstellung der Ergebnisse durch die Visualisierung eines ThemeRivers, wodurch erkannt werden kann, welche Bereiche des Bildschirms während des Experiments betrachtet wurden und wie sich die Anzahl der Teilnehmer, die in diese Bereiche blickten, über die Zeit verändert.

Durch Interaktion ist es möglich, eine detailliertere Ansicht zwischen jeweils zwei Zeitpunkten zu erhalten, die Informationen darüber liefert, wie Teilnehmer zwischen verschiedenen Bereichen ihre Augen bewegen und ob sie die als AOIs festgelegten Bereiche des Bildschirms mit ihren Blicken verlassen bzw. ob sie wieder in die Bereiche schauen. Hiermit können Strategien der Teilnehmer besser untersucht werden.

Während bei den AOI Rivers Eye-Tracking-Daten aus einer Benutzerstudie analysiert wurden, sollen im Folgenden Entwicklerzahlen eines Softwaresystems untersucht werden können.

2.3 Softwarevisualisierung

Die Bereiche in denen Softwarevisualisierung eingesetzt werden kann variieren sehr stark. In der Beschreibung von Softwarevisualisierung geht Stephan Diehl in seinem Buch [Die07] auf drei Unterteilungsbereiche ein: *Struktur* (structure), *Verhalten* (behaviour) und *Entwicklung* (evolution). Unter der Struktur ordnet er statische Elemente und Beziehungen eines Systems ein. Dazu zählen unter anderem der Programm-Code, Datenstrukturen und statische Call-Graphen. Beim Verhalten bezieht er sich auf die Ausführung des Programms und bei der Entwicklung werden der Entwicklungsprozess des Softwaresystems und Codeveränderungen über die Zeit betrachtet.

Die hier entwickelte Visualisierungstechnik kann in den Bereich der *Entwicklung* eingeordnet werden, da sie den zeitlichen Verlauf der Entwicklung, aber nicht die Software selbst, visualisiert. In diesem Gebiet kann die Analyse verschiedene Schwerpunkte haben. Es kann der Code berücksichtigt werden, wie die einzelnen Zeilen über die Zeit verändert werden oder es können einzelne Dateien

oder Klassen betrachtet werden. Es gibt sowohl statische als auch dynamische Verfahren, im zweidimensionalen und dreidimensionalen Bereich. Einen groben Überblick über verschiedene Techniken der Softwarevisualisierung gibt es beispielsweise in [Die07] und [SČG05].

Im Folgenden wird auf einige dieser Techniken genauer eingegangen. Hierbei stehen Verfahren im Vordergrund, bei denen das Verhalten von Entwicklern betrachtet wird. Zuerst werden die animierte Techniken Gource [Cau10] und code_swarm [OM09] vorgestellt, danach die statischen Verfahren Software Evolution Storyline [OM10] und Code Flows [TA08].

2.3.1 Animierte Softwarevisualisierung

Animierte Visualisierungen eines Softwareentwicklungsprozesses sind meist sehr ansprechend gestaltet. Sie zeigen wie das System dynamisch bearbeitet und vergrößert wird. Man sieht wie neue Dateien erstellt werden, wie die Dateien ihre Größe oder Wichtigkeit verändern, wie Entwickler zum Geschehen hinzukommen, in den Vordergrund treten oder wieder verschwinden. Sie lassen an dem Entwicklungsprozess in gewisser Weise teilhaben und ihn gebannt verfolgen. Allerdings ist es schwer, genaue Details festzuhalten und die Animation muss unter Umständen mehrere Minuten lang aufmerksam verfolgt werden.

code_swarm Mit code_swarm [OM09] kann ein Softwareentwicklungsprozess animiert dargestellt werden. Hierbei werden die einzelnen Entwickler und die Dateien, die über die Zeit committet werden, berücksichtigt. Die Dateien werden durch Kreise dargestellt, die Entwickler durch Namenslabel. Bei jedem Commit werden der Entwickler und die entsprechenden veränderten Dateien hervorgehoben und die veränderten Dateien bewegen sich auf den Entwickler zu. Die Farben der Dateien deuten den Dateityp an; ihre Helligkeit, wann die Datei das letzte Mal verändert wurde; die Größe der Datei-Repräsentationen ist in Abhängigkeit der Anzahl committeter Dateien. Wenn Dateien oder Entwickler über längere Zeit inaktiv sind, verschwinden sie allmählich wieder aus der Visualisierung. Zusätzlich wird ein Histogramm für die Anzahl und Art der Commits angezeigt. Durch diese Visualisierung entsteht eine gute Übersicht über die Entwicklung.

Gource Auch Gource [Cau10] ist ein animiertes Visualisierungswerkzeug, das zusätzlich interaktive Features bietet. Die einzelnen Entwickler werden zusammen mit einer Projekthierarchie, die sie dynamisch aufbauen und verändern, visualisiert. Einzelne Dateien werden durch Kugeln visualisiert und in Abhängigkeit ihres Dateityps eingefärbt. Wenn ein Entwickler eine Datei erzeugt, verändert oder löscht, bewegt er sich auf sie zu und sendet einen eingefärbten Strahl in Richtung der Datei aus, der die Art der Änderung andeutet. Verzeichnisse werden durch Splines miteinander verbunden und Dateien werden in spiralförmigem Muster um das Zentrum des zugehörigen Verzeichnisses angeordnet. Für die Anordnung der aktiven Verzeichnisse der Projekthierarchie wird ein Force-Directed-Layoutalgorithmus verwendet.

2.3.2 Statische Softwarevisualisierung

Bei statischen Visualisierungen fällt es meist leichter, Details über den Softwareentwicklungsprozess zu erhalten. In nur einer Abbildung erhält man eine Übersicht über den gesamten Entwicklungsprozess. Für den dynamischen Aspekt wird häufig eine Zeitachse verwendet,

Software Evolution Storylines Software Evolution Storylines [OM10] veranschaulichen die Zusammenarbeit von Entwicklern in der Softwareprojektentwicklung über die Zeit. Die Zeit verläuft in Zeitschritten von links nach rechts. Einzelne Entwickler werden in Form von Röhren (Tubes) unterschiedlicher Farbe dargestellt und durch einen Clustering-Algorithmus in Abhängigkeit der bearbeiteten Dateien in der Nähe anderer Entwickler platziert. Entwickler die zusammenarbeiten werden demnach näher beieinander gezeichnet. Zusätzlich wird durch ein Histogramm die Menge und Art der committeten Dateien in den einzelnen Zeitschritten angezeigt. Bei dieser Visualisierungstechnik entstehen ästhetische Bilder und sie zeigt mehr Details als animierte Techniken. Der zeitliche Verlauf ist in einer einzigen Visualisierung zu sehen, welche weiter untersucht werden kann. Ein Nachteil ist, dass die Technik für große Projekte nicht skaliert, Visual Clutter entsteht und viele Entwickler in das gleiche Cluster fallen können.

Code Flows Durch Code Flows [TA08] kann die zeitliche Entwicklung der Struktur von Programmcode genauer untersucht werden. Hierbei können Änderungen wie das Verschieben, Teilen, Zusammenfügen, Einfügen oder Löschen von Code genauer betrachtet werden. Für die Visualisierung werden gespiegelte Layered Icicle Plots verwendet, welche Codeblöcke aufeinanderfolgender Versionen repräsentieren. Diese werden durch Splines zur Darstellung von Beziehungen miteinander verbunden. Dabei wird unterschieden, ob Codefragmente während der Entwicklung fast unverändert bleiben oder ob sie häufig verändert werden.

3 Visualisierungstechnik

In diesem Kapitel wird die verwendete Visualisierungstechnik für die Systemstruktur und die WOI River-Visualisierung vorgestellt. Zuerst wird auf das verwendete Datenmodell eingegangen, danach werden die Visualisierungstechniken detailliert beschrieben.

Als Grundlage werden Daten von Softwaresystemen verwendet. Hierbei sind die einzelnen Commits von Bedeutung, diese sollten folgende Informationen enthalten:

einen **Zeitstempel**, der das Änderungsdatum angibt,

einen **Entwicklernamen**, der im Repository für den Entwickler festgelegt wurde und

veränderte Dateien, also alle Dateien, die erzeugt, gelöscht oder bearbeitet wurden.

Aus diesen Daten kann die Projekthierarchie ermittelt und visualisiert werden. Durch die Festlegung von WOIs innerhalb der Projekthierarchie, eines Zeitbereichs, der Art der Intervalle und durch die Auswahl (mehrerer) Entwickler kann der zugehörige WOI River berechnet und visualisiert werden.

3.1 Datenmodell

3.1.1 Commit

Ein Commit $c \in C$ ist ein 3-Tupel $c = (t,\ d,\ f)$, bestehend aus einem **Zeitstempel** t, einem **Entwicklernamen** $d \in D$ und einer Menge **veränderter Dateien** $f \in \mathcal{P}(F)$.
Hierbei ist C die Menge aller Commits, D die Menge aller Entwickler und F die Menge aller Dateien, die in den Commits vorkommen.

3.1.2 Geschichte

Die gesamte Geschichte eines Softwareprojekts wird als Sequenz aller Commits modelliert:
$H = c_1 \to c_2 \to \cdots \to c_n$ mit $c_i \in C$

3.1.3 Hierarchie

Die Hierarchie wird als gerichteter Graph $H = (V, E)$ modelliert, wobei $V = \{v_1, \ldots, v_k\}$ die Menge von k Knoten und $E \subset V \times V$ die Menge der Kanten repräsentiert. Es kann beliebig viele Wurzelknoten geben (die keine eingehenden Kanten besitzen); die restlichen Knoten besitzen einen Eingangsgrad von eins. Der Ausgangsgrad aller Knoten ist beliebig. Blattknoten sind die Knoten der Hierarchie, die keine ausgehenden Kanten besitzen.

Eine Teilhierarchie $h = (V_h, E_h)$ mit $V_h \subseteq V$ und $E_h \subseteq E$ wird entsprechend modelliert, es gibt allerdings jeweils nur einen Wurzelknoten. Zwei Teilhierarchien sind disjunkt, wenn sie keine gemeinsamen Knoten besitzen.

Jede Datei in F wird durch einen Dateipfad dargestellt, dieser besteht aus einer Sequenz von Knoten: $f_i = v_{i_1} \to v_{i_2} \to \cdots \to v_{i_m}$. Die Projekthierarchie wird aus allen vorkommenden Knoten in den Dateipfaden aller Commits gebildet.

3.1.4 WOI

Ein WOI ist eine Menge aus Teilhierarchien der Hierarchie H:
$W = \{h_1, h_2 \ldots, h_p\}$

Alle WOIs ergeben eine Menge an l WOIs:
$\mathbb{W} = \{W_1, \ldots, W_l\}$

Hierbei dürfen sich zwei WOIs nicht überlappen, d.h. WOIs dürfen keine gemeinsamen Knoten enthalten.

3.1.5 Entwicklergruppe

Eine Entwicklergruppe ist eine Menge mehrerer Entwickler:
$G \subseteq D$

Es können mehrere Entwicklergruppen festgelegt werden, deren Vereinigung $\mathbb{G} = G_1 \cup G_2 \cup \cdots \cup G_q$ alle Entwickler der ausgewählten Gruppen beinhaltet.

3.1.6 Zeitbereich und Intervalle

Für die WOI River-Visualisierung kann ein beliebiger Zeitbereich $[t_{min}, t_{max}]$ angegeben werden.

Falls genau die Zeitspanne, in der alle Commits liegen, dargestellt werden soll, müssen der minimale und maximale Zeitstempel aller Commits verwendet werden:
$t_{min} = \min(t_1, t_2, \ldots, t_n)$,
$t_{max} = \max(t_1, t_2, \ldots, t_n)$.

Intervalle können auf zwei verschiedene Arten gebildet werden – entweder in Abhängigkeit der Zeit oder der Commits.

Zeitintervalle

Für die Aufteilung des Zeitbereiches $[t_{min},\ t_{max}]$ in eine festgelegte Anzahl von I Intervallen erhält jedes Intervall eine Länge $t = \lfloor \frac{t_{max} - t_{min}}{I} \rfloor$.

Sollen Zeitintervalle einer festen Länge t gebildet werden, ergeben sich $I = \lceil \frac{t_{max} - t_{min}}{t} \rceil$ Intervalle.

Commit-Intervalle

Wenn x Commits in ein Intervall fallen sollen, wird die Gesamtzahl i aller Commits, die im angegebenen Zeitbereich liegen, in $I = \lceil \frac{i}{x} \rceil$ Intervalle aufgeteilt bzw. bei I Intervallen befinden sich $x = \lfloor \frac{i}{I} \rfloor$ Commits in jedem Zeitintervall.

3.1.7 Transitionsmatrizen

Aus den festgelegten WOIs $\mathbb{W}$, dem Zeitbereich $[t_{min},\ t_{max}]$, der Anzahl an Intervallen I und den Entwicklern in den ausgewählten Entwicklergruppen $\mathbb{G}$ können zeitveränderliche Transitionsmatrizen

$M_i \in M(l + 1 \times l + 1, \mathbb{R} \times \mathbb{R})$

berechnet werden. Hierbei wird ein zusätzlicher WOI eingeführt, der für Zu- und Abflüsse zuständig ist.

Die genaue Berechnung der Transitionsmatrizen wird in Abschnitt 3.3 schrittweise hergeleitet.

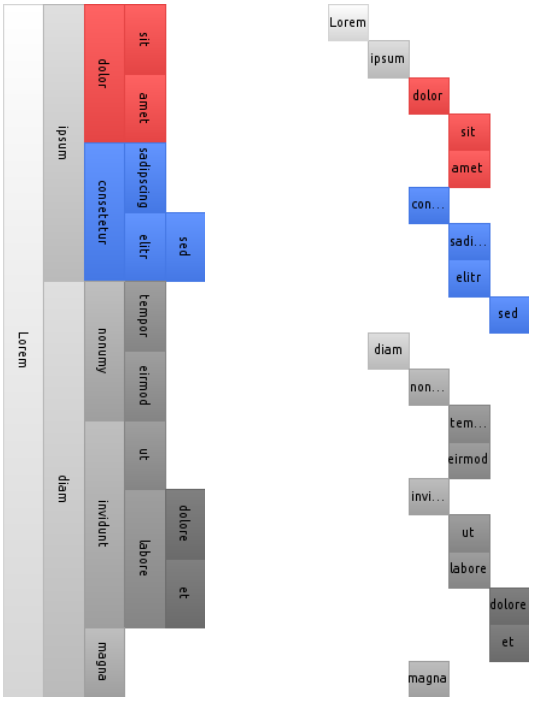
3.2 Hierarchie

Die Hierarchievisualisierung repräsentiert die Systemstruktur des Softwareprojekts; in ihr werden die ausgewählten WOIs farblich hervorgehoben. Dies bietet eine Zuordnung der Arbeitsbereiche zu den Flüssen des WOI Rivers.

Konstruktion

Die Projekthierarchie kann sowohl als Layered Icicle Plot als auch als Indented Plot visualisiert werden. Beide visuelle Metaphern können neben der WOI River-Visualisierung platziert werden und benötigen horizontal wenig Platz. Dabei können die Hierarchiedarstellungen in x- und y-Richtung gestaucht werden, ohne dass die Grundstruktur verloren geht. Sie eignen sich auch für größere Hierarchien, wobei die Struktur immer noch gut erkennbar ist. In Abbildung 3.1 sind beide Varianten anhand einer Beispielhierarchie dargestellt.

Die Wurzelknoten befinden sich links, die Kinderknoten jeweils rechts von ihren Elternknoten. Die Hierarchie wächst somit von links nach rechts.



(a) **(b)**

Abbildung 3.1: Hierarchievisualisierung als (a) Layered Icicle Plot und (b) Indented Plot.

In beiden Fällen werden Knoten in Form von Rechtecken dargestellt, die jeweils eine gleiche Breite besitzen.

Layered Icicle Plot Bei Layered Icicle Plots werden Kinderknoten rekursiv auf ihre Elternknoten gesetzt. Dabei werden alle Blattknoten in der gleichen Größe dargestellt. Die Höhe bzw. der Flächeninhalt eines Elternknotens wird aus der Summe der Höhe bzw. der Flächeninhalte seiner Kinderknoten ermittelt.

Indented Plot Bei Indented Plots sind Kinderknoten im Vergleich zu ihren Elternknoten nach rechts eingerückt, alle Knoten gleicher Tiefe werden bei der gleichen x-Koordinate platziert. Hier werden alle Knoten durch gleich große Rechtecke dargestellt.

Die Bestimmung der Höhe könnte auch in Abhängigkeit der Dateigröße oder der bearbeiteten Dateien erfolgen. Da im Folgenden nur die Systemstruktur interessiert und der Schwerpunkt auf der Visualisierung des WOI Rivers liegt, wird die eben beschriebene Festlegung für die Höhe verwendet.

Farben

Wenn keine WOIs ausgewählt sind, wird die Hierarchie in Grautönen eingefärbt. Der verwendete Grauton spiegelt die Tiefe eines Knotens innerhalb der Hierarchie wider. Alle Knoten erhalten

zusätzlich einen Farbverlauf von einem helleren zu einem dunkleren Grauton, um die Abbildung ästhetischer wirken zu lassen.

Alle Knoten, die zu einem WOI gehören, werden entsprechend der zugeordneten WOI-Farbe eingefärbt. Hierbei wird ein Farbverlauf verwendet, sodass die Farbe des WOIs in der Mitte des Knotens sichtbar ist; nach oben wird sie etwas heller, nach unten dunkler. In diesem Fall gibt es keine zusätzliche Farbänderung für die Tiefe, um keine Verwirrung mit zusätzlichen Farbtönen zu erzeugen.

Labels

Jeder Knoten erhält ein Label des Datei- bzw. Verzeichnisnamens. Je nachdem ob das Rechteck, das den Knoten repräsentiert, breiter oder höher ist, wird das Label horizontal bzw. vertikal gezeichnet. Falls nicht genügend Platz für die gesamte Zeichenkette zur Verfügung steht werden ggf. Ellipsen verwendet.

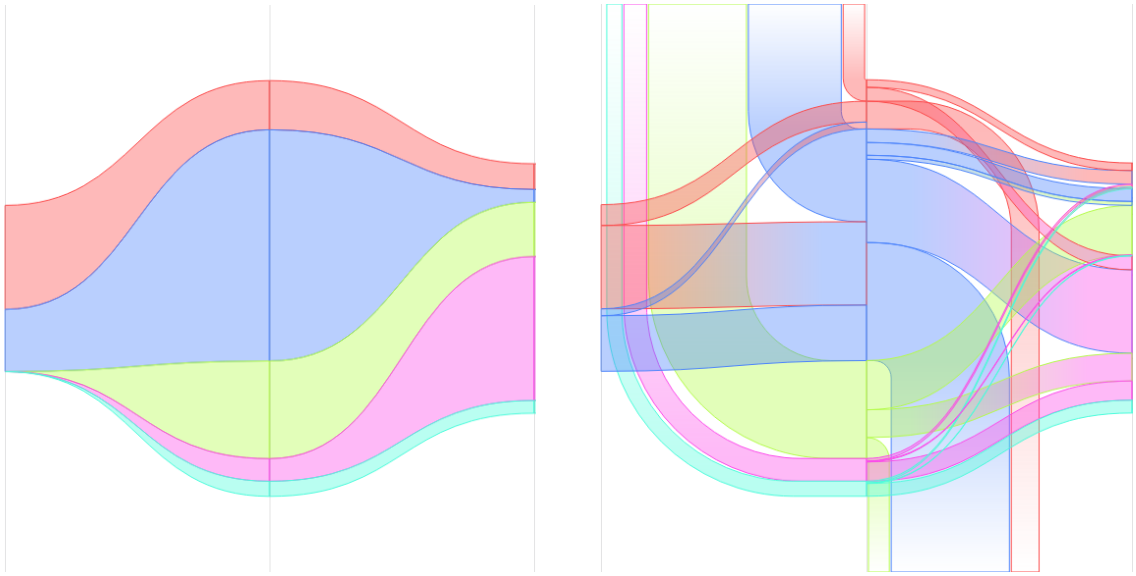
3.3 WOI Rivers

Die Visualisierung der WOI Rivers erfolgt im Wesentlichen wie bei der AOI River-Visualisierung. Zuerst wird eine gestapelte Übersicht der zeitveränderlichen Quantitäten durch einen Fluss, ähnlich der ThemeRiver-Visualisierung angezeigt; zusätzlich können daraufhin Übergänge, Zu- und Abflüsse zwischen Teilflüssen eingeblendet werden. Teilweise wurden hier Änderungen im Vergleich zu den AOI Rivers vorgenommen oder zusätzliche Eigenschaften hinzugefügt.

Im Folgenden wird zunächst die Konstruktion der ThemeRiver-Visualisierung und der einzelnen Elemente für die Detailansicht (Übergänge, Zu- und Abflüsse) beschrieben. Danach wird auf die Erstellung der Transitionsmatrizen eingegangen; diese werden ähnlich der Transitionsmatrizen der AOI Rivers erstellt, müssen allerding für die Visualisierung eines Softwareentwicklungsverlaufs angepasst werden. Zudem werden hierbei verschiedene Varianten vorgestellt, die unterschiedliche Darstellungen für die vorliegenden Daten liefern, indem die Gewichtsverteilung für einzelne Entwickler unterschiedlich vorgenommen wird. Die Transitionsmatrizen bilden die Grundlage für die Visualisierung der einzelnen Komponenten eines WOI Rivers. Anhand eines Beispieldatensatzes können die Ergebnisse der verschiedenen Vorgehensweisen zur Berechnung der Transitionsmatrizen miteinander verglichen werden. Zum Schluss wird noch auf weitere visuelle Besonderheiten (Anordnung der visuellen Elemente und Labels) eingegangen.

3.3.1 Konstruktion

Die Visualisierung des WOI Rivers besteht aus einzelnen Elementen, die zusammen den WOI River bilden. Im Folgenden wird beschrieben, wie diese einzelnen Komponenten konstruiert werden. Das Ergebnis des WOI Rivers bildet sich aus dem Zusammenfügen dieser einzelnen Elemente, wie der Abbildung 3.2 entnommen werden kann.



(a) (b)

Abbildung 3.2: Aufbau eines WOI Rivers aus einzelnen Elementen: (a) für die ThemeRiver-Ansicht und (b) für die Detailansicht

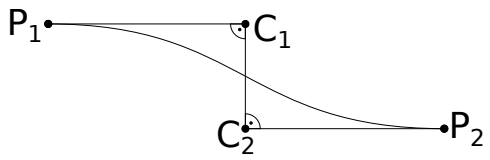
ThemeRiver-Darstellung

Die ThemeRiver-Darstellung entsteht durch gestapelte Darstellung zeitveränderlicher quantitativer Werte. Als Grundlage für die Werte werden Transitionsmatrizen verwendet, bei denen die Summe einzelner Zeilen den verwendeten Quantitäten entspricht. (Die Berechnung der Transitionsmatrizen wird in Kapitel 3.3.2 beschrieben.) Die Anordnung erfolgt hierbei symmetrisch um die x-Achse. Zwischen den Werten aufeinanderfolgender Intervalle wird aus ästhetischen Gründen interpoliert.

Für die Berechnung der Interpolation wird wie bei den AOI Rivers vorgegangen. Hierbei werden kubische Bézier-Kurven verwendet, die aus dem Startpunkt $P_1(x_{P_1}, y_{P_1})$, dem Endpunkt $P_2(x_{P_2}, y_{P_2})$ und zwei weiteren Kontrollpunkten ($C_1(x_{C_1}, y_{C_1})$ und $C_2(x_{C_2}, y_{C_2})$) ermittelt werden (vgl. Abbildung 3.3). Der Start- und Endpunkt entspricht hierbei den Punkten, die interpoliert werden sollen. Die Koordinaten der weiteren Kontrollpunkte berechnen sich folgendermaßen:

$$y_{C_1} = y_{P_1}$$
$$y_{C_2} = y_{P_2}$$
$$x_{C_1} = x_{C_2} = \frac{(x_{P_1} + x_{P_2})}{2}$$

Die Darstellung des WOI Rivers kann aus Elementen aufgebaut werden, die aus jeweils zwei vertikal benachbarten Kurven gebildet werden. Je nachdem, ob die ThemeRiver-Visualisierung oder die Detailansicht angezeigt werden soll, können entsprechende Elemente ein- bzw. ausgeblendet werden.



(a)

Abbildung 3.3: Konstruktionsskizze für eine Bézier-Kurve.

Transitionen, Zu- und Abflüsse

Die AOI Rivers besitzen in der Detailansicht Transitionen, Zu- und Abflüsse. Bei den WOI Rivers kann das Verhalten von Entwicklern zwischen angrenzenden Intervallen dargestellt werden. Die genauen Werte, die für die Berechnung dieser Elemente verwendet werden, sind in den Transitionsmatrizen gespeichert.

Für die Visualisierungen der einzelnen Elemente werden vier Punkte ($P_1(x_{P_1}, y_{P_1})$, $P_2(x_{P_2}, y_{P_2})$, $P_3(x_{P_3}, y_{P_3})$ und $P_4(x_{P_4}, y_{P_4})$) benötigt, zwischen denen der Linienverlauf interpoliert wird.

Transitionen Transitionen werden verwendet um Übergänge von Entwicklern zwischen verschiedenen Arbeitsbereichen darzustellen. Sie werden, wie auch schon die Elemente der ThemeRiver-Ansicht, unter Verwendung kubischer Bézier-Kurven konstruiert. Die Berechnung erfolgt analog zur Beschreibung bei der Erzeugung der einzelnen Elemente für die ThemeRiver-Ansicht. Eine Konstruktionsskizze kann den Abbildungen 3.4a bzw. 3.4b entnommen werden.

Bei den AOI Rivers ist der Abstand zwischen P_1 und P_2 bzw. P_3 und P_4 gleich (Abbildung 3.4a). Dies ist bei den WOI Rivers nicht unbedingt vorausgesetzt, hier können die Abstände auch unterschiedlich sein, was zu einer Verallgemeinerung führt, die in Abbildung 3.4b schematisch dargestellt ist.

Die Farbe bzw. der Farbverlauf, mit dem eine Transition dargestellt wird, ist davon abhängig, zwischen welchen WOIs die Transition verläuft.

Zu- und Abflüsse Für Entwickler, die in einem Zeitintervall neu zu der Entwicklung hinzukommen bzw. diese verlassen, werden Zu- bzw. Abflüsse für die visuelle Repräsentation verwendet. Zuflüsse werden dabei so gezeichnet, dass sie von oben nach unten verlaufen und an gegebenen Positionen in den Hauptfluss einfließen; Abflüsse verlassen entsprechend den Hauptfluss und fließen orthogonal nach unten.

Die Konstruktion erfolgt hierbei mit Hilfe von Kreisen. In Abbildung 3.4c und 3.4d sind die zugehörigen Konstruktionszeichnungen abgebildet.

Die Beschreibung im Folgenden bezieht sich auf einen Zufluss; für den Abfluss kann die Berechnung entsprechend angepasst werden.

Ein Zufluss verläuft von P_1P_2 nach P_3P_4. Während die Punkte P_3 und P_4 durch Positionen im Fluss gegeben sind, werden P_1 und P_2 so berechnet, dass die entsprechenden Punkte aller Zu- und Abflüsse

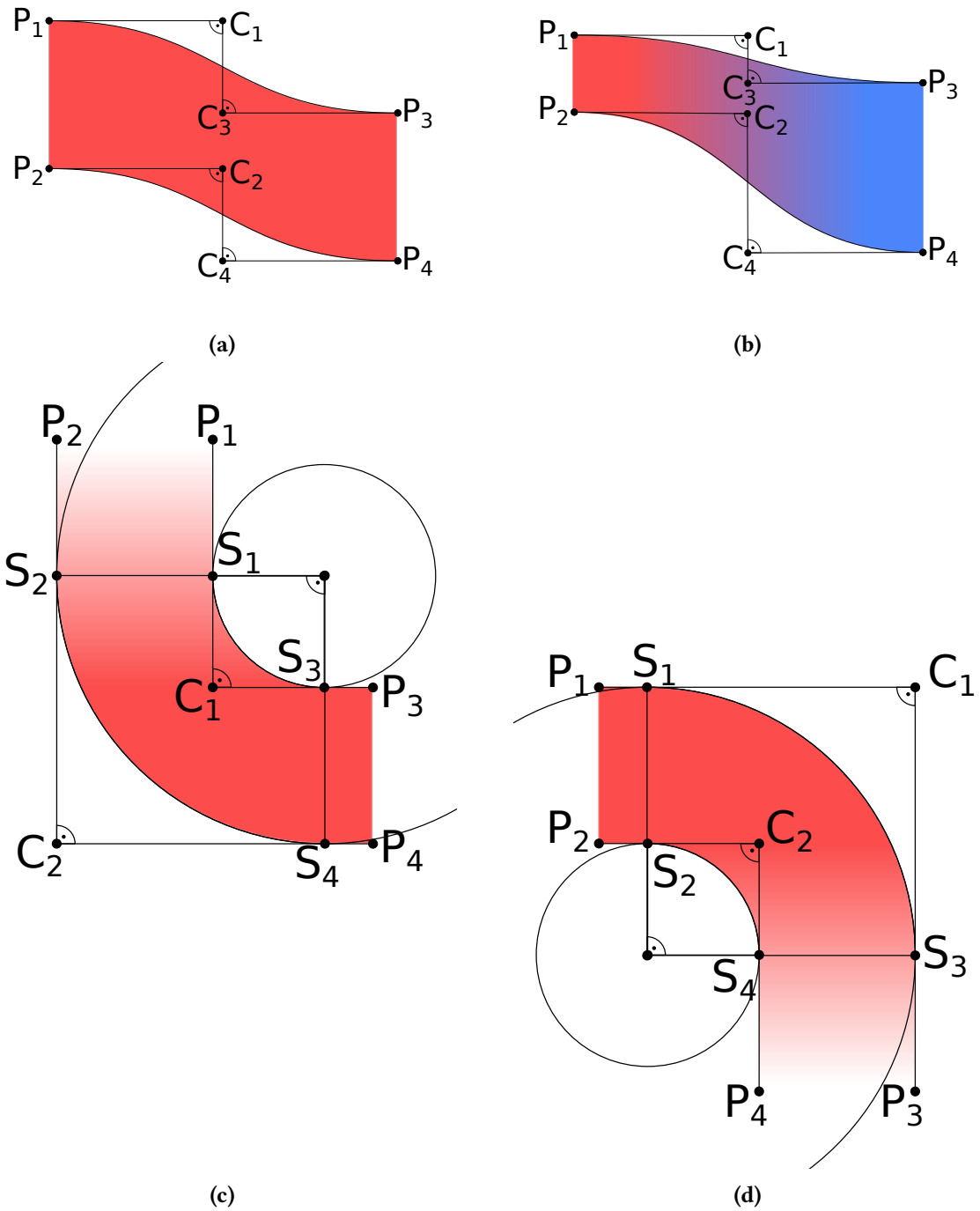


Abbildung 3.4: Konstruktionsskizzen für (a) eine Transition zwischen gleichen WOIs, (b) eine Transition zwischen unterschiedlichen WOIs, (c) einen Zufluss und (d) einen Abfluss.

(a) **(b)** **(c)**

Abbildung 3.5: Zuflüsse mit unterschiedlichen Werten für v: (a) $v = 0$ (b) $v = 1$ (c) $v = 0,7$.

horizontal nebeneinander angeordnet sind. Um keine Überschneidungen zu erzeugen, werden die Koordinaten dieser Punkte so gewählt, dass diese umso weiter links platziert werden, je weiter unten sich der zugehörige Hauptfluss in der Visualisierung befindet.

Der Zufluss soll vertikal nach unten in die Visualisierung und horizontal in den Hauptfluss einfließen. Daher werden für die Konstruktion vier weitere Punkte $S_i(x_{Si}, y_{Si})$ benötigt; deren Koordinaten berechnen sich folgendermaßen:

Mit dem Radius für den inneren Kreis

$r = \min(x_{P3} - x_{P1}, y_{P3} - y_{P1}) \cdot v$ mit $v \in [0, 1]$

ergeben sich folgende Koordinaten für S_i:

$x_{S1} = x_{P1}$
$x_{S2} = x_{P2}$
$y_{S1} = y_{S2} = y_{P3} - r$
$y_{S3} = y_{P3}$
$y_{S4} = y_{P4}$
$x_{S3} = x_{S4} = x_{P1} + r$

Die Abschnitte P_1S_1, P_2S_2, P_3S_3 und P_4S_4 sind gerade Linien. $\overline{S_1C_1}$ bzw. $\overline{S_3C_1}$ legt den Radius des äußeren Kreises fest und $\overline{S_2C_2}$ bzw. $\overline{S_4C_2}$ entsprechend den inneren. Kreisausschnitte verlaufen von S_1 nach S_3 und S_2 nach S_4.

Hierbei gibt v ein Verhältnis für den Radius an. Je Größer v gewählt wird, umso größer ist der Radius des Kreises, d.h. bei großen v werden die geraden Strecken kürzer, bei kleinen v größer. In Abbildung 3.5 sind Zuflüsse für unterschiedlich gewählte Werte für v dargestellt. In den Visualisierungen wurde $v = 0,7$ gewählt, da hierdurch noch kein zu starker „Knick“ entsteht und die Ergebnisse der Visualisierung ästhetisch wirken.

Bei der Untersuchung realer Softwareprojekte wurde festgestellt, dass die Zu- und Abflüsse in der Visualisierung stark dominieren. Damit sie in den Abbildungen nicht als zu störend auffallen, werden sie nach oben bzw. unten durch Verringerung der Deckkraft ausgeblendet.

3.3.2 Anpassung der Transitionsmatrizen

Bei den AOI Rivers erhält jeder Teilnehmer im Verlauf des Flusses durchgehend die gleiche Höhe in der Visualisierung zugewiesen (falls er auf eines der festgelegten AOIs schaut). Hier ist es nicht möglich, dass ein Teilnehmer gleichzeitig auf mehrere Bereiche des Bildschirms schaut. Übertragen auf unseren Fall würde dies bedeuten, dass Entwickler in jedem Zeitintervall nur einen WOI bearbeiten; Entwickler bearbeiten jedoch meistens Dateien in mehreren WOIs gleichzeitig bzw. sorgt hier die Bildung der Intervalle dafür, dass in jedes Intervall Dateiänderungen aus mehreren WOIs fallen.

Würde die Vorgehensweise der AOI Rivers also genau so auf die WOI Rivers übertragen werden, wäre unter der Berücksichtigung, dass Entwickler mehrere WOIs gleichzeitig bearbeiten, keine Visualisierung möglich. Da dies allerdings unter Umständen sehr häufig vorkommt, muss die Erstellung der Transitionsmatrizen angepasst werden.

Deshalb können Übergänge zwischen zwei Zeitschritten bzw. Intervallen nicht mehr nur als Wechsel zwischen zwei Bereichen bzw. als ein Zu- oder Abfluss eines Entwicklers dargestellt werden. Dadurch, dass mehrere WOIs gleichzeitig bearbeitet werden, findet ein Wechsel zwischen verschiedenen Bereichen statt. Demnach ist hier eine Aufteilung der Höhe in den Transitionsbereichen (den Bereichen zwischen zwei Intervallen) notwendig. Außerdem kommt hinzu, dass Entwickler, wenn sie zu dem Projekt hinzu kommen bzw. wenn sie es verlassen, unter Umständen nicht nur ein WOI, sondern mehrere WOIs bearbeiten, deshalb werden sie hierbei auf mehrere Zu- bzw. Abflüsse aufgeteilt.

Zusätzlich arbeiten Entwickler mit unterschiedlich großer Beteiligung an Dateien, was ebenfalls berücksichtigt werden sollte.

Im Folgenden wird für die Berechnung der Transitionsmatrizen jedem Entwickler in jedem Intervall ein Gewicht zugewiesen, das seine Gesamthöhe in der Visualisierung festlegt.

Für den folgenden Verlauf ergeben sich drei Probleme für die Visualisierung jedes Entwicklers, die sinnvoll gelöst werden sollten:

Welches Gewicht erhält ein Entwickler über die Zeit?

Das Gewicht eines Entwicklers gibt die Höhe an, mit der er in dem Fluss berücksichtigt wird. Bei den AOI Rivers wird jeder Teilnehmer jeweils mit der gleichen Höhe berücksichtigt. Auch hier könnte jeder Entwickler die gleiche Höhe erhalten; hierdurch wäre erkennbar, wann wie viele Entwickler in den verschiedenen WOIs arbeiten. Die Höhe jedes Entwicklers könnte sich jedoch auch in jedem Zeitschritt und im Vergleich zu anderen Entwicklern ändern. Wenn nicht die Anzahl der Entwickler, sondern die Beteiligung an dem Projekt interessiert, kann das Gewicht beispielsweise in Abhängigkeit der veränderten Dateien in den entsprechenden Intervallen ermittelt werden. Hierdurch kann erkannt werden, wann bzw. wo es die meisten Änderungen an dem Projekt gab.

Wie wird das Gewicht eines Entwicklers auf die WOIs aufgeteilt, an denen er arbeitet?

Ein Entwickler kann in jedem Intervall verschiedene WOIs bearbeiten. Hier besteht die Möglichkeit, jedem WOI, der in dem entsprechenden Intervall bearbeitet wurde, den gleichen Anteil vom Gewicht des Entwicklers zuzuweisen oder die Gewichtung in Abhängigkeit der bearbeiteten Dateien durchzuführen.

Wie werden die Gewichte für die Transitionen aufgeteilt?

Wenn ein Entwickler in aufeinanderfolgenden Intervallen an unterschiedlichen WOIs beteiligt ist, müssen die Gewichtsanteile für die Transitionen aufgeteilt werden. Ein trivialer Ansatz ist, jeden Gewichtsanteil eines WOIs im ersten Intervall gleichmäßig auf alle WOIs im zweiten Intervall aufzuteilen. Dies hat allerdings teilweise viele Überschneidungen zur Folge; dies wird auch als „Visual Clutter“ [RLN07] bezeichnet. Darüber hinaus ist diese Aufteilung nicht sehr intuitiv. Ein besseres Ergebnis wird erzielt, wenn gezeigt werden kann, dass manche Bereiche in beiden Intervallen bearbeitet werden, und somit ein Übergang mit einem höheren Gewicht erhalten bleiben kann, statt diesen unnötig aufzuteilen.

Im Folgenden werden mögliche Lösungsansätze für diese Fragestellungen beschrieben und wie sie bei der Erstellung der Transitionsmatrizen berücksichtigt werden könnten. Bei der Erklärung werden meist Visualisierungen verwendet, die das Prinzip für einen Entwickler innerhalb von zwei Intervallen veranschaulichen.

Vorbereitung

Zunächst muss der gesamte Zeitbereich in Intervalle eingeteilt werden. Ein Intervall deckt eine gewisse Zeitspanne ab, in die alle Commits fallen, die einen Zeitstempel besitzen, der in diesen Bereich fällt. Durch die Bildung der Intervalle erfolgt eine Aggregation aller Commits, die in die entsprechenden Zeitspannen fallen. Bei den Visualisierungen sollte demnach beachtet werden, dass in den WOI Rivers nur die aggregierten Werte verschiedener Commits innerhalb eines Zeitintervalls korrekt angezeigt werden, dazwischen (in den Transitionsbereichen) wird interpoliert.

Um an die quantitativen Werte für die Visulisierung zu gelangen, müssen Transitionsmatrizen berechnet werden. Deren Berechnung erfolgt grundlegend wie bei den AOI Rivers, wurde allerdings erweitert; Grundlage für die Berechnung bilden die Intervalle, die WOIs $\mathbb{W}$ und alle Entwickler in $\mathbb{G}$.

Es wird so vorgegangen, dass für alle Entwickler $d \in \mathbb{G}$ und jedes Intervall $i : 1 \leq i < I$ bestimmt wird, wie viele und welche WOIs in i und dem darauffolgenden Intervall $i + 1$ bearbeitet wurden.
Falls ein Entwickler im Intervall i keine WOIs bearbeitet hat, bedeutet dies, dass unter Umständen ein Zufluss entsteht, da der Entwickler im Intervall i nichts zum Projekt beigetragen hat. Danach wird für alle WOIs t die von ihm im Intervall $i + 1$ bearbeitet wurden, der zugehörige Eintrag $M_i[s][t]$ der Transitionsmatrix angepasst, wobei s einen Zufluss repräsentiert.
Falls im Intervall $i + 1$ keine WOIs bearbeitet wurden, könnten entsprechend Abflüsse entstehen. Hier wird für alle WOIs t, die im Intervall i bearbeitet wurden, $M_i[s][t]$ angepasst, wobei t einen Abfluss darstellt.
Falls sowohl im Intervall i als auch im Intervall $i+1$ WOIs bearbeitet wurden, wird für alle relevanten WOIs s des Intervalls i für alle bearbeiteten WOIs t des Intervalls $i + 1$ $M_i[s][t]$ angepasst.

Durch I Matrizen $X_i \in X(|D| \times l, \mathbb{N})$ wird repräsentiert welche WOIs bearbeitet wurden und evtl. wie oft. Um diese zu ermitteln, wird über jeden Commit c_p der Geschichte H iteriert und geprüft, ob der zugehörige Zeitstempel t_p in dem darzustellenden Zeitbereich, d.h. in $[t_{min}, t_{max}]$, liegt. Falls dies zutrifft, wird ermittelt, in welches Zeitintervall i t_p fällt. Anschließend wird über alle Dateien, die

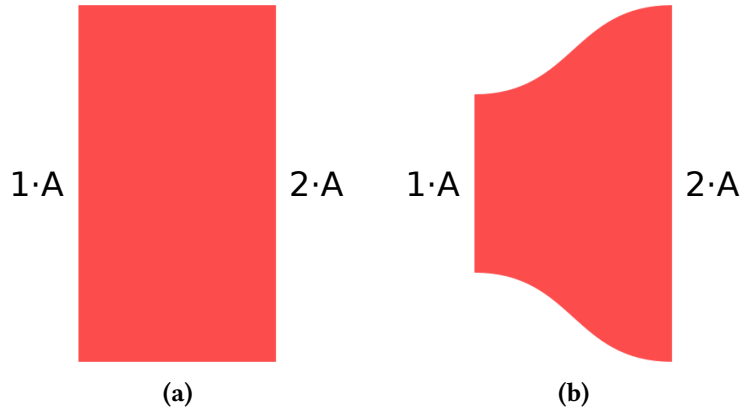


Abbildung 3.6: Verschiedene Methoden zur Vergabe der Höhe eines Entwicklers: (a) jeder Entwickler ist in der Visualisierung über den gesamten Verlauf des Flusses durch die gleiche Höhe dargestellt; (b) die Höhe eines Entwicklers ist abhängig von der Anzahl seiner veränderten Dateien in den entsprechenden Intervallen.

für den Commit verändert wurden, iteriert, der WOI W_a ermittelt, zu dem die entsprechende Datei gehört und der Eintrag in $X_p(d_p, a)$ angepasst.

Gesamthöhe

Die Höhe bzw. die Gewichte eines Entwicklers können entweder für jeden Entwickler und jedes Intervall gleich sein, in Abhängigkeit der Anzahl der Dateien, die in einem Zeitintervall bearbeitet wurden oder in Abhängigkeit der Anzahl der bearbeiteten WOIs bestimmt werden.

Wenn jedem Entwickler in jedem Zeitintervall das gleiche Gewicht zugeordnet wird, entspricht die Gesamthöhe des WOI Rivers der Anzahl an Entwicklern, die im jeweiligen Intervall mindestens eine Datei bearbeitet haben; es ist also erkennbar, wie viele Entwickler über die Zeit an jedem WOI gearbeitet haben. Ein Entwickler wird durchgehend mit der gleichen Höhe gezeichnet.

Bei Gewichten in Abhängigkeit der bearbeiteten Dateien kann in der Visualisierung besser erkannt werden, in welchen Arbeitsbereichen vermehrt gearbeitet wird. Hierbei ist allerdings nicht mehr erkennbar, wie viele Entwickler an den Änderungen beteiligt sind. In diesem Fall wird ein Entwickler im Verlauf des Flusses nicht mehr durchgehend mit der gleichen Höhe dargestellt, sondern sein Anteil verändert sich, je nachdem, ob er im nachfolgenden Intervall jeweils mehr oder weniger Dateien des Projekts bearbeitet hat. Das Gewicht für jeden Entwickler in jedem Intervall kann aus X_i erhalten werden, indem die einzelnen Werte eines Entwicklers in einem Zeitintervall summiert werden.

Außerdem kann das Gewicht in Abhängigkeit der Anzahl der bearbeiteten WOIs ermittelt werden; hierbei wird nicht berücksichtigt, wie viele Dateien in einem entsprechenden WOI bearbeitet wurden, sondern nur, dass sie bearbeitet wurden. Hiermit ist wieder erkennbar, von wie vielen Bearbeitern einzelne WOIs bearbeitet wurden, da jeder Entwickler in jedem bearbeiteten WOI die gleiche Höhe erhält. Hier wird das Gewicht durch Summierung der Vorkommnisse der bearbeiteten WOIs eines Entwicklers in einem Zeitintervall ermittelt.

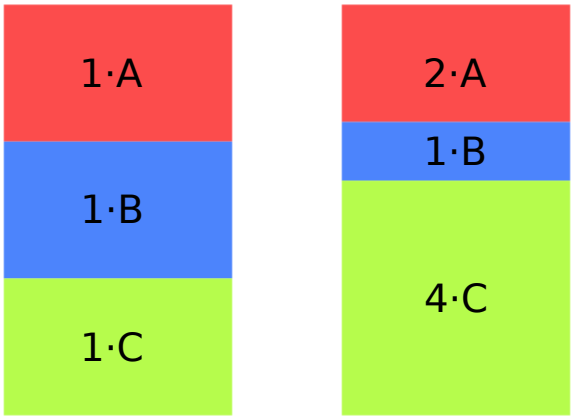


Abbildung 3.7: Veranschaulichung der Aufteilung der Höhe eines Entwicklers: (a) gleichmäßig auf alle bearbeiteten WOIs; (b) in Abhängigkeit der veränderten Dateien innerhalb der WOIs.

In den Abbildungen 3.6a und 3.6b sind Beispiele für die Höhenfestlegung eines Entwicklers dargestellt, welcher jeweils im ersten Intervall eine Datei und im zweiten Intervall zwei Dateien bearbeitet. In Abbildung 3.6a hat der Entwickler durchgehend die gleiche Höhe; in Abbildung 3.6b verändert sie sich in Abhängigkeit der bearbeiteten Dateien.

Aufteilung der Höhe

Wie bereits erwähnt, arbeitet ein Entwickler meistens an mehreren Dateien gleichzeitig, daher ist eine Aufteilung seines Gewichts auf mehrere WOIs in jedem Intervall notwendig.

Hier gibt es die Möglichkeit, dass das Gesamtgewicht eines Entwicklers entweder gleichmäßig auf alle WOIs aufgeteilt wird oder in Abhängigkeit der veränderten Dateien.

Beispiele für eine mögliche Aufteilung der Höhe für die gleichen Daten sind in Abbildung 3.7 dargestellt.

Höhe der Transitionen, Zuflüsse und Abflüsse

Das Teilgewicht, das ein Entwickler für einen WOI in einem Intervall erhalten hat, muss in den Transitionen auf die WOIs im nächsten Intervall aufgeteilt werden.

Da Entwickler in zwei aufeinanderfolgenden Intervallen ein unterschiedliches Gewicht besitzen können, müssen für Transitionen in der Transitionsmatrix jeweils zwei Werte gespeichert werden – einer für das Intervall i und einer für das Intervall $i + 1$. Wenn ein Entwickler im Verlauf des Flusses immer das gleiche Gewicht besitzt, sind beide Werte gleich; dies trifft auch bei Zu- und Abflüssen zu. Die beiden Werte unterschieden sich nur für die Transitionen, die gebildet werden, wenn sich das Gewicht eines Entwicklers über die Zeit verändert.

Im einfachsten Fall kann das Teilgewicht, das ein Entwickler für ein WOI zugewiesen bekommen hat, gleichmäßig auf alle WOIs im nächsten Intervall aufgeteilt werden. Diese Vorgehensweise ist zwar recht einfach zu berechnen, allerdings nicht intuitiv für das Verständnis der Visualisierung, da viele unnötige Übergänge entstehen können.

Für die folgenden Beispiele sind in Abbildung 3.8 Visualisierungen für jeweils die ThemeRiver-Darstellungen, die eben beschriebene Methode und eine Optimierung dargestellt.
Angenommen ein Entwickler arbeitet über längere Zeit an zwei verschiedenen WOIs A und B, durchgehend zu gleichen Anteilen, dann sollten diese WOIs nicht aufgeteilt werden (vgl. Abbildung 3.8b) sondern die Übergänge jeweils zwischen den gleichen WOIs verlaufen, wie in Abbildung 3.8c dargestellt.
Der gleiche Grundgedanke gilt auch, wenn ein Entwickler zuerst weniger an A arbeitet und im nächsten Intervall mehr. Hier wäre es entsprechend nachvollziehbarer, größtmögliche Arbeitskapazitäten zu gleichen WOIs überfließen zu lassen und nur die restliche Kapazität auf die weiteren WOIs aufzuteilen (Abbildung 3.8f).
Entsprechendes gilt auch beispielsweise, wenn ein Wechsel von zwei zu drei WOIs stattfindet. Zum einen könnte hier auch die Höhe von WOI A und B in je drei Anteile unterteilt werden, sodass die neuen WOIs aus je einem Anteil von A und B gebildet werden. Auch hier wäre es allerdings sinnvoller und intuitiver, wenn die größtmögliche Höhe von A bzw. B beibehalten wird und nur der neue WOI aus der restlichen Kapazität entsteht (Abbildung 3.8i).
In Abbildung 3.9 ist zudem noch ein Extremfall dargestellt, bei dem in zwei aufeinanderfolgenden Intervallen die genau gleichen WOIs bearbeitet werden.

Für diese Erweiterung muss zuerst für alle WOIs geprüft werden, ob der gleiche WOI auch im nachfolgenden Intervall bearbeitet wird. Falls dies zutrifft, wird das maximale Gewicht ermittelt, das zwischen den beiden WOIs fließen kann. Diese Werte müssen gespeichert werden, um später noch das restliche Gewicht auf die weiteren WOIs aufteilen zu können.

Durch Rundungsfehler bei der Fließkommazahlenrechnung können sehr viele sehr dünne Transitionen entstehen, da ein durch Rundungsfehler übrig gebliebener Wert auf die weiteren WOIs verteilt wird. Daher wird zusätzlich gespeichert, ob das gesamte Gewicht eines WOIs für eine Transition bereits vergeben ist.

Die Werte, die hier für jeden Entwickler ermittelt werden, müssen in der Transitionsmatrix für die entsprechenden WOIs und Intervalle addiert werden.

Übersicht der Darstellungsmöglichkeiten

In den Abbildungen 3.10, 3.11 und 3.12 sind die Ergebnisse der hier vorgestellten Visualisierungstechnik anhand eines Beispieldatensatzes veranschaulicht. Der Datensatz enthält 5 Entwickler und es wurden 5 WOIs festgelegt.

Während in Abbildung 3.10a die Hierarchievisualisierung der Systemstruktur mit den ausgewählten WOIs dargestellt ist, können in 3.10b und 3.10c WOI River-Visualisierungen als Übersicht und Detailansicht gefunden werden. Bei den hier abgebildeten Visualisierungen wird jeder Entwickler mit der gleichen Höhe dargestellt. Die Höhe jedes Entwicklers wird in Abhängigkeit der bearbeiteten

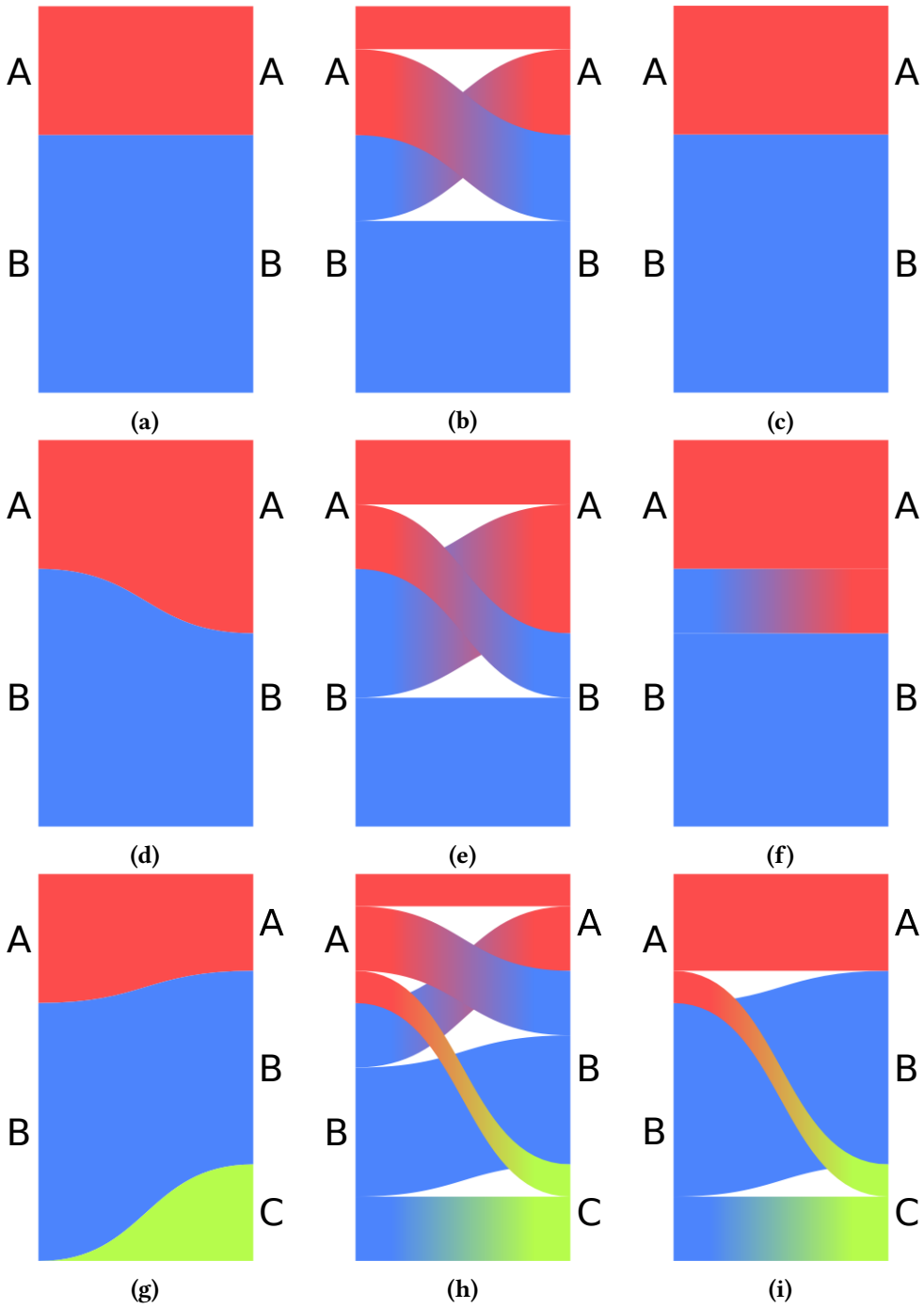
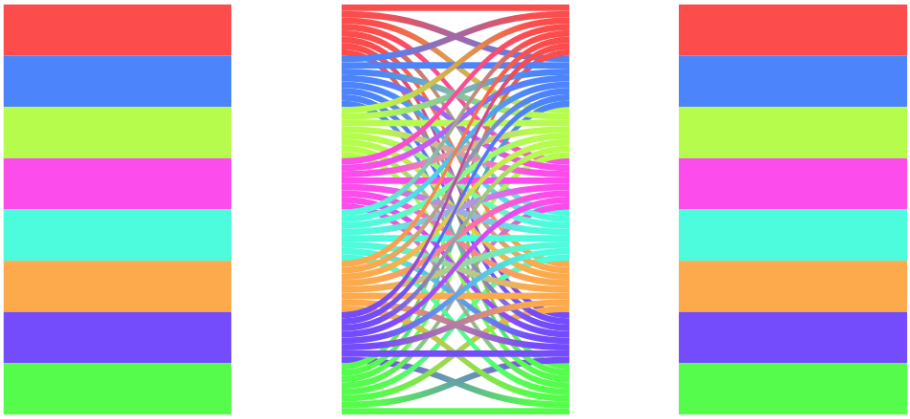


Abbildung 3.8: Veranschaulichung des WOI Rivers für zwei Intervalle und einen Entwickler, der an zwei (a – f) bzw. drei (g – i) WOIs arbeitet. Während bei (a – c) beide WOIs in beiden Intervallen zu gleichen Anteilen bearbeitet werden, wird in (d – f) eine unterschiedliche Gewichtung für die WOIs verwendet. In (g – h) kommt eines weiterer WOI hinzu. (a), (d) und (g) zeigen die ThemeRiver-Visualisierungen, die zwei nachfolgenden Bilder jeweils die dazugehörige Detailansicht. Bei (b), (e) und (h) werden Übergänge gleichmäßig aufgeteilt; bei (c), (f) und (i) werden Transitionen zwischen gleichen WOIs nicht unnötig aufgeteilt.



(a) (b) (c)

Abbildung 3.9: Veranschaulichung des WOI Rivers für zwei Intervalle und einen Entwickler, bei dem in jedem Intervall die gleichen WOIs mit jeweils gleichen Gewichten dargestellt werden: (a) ThemeRiver-Ansicht; (b) Detailansicht, bei der Transitionen gleichmäßig aufgeteilt werden; (c) Detailansicht, bei der Transitionen nicht unnötig aufgeteilt werden.

Dateien in den einzelnen WOIs aufgeteilt und Transitionen werden nicht unnötig aufgeteilt. Diese Visualisierung gibt demnach einen Überblick über das Entwicklerverhalten, aus dem ersichtlich werden kann, wie viele Entwickler ihre Arbeitsleistung prozentual in die verschiedenen Arbeitsbereiche investiert haben.

In den Abbildungen 3.11 und 3.12 sind Visualisierungen zum gleichen Datensatz als Übersicht über die Ergebnisse der beschriebenen Varianten der Transitionsmatrizen dargestellt. Während in 3.11 alle WOI Rivers nur einen Entwickler berücksichtigen, werden in 3.12 alle Entwickler berücksichtigt. In den oberen Darstellungen ist die Höhe jedes Entwicklers gleich und in den unteren abhängig von der Anzahl der bearbeiteten Dateien (rechts) bzw. WOIs (links). Links erfolgt eine gleichmäßige Aufteilung der Höhe auf alle bearbeiteten WOIs, rechts in Abhängigkeit der bearbeiteten Dateien. Es sind jeweils eine Übersicht und die Detailansichten mit einer gleichmäßigen Aufteilung der Transitionen und der optimierten Variante (die ein unnötiges Aufteilen der Transitionen verhindert) zu sehen.

3.3.3 Vertikale Anordnung

In der vertikalen Anordnung der Teilflüsse eines WOIs werden zuerst die Zuflüsse gezeichnet, danach alle Transitionen und zum Schluss folgen die Abflüsse. Die Transitionen werden dabei in der Reihenfolge, in der sie in den Transitionsmatrizen bzw. im WOI River vorkommen angeordnet, um keine unnötigen Überschneidungen zu erzeugen. Diese Anordnung wurde im Vergleich mit der Umsetzung bei den AOI Rivers verändert. Bei diesen werden zuerst Transitionen zum selben WOI und danach erst die Transitionen, die zu den restlichen WOIs fließen, in vertikaler Reihenfolge angeordnet. Bei der optimierten Anordnung für WOI Rivers überschneiden sich die verschiedenen Arten der Transitionen nicht unnötig (wie in Abbildung 3.13 anhand eines Beispiels für einen AOI River und einen WOI River zu sehen ist).

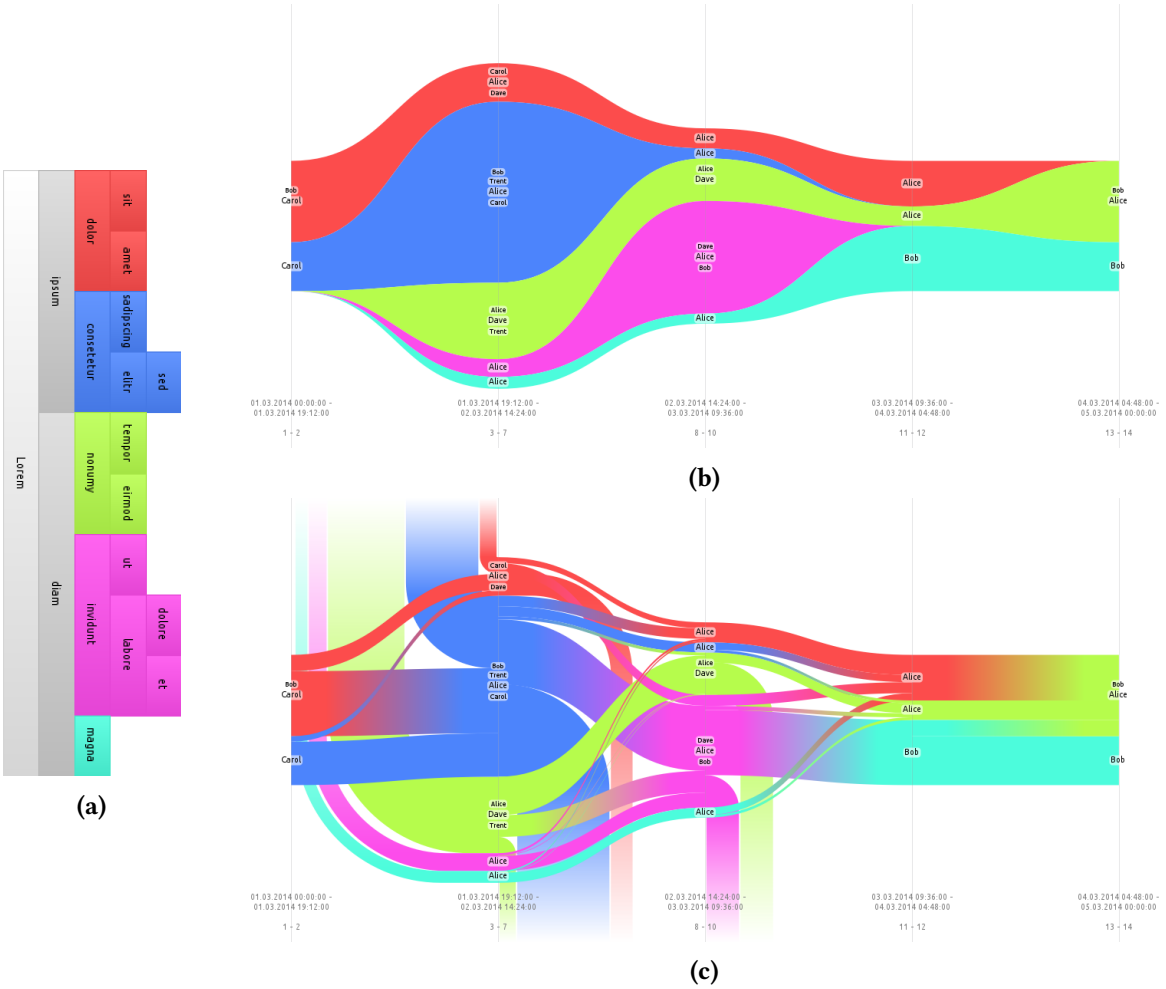


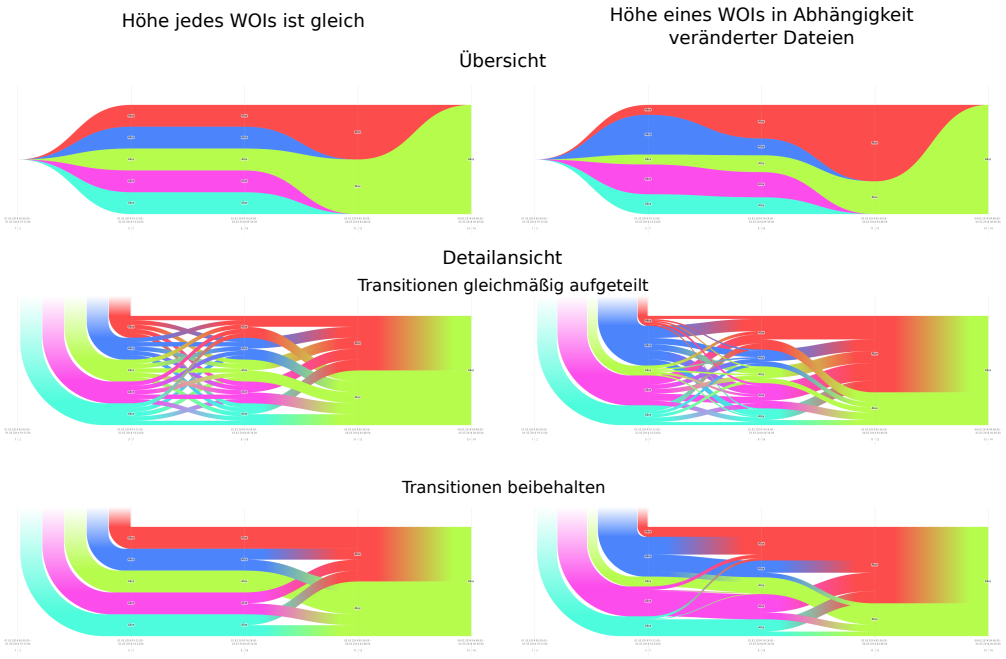
Abbildung 3.10: Visualisierung für einen Beispieldatensatz mit (a) der Projekthierarchie, (b) der Übersichts-Visualisierung und (c) der Detailansicht.

3.3.4 Zeichenreihenfolge

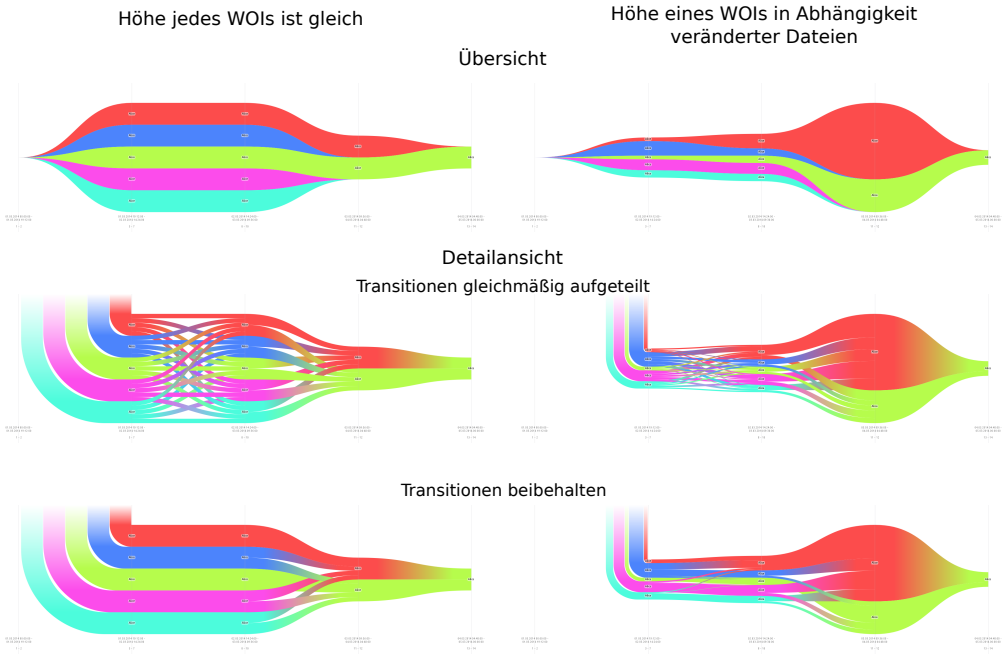
Im Hintergrund werden die Zu- und Abflüsse gezeichnet. Die restlichen Transitionen werden im Vordergrund abgebildet. Alle Transitionen eines Transitionsbereichs werden nach ihrer Höhe sortiert und so angeordnet, dass dünnere Übergänge weiter vorne gezeichnet werden, damit sie nicht von dickeren Übergängen überdeckt werden.

3.3.5 Labels

Es gibt verschiedene Labels für die WOI River Visualisierung. Nicht alle Labels können gleichzeitig angezeigt werden, da die Berechnungen der Positionen unabhängig voneinander ablaufen.

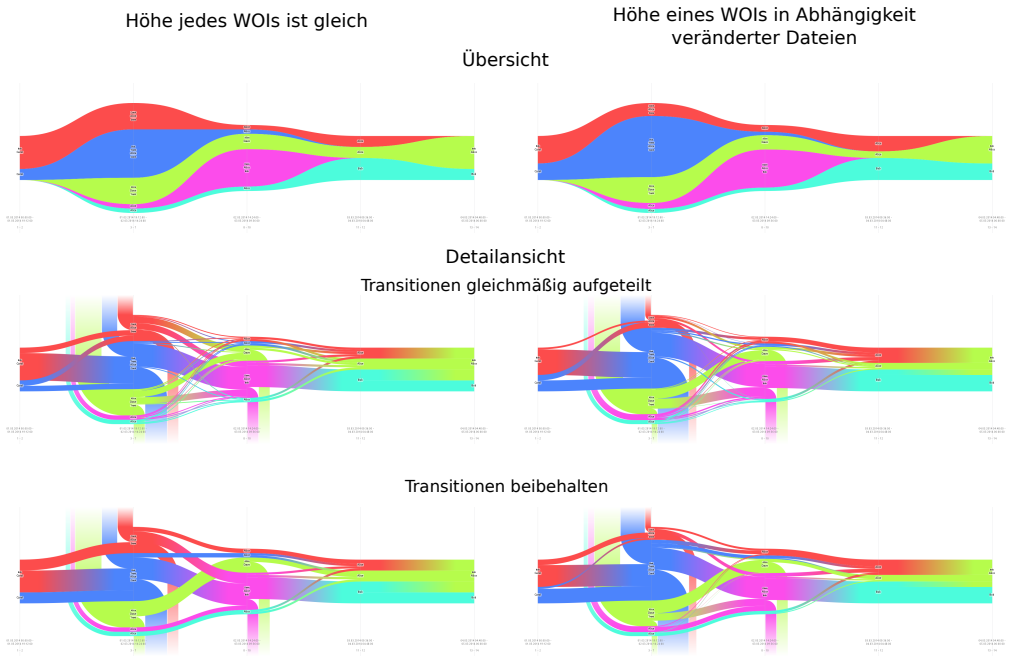


(a)

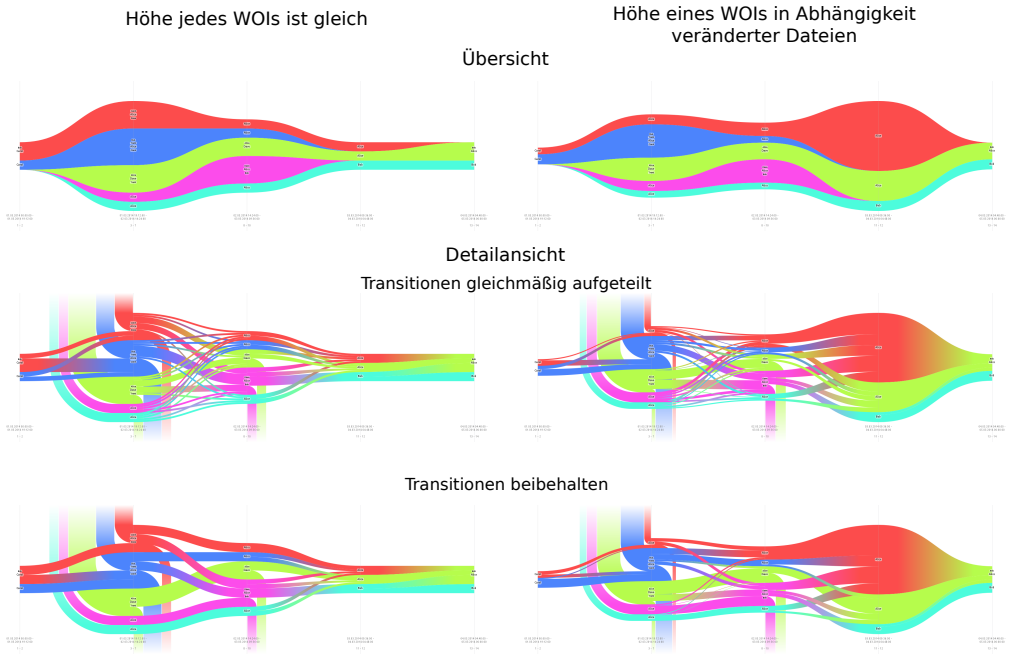


(b)

Abbildung 3.11: Übersicht über verschiedene Darstellungsmöglichkeiten für einen Beispieldatensatz anhand eines Entwicklers. (a) In Abhängigkeit der Anzahl an Entwicklern und (b) in Abhängigkeit der Anzahl an veränderten Dateien.

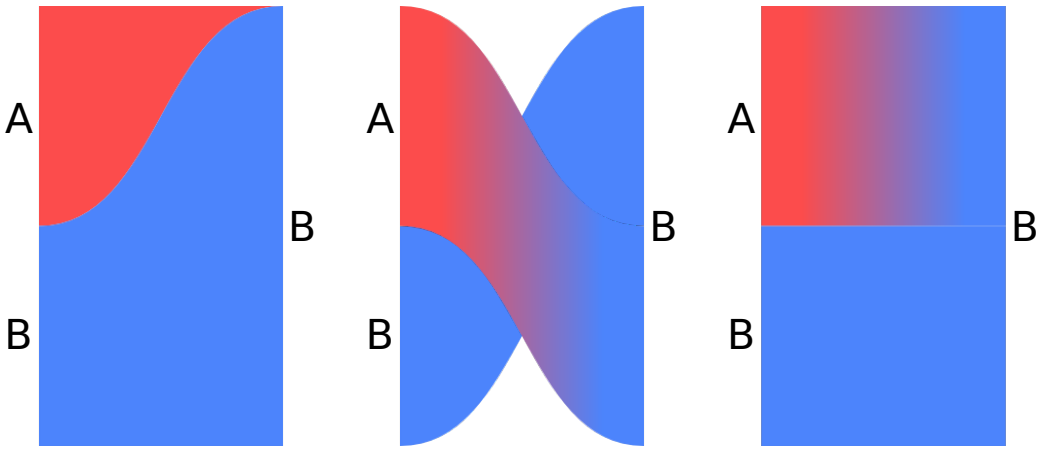


(a)



(b)

Abbildung 3.12: Übersicht über verschiedene Darstellungsmöglichkeiten für einen Beispieldatensatz mit mehreren Entwicklern. (a) In Abhängigkeit der Anzahl an Entwicklern und (b) in Abhängigkeit der Anzahl an veränderten Dateien.



(a) (b) (c)

Abbildung 3.13: Anpassung der Reihenfolge im Vergleich mit den AOI Rivers: (a) Übersicht; (b) AOI River-Reihenfolge; (c) WOI River-Reihenfolge.

(a) (b) (c)

Abbildung 3.14: Veranschaulichung einiger Methoden zur Anzeige von Labels: (a) Intervall-Labels, (b) Entwickler-Labels und (c) WOI-Labels.

Allgemein bestehen die Labels (bis auf die Intervall-Labels) aus semi-transparenten abgerundeten Rechtecken, auf die der entsprechende Text gezeichnet wird. Die Farbe der Rechtecke ist je nach Art des Labels weiß bzw. schwarz und der Text schwarz bzw. weiß. Durch Verwendung von Transparenz ist es möglich, den Umriss des Flusses trotz Labels zu erkennen.

Die Intervall-Labels werden grau gezeichnet.

Intervall-Labels

Für jedes Intervall kann ein Label mit dem Zeitbereich, aus dem die Daten für die Flussdarstellung stammen, sowie mit den Nummern der in dem Intervall vorkommenden Commits (alle Commits sind dabei von 1 bis zur Anzahl an verwendeten Commits durchnummeriert). Ein Beispiel ist in Abbildung 3.14a dargestellt.

Entwickler-Labels

In jedem Zeitintervall können für jeden WOI die Entwickler mit den meisten Beteiligungen angezeigt werden (vgl. Abbildung 3.14b). Der Entwickler, der die meisten Dateien verändert hat, wird in der Mitte dargestellt; nach außen sinkt der Beitrag der Entwickler. Zusätzlich spiegelt sich in der verwendeten Schriftgröße die Beteiligung der Entwickler wider: Je mehr Dateien von einem Entwickler verändert wurden, umso größer ist dessen Schriftgröße; dabei wird allerdings die Schriftgröße beim Erreichen eines minimalen Wertes nicht weiter verkleinert, um die Lesbarkeit zu erhalten. Der vertikale Platz, der mit Labels aufgefüllt werden kann, entspricht der Höhe des jeweiligen Flusses in dem entsprechenden Intervall. Weniger wichtige bzw. kleinere Flüsse erhalten dadurch keine Labels während in den größeren mehrere Entwickler angezeigt werden können.

WOI-Labels

Um einen schnellen Überblick über die Bedeutungen der einzelnen Flüsse in der Projekthierarchie zu erhalten, können die Bezeichnungen der WOIs als Labels angezeigt werden, wie z.B. in Abbildung 3.14c zu sehen ist. Für jeden Fluss werden die Labels horizontal an der Stelle mit der weitesten Ausdehnung gezeichnet; gibt es mehrere Stellen mit der gleichen vertikalen Ausdehnung, wird eine Position verwendet, die nach Möglichkeit weit links im Fluss liegt. Horizontal befinden sich die Labels mittig im Fluss des entsprechenden WOIs. Die Labels werden in der Reihenfolge ihrer maximalen Ausdehnungen gezeichnet, sodass zuerst Labels für Bereiche mit größeren Ausdehnungen, gefolgt von den restlichen gezeichnet werden. Zudem werden sie nur dann gezeichnet, wenn sie sich mit keinen bereits gezeichneten Labels überschneiden. Hierdurch sind wichtigere Labels sichtbar und überdecken sich nicht mit weiteren Labels.

Werte-Labels

Für jeden Fluss können in jedem Intervall Werte-Labels angezeigt werden, um herauszufinden, wie viele Entwickler an einem Fluss arbeiten (relativ gesehen) bzw. wie viele Dateien bearbeitet wurden. Hierbei wird ähnlich wie bei den WOI-Labels vorgegangen: Die horizontale Positionierung erfolgt in jedem Fluss mittig; zuerst werden höhere Werte gezeichnet und nur dann wenn sie sich nicht mit bereits gezeichneten Labels überschneiden.

Summe der Entwickler bzw. veränderter Dateien

Zudem kann über jedem Intervall ein Label platziert werden, das die Anzahl der Entwickler bzw. die Anzahl der veränderten Dateien in dem entsprechenden Intervall anzeigt.

3.4 Farbvergabe

Durch Verwendung von Farbe wird die Hierarchievisualisierung mit der WOI River-Visualisierung verbunden, indem ein Fluss des WOI River jeweils die gleiche Farbe wie die zugehörigen Arbeitsbereiche der Systemstruktur erhält. Dabei sollte versucht werden, Farben mit einem möglichst hohen Kontrast zueinander zu verwenden, um die einzelnen Bereiche unterscheiden zu können und eine eindeutige Zuordnung zwischen den Arbeitsbereichen in der Hierarchie und dem entsprechenden Fluss zu ermöglichen. Eine zusätzliche Schwierigkeit ist, dass die Anzahl der Farbwerte, die benötigt werden, nicht von vornherein bekannt ist, da in der Anwendung beliebig viele weitere WOIs angelegt werden können und eine Anpassung der Farben bei der Festlegung jedes neuen WOIs keine sinnvolle Lösung wäre.

Die Farbvergabe für die WOIs kann auf drei verschiedene Arten erfolgen:

Mit Hilfe des Verfahrens von Ankerl [Ank09] können Farben mit hohem Kontrast für jeweils zwei aufeinander folgende Farben erzeugt werden. Hierfür werden Farben im HSV-Farbraum berechnet, wobei die Farbsättigung (**s**aturation) und die Helligkeit (**v**alue) konstant bleiben; nur der Farbton (**h**ue) ändert sich. Die Berechnung des Farbtons basiert auf der Verwendung des Goldenen Schnitts, um gleichmäßig verteilte Werte zu erhalten, unabhängig davon, wie viele Farben verwendet werden. Farben wiederholen sich bei dieser Technik nie, allerdings können sie mit der Zeit ähnlich zu bereits verwendeten Farben sein.

Außerdem besteht die Möglichkeit, Farbwerte aus einer vordefinierten Liste an Farben zu verwenden, die so gewählt wurden, dass sie ästhetisch wirken und unterscheidbar sind. Die Liste besitzt allerdings eine begrenzte Anzahl an Farbwerten, wodurch eine Wiederholung der Farben in der Visualisierung nicht ausgeschlossen ist.

Darüber hinaus können die Farben auch für jeden WOI manuell festgelegt werden.

4 Implementierung

Die Anwendung für die Visualisierung der WOI Rivers ist in C++ und Qt implementiert. Sie bietet die Möglichkeit, die Systemstruktur eines Softwareprojekts anzuzeigen und für ausgewählte Arbeitsbereiche den zugehörigen WOI River zu visualisieren. Dabei gibt es verschiedene Einstellungs- und Interaktionsmöglichkeiten um den Entwicklungsprozess eines Softwareprojekts zu untersuchen.

In dem hier abgebildeten Diagramm ist der schematische Ablauf dargestellt, wie Visualisierungen zu einem Softwareentwicklungsprozess erhalten werden können; dabei werden die Gewinnung der Eingabedaten und die Hauptmerkmale der Anwendung berücksichtigt.

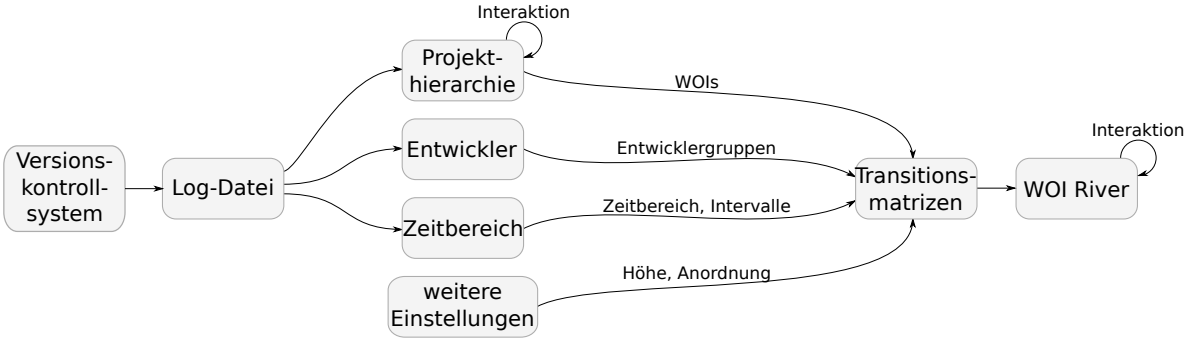


Abbildung 4.1: Veranschaulichung der wichtigsten Elemente der Datei-Erzeugung und Implementierung – von der Datenquelle bis zur Visualisierung des WOI Rivers.

Zuerst muss aus einem Repository eines Softwareprojekts eine Log-Datei der Entwicklungsgeschichte erzeugt werden (genaueres dazu folgt in Abschnitt 4.3). Diese Datei kann in die Anwendung geladen werden und aus den enthaltenen Commit-Informationen wird die zugehöriger Projekthierarchie aufgebaut und visualisiert; außerdem werden die Zeitpunkte des ersten und letzten Commits und die Namen aller beteiligten Entwickler ermittelt.
In der Hierarchie können anschließend interaktiv WOIs definiert werden; außerdem ist es möglich, den Zeitbereich und die Art der Intervalle festzulegen sowie Entwicklergruppen zu erstellen, die für die Visualisierung berücksichtigt werden sollen. Beschreibungen dieser Grundfunktionen folgen in Abschnitt 4.4. Aus diesen Informationen bzw. Einstellungen werden die Transitionsmatrizen für den WOI River berechnet, welche daraufhin für dessen Visualisierung Verwendung findet.

In der Anwendung spielt Interaktion eine sehr große Rolle. Bereits bei der Auswahl und Festlegung verschiedener WOIs können innerhalb der Systemstruktur interaktiv verschiedene Arbeitsbereiche als WOIs ausgewählt werden. Nachdem der zugehörige WOI River angezeigt wird, bietet dieser zunächst als ThemeRiver einen Überblick darüber, wie viele Entwickler an dem Projekt beteiligt sind bzw. wie

viele Veränderungen an dem Projekt in den verschiedenen Arbeitsbereichen über die Zeit vorgenommen wurden. Erst durch Interaktion ist es möglich, weitere Besonderheiten herauszufinden.

In vielen Anwendungen findet sich das Prinzip des Visual Information Seeking Mantra: „Overview first, zoom and filter, then details-on-demand“ [Shn96]. In der hier entwickelten Anwendung kann dieses Prinzip sowohl bei der Hierarchie- als auch bei der WOI-River-Visualisierung wiedergefunden werden.

Betrachten wir zuerst die Hierarchiedarstellung: Zunächst wird eine Übersicht über die Systemstruktur des gesamten Projekts angezeigt. Anschließend kann sowohl geometrisch als auch semantisch gezoomt werden und weitere Details zu einem Knoten können durch Klick auf diesen herausgefunden werden.

Entsprechend bietet die WOI River-Visualisierung zunächst als ThemeRiver eine Übersicht des Entwicklungsverlaufs; durch das Anzeigen von Transitionen sowie Zu- und Abflüssen und das Einblenden weiterer Informationen in Bezug auf einen Fluss, kann das Visualisierte besser verstanden werden.

Beide Visualisierungen bieten noch weitere Einstellungs- und Interaktionsmöglichkeiten, auf die in Abschnitt 4.5 genauer eingegangen wird.

Im Folgenden wird zuerst beschrieben, wie eine Datei, welche Informationen bezüglich der Entwicklungsgeschichte enthält, aus dem Repository eines Versionskontrollsystems gewonnen und für die Anwendung angepasst werden kann. Danach werden die Benutzeroberfläche und ihre Hauptbestandteile grob vorgestellt und es folgt eine Beschreibung des Imports, der Grundfunktionen sowie weiterer Funktionalitäten und Interaktionsmöglichkeiten des Programms. Zum Schluss werden noch verschiedene Export-Möglichkeiten beschrieben.

4.1 Vorverarbeitung

Es gibt heute sehr viele verschiedene Versionskontrollsysteme. Bei jedem entstehen unterschiedlich formatierte Log-Ausgaben, welche Informationen zur Entwicklungsgeschichte eines Projekts enthalten. Die Datensätze, die im Rahmen dieser Arbeit verwendet wurden, stammen aus Git-, Mercurial-, Subversion- und Bazaar-Repositories. Die hier entwickelte Anwendung verwendet ein unabhängiges Format und ist daher nicht auf ein bestimmtes Versionskontrollsystem beschränkt.

Mercurial- oder Subversion-Repositories bieten bereits die Möglichkeit, die Versionsgeschichte im XML-Format zu erzeugen. Diese Dateien können direkt in die Anwendung geladen werden. Zur Erzeugung der Log-Dateien können folgende Programmaufrufe verwendet werden:

Mercurial: *hg log --style=xml -v*

Subversion: *svn log --xml -v*

Bei der Suche nach Open-Source-Softwareprojekten wurde festgestellt, dass viele Projekte inzwischen mit Git verwaltet werden. Da es hierfür keine XML-Ausgabe der Versionsgeschichte gibt, muss dies auf eine andere Art, beispielsweise durch ein Skript erzeugt werden. Bei Bazaar gibt es zwar eine XML-Ausgabe, diese muss allerdings zuerst an das entsprechende Format angepasst werden.

Der genaue Aufbau des Eingabeformats folgt in Kapitel 4.3.

4.1.1 Vorfilterung

Es kommt vor, dass Entwickler in einem Commit nur eine kleine Änderung machen, die jedoch eine Auswirkung auf sehr viele andere Dateien hat. Beispielsweise könnte der Name einer Klasse verändert werden, was zur Folge hätte, dass alle Dateien, die diese Klasse verwenden, auch verändert werden müssen. Solche Commits sind meistens die Folge von Änderungen an der Infrastruktur und sorgen für Rauschen. In [ZW04] wird dieses Problem durch Filtern aller Commits, die mehr als N Änderungen enthalten, behoben. Eine sinnvolle obere Grenze für N ist dabei von dem jeweiligen Projekt abhängig.

4.2 Benutzeroberfläche

In Abbildung 4.2 ist die Benutzeroberfläche der Anwendung zu sehen. Sie besteht aus vier Hauptteilen: der Hierarchievisualisierung zur Auswahl von WOIs, der Hauptansicht mit der Darstellung des WOI Rivers und den Dock Widgets, in denen der Zeitbereich, Intervalle und Entwicklergruppen festgelegt sowie weitere Einstellungen vorgenommen werden können. Ein weiteres Dock Widget bietet eine Übersicht über den WOI River, in der die aktuelle Position des dargestellten Bereichs in Bezug auf die Hauptansicht erkennbar ist.

Hierarchieansicht

Für die Visualisierung der Projekthierarchie stehen zwei verschiedene visuelle Metaphern zur Verfügung: zum einen ein Layered Icicle Plot, zum anderen ein Indented Plot (vgl. Abbildung 3.1). Die inneren Knoten repräsentieren im Normalfall Verzeichnisse und die Blattknoten Dateien (oder leere Verzeichnisse).

Die Hierarchievisualisierung bietet die Grundlage zur Festlegung von WOIs für die WOI River-Visualisierung; zusätzlich bietet sie Interaktions- und Navigationsmöglichkeiten um die visualisierten Daten zu reduzieren.

WOI River-Ansicht

In der Hauptvisualisierung wird der WOI River zu den zuvor festgelegten WOIs, Entwicklern und dem angegebenen Zeitbereich dargestellt. Hierbei ist die Darstellung in Hinsicht auf mehrere Parameter veränderbar und Interaktionstechniken unterstützen die Analyse der visualisierten Daten.

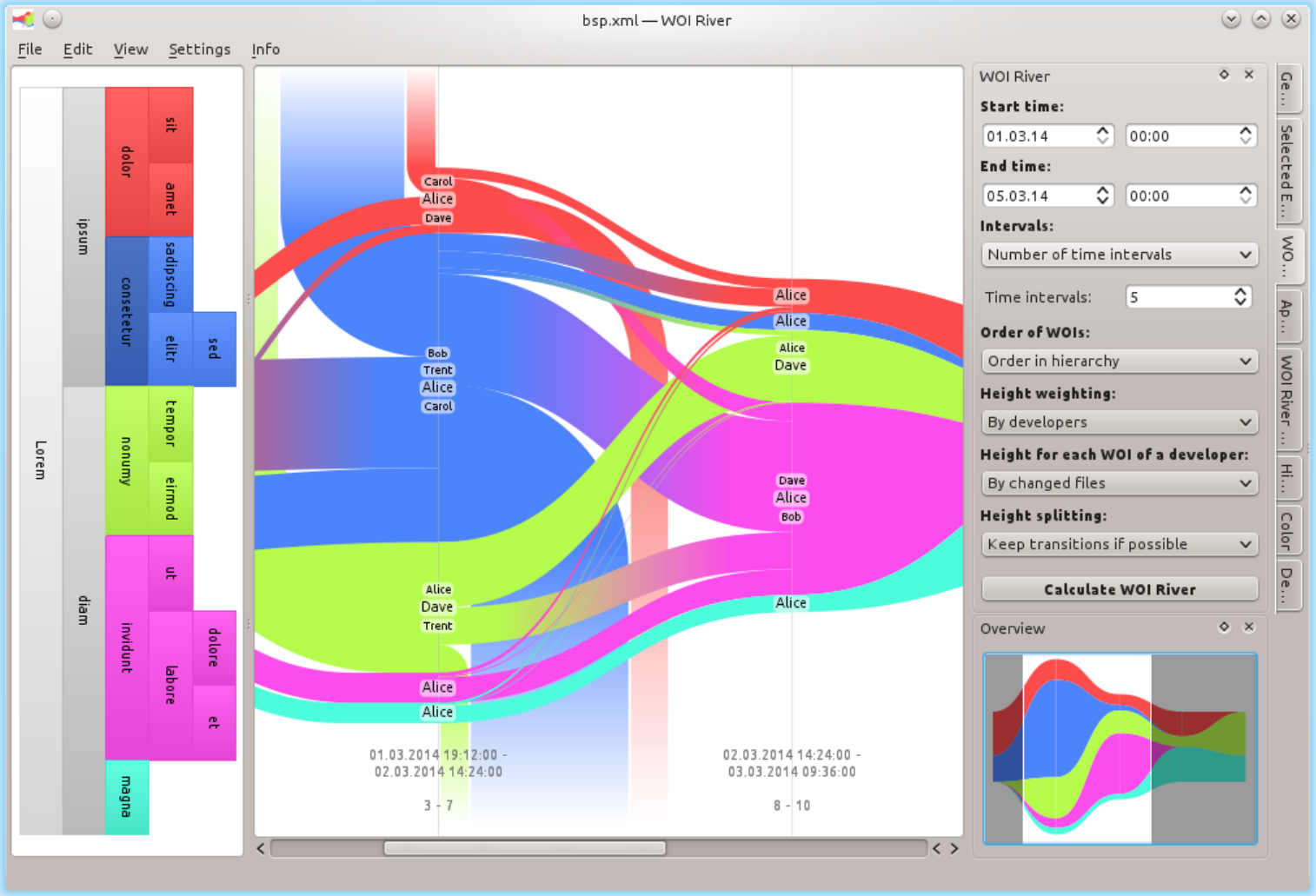


Abbildung 4.2: Benutzeroberfläche der Anwendung zur Visualisierung der WOI Rivers.

Dock Widgets

In den Dock Widgets können verschiedene Einstellungsmöglichkeiten für die Hierarchie- und WOI River-Visualisierung gefunden werden. Außerdem können allgemeine Informationen zu den geladenen Daten und visualisierten Elementen angezeigt werden.

Es stehen insgesamt neun Dock Widgets mit unterschiedlichen Informationen, Funktionen und Einstellungsmöglichkeiten bereit. Die einzelnen Dock Widgets können ausgeblendet sowie aus dem Fenster gelöst werden, um dadurch mehr Platz für die Hauptvisualisierungen zur Verfügung zu haben.

Menü

Im Anwendungsmenü findet sich die Möglichkeit zum Importieren der Log-Datei und verschiedene Export-Funktionalitäten. Darüber hinaus gibt es hier die Möglichkeit, Teile des Fensters ein- bzw. auszublenden und eine Funktion zur Festlegung von WOIs durch die Verwendung von Filtern.

4.3 Import

Wie bereits beschrieben werden für die Visualisierung Commit-Informationen eines Repositories benötigt, welche jeweils Informationen bezüglich des Änderungsdatums, des Entwicklernamens und der veränderten Dateien enthalten. Als Eingabe erwartet die Anwendung eine XML-Datei, welche diese Informationen enthält; das Format dieser Datei muss folgendem Schema entsprechen:

```
<log>
  <logentry>
    <date>2013-15-09T00:00:00+01:00</date>
    <name>Alice</name>
    <paths>
      <path>Lorem/ipsum/dolor/amet</path>
      <path>Lorem/diam/nonumy/eirmod</path>
      ...
    </paths>
    </logentry>
    ...
</log>
```

Innerhalb eines *log*-Elements können die einzelnen Commits in beliebiger Reihenfolge angegeben werden. Jeder *logentry*-Eintrag repräsentiert einen Commit, *date* enthält das Datum des Commits, *name* den Namen des Entwicklers und in *paths* werden alle Pfade separat zwischen *path*-Tags angegeben.

Der Name eines Entwicklers kann aus einer beliebigen Zeichenkette (in UTF-8 kodiert) bestehen, der Zeitstempel muss im ISO-8601-Format [iso] angegeben sein und die Pfade können beliebig aufgebaut werden, wobei ein Schrägstrich („/“) als Trennzeichen erwartet wird.

Weitere Elemente, die beispielsweise Commit-Kommentare enthalten, oder Attribute für Elemente sind im Eingabeformat erlaubt, werden allerdings ignoriert. Fehlen bei einem Commit Änderungsdatum oder Pfade, ist dieser für die Visualisierung nicht relevant und wird ignoriert. Für den Fall, dass bei einem Commit kein Entwicklername angegeben ist, wird dieser Commit einem weiteren Entwickler zugeordnet, dessen Name aus einer leeren Zeichenkette besteht.

Falls ein Entwickler unter verschiedenen Namen in der Eingabedatei vorkommt, wird dieser wie mehrere selbständige Entwickler behandelt. Eine Vereinheitlichung der Namen in der Eingabedatei kann dieses Problem beheben.

Aus den Commits, die in der Log-Datei enthalten sind, kann die gesamte Geschichte des Softwareprojekts extrahiert und die hierarchische Struktur des Softwareprojekts erzeugt werden. Die Hierarchie wird dabei in der Reihenfolge aufgebaut, in der die Commits in der Eingabedatei vorkommen. Dabei müssen die Commits nicht zwingend zeitlich sortiert sein. Je nachdem in welcher Reihenfolge die Commits in der Datei vorkommen (zeitlich aufsteigend, absteigend oder ungeordnet), können unterschiedliche Anordnungen der Knoten in der Hierarchievisualisierung entstehen. Außerdem können allgemeine Informationen, wie die vorkommenden Entwicklernamen und der Zeitbereich, innerhalb welchem alle Commits liegen, ermittelt werden.

Nach dem Laden wird die Hierarchie direkt in der Hierarchieansicht visualisiert; der Zeitbereich wird als Grundeinstellung für den zu visualisierenden WOI River festgelegt, alle Entwicklernamen werden in eine Liste in einem der Dock Widgets eingefügt und eine Entwicklergruppe bestehend aus allen Entwicklern wird festgelegt. Darüber hinaus werden in einem Dock Widget Informationen über die geladenen Daten angezeigt; diese liefern einen Überblick über die Anzahl an Commits und Entwicklern sowie den Zeitbereich und die Anzahl an Knoten in der Projekthierarchie.

4.4 Grundfunktionen

Bevor die Visualisierung eines WOI River erfolgen kann, müssen verschiedene Grundeinstellungen vorgenommen werden. In Kapitel 3.3 wurde bereits auf die verschiedenen Möglichkeiten eingegangen, wie die Höhe einzelner Entwickler in der Visualisierung behandelt werden kann; entsprechende Einstellungen können in einem Dock Widget angepasst werden; durch diese Einstellungen wird festgelegt, wie die visualisierten Daten zu interpretieren sind. Anschließend müssen WOIs in der Projekthierarchie festgelegt werden, die grundlegend für die zu visualisierenden Flüsse sind. Zusätzlich können Entwicklergruppen, der Zeitbereich und die Art der Intervalle angegeben werden.

4.4.1 WOIs festlegen

Die Grundlage zur Festlegung der WOIs bietet die Projekthierarchie. Hier ist der Benutzer in der Lage, interaktiv WOIs festzulegen, um später den WOI River für die entsprechenden Arbeitsbereiche anzeigen zu lassen.

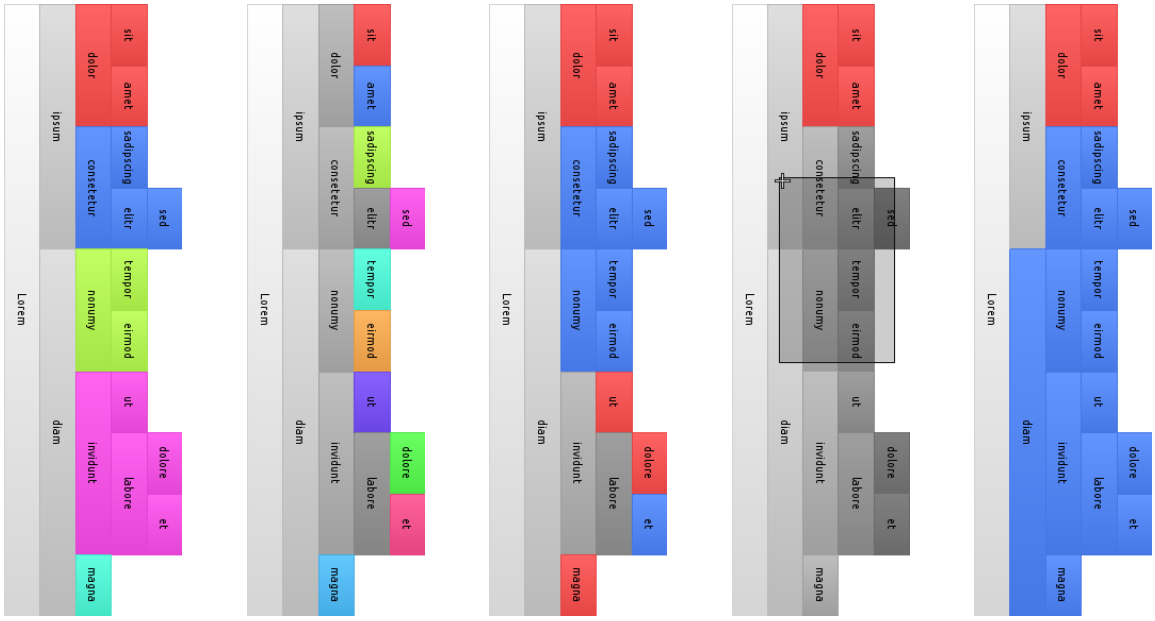


Abbildung 4.3: Unterschiedliche Möglichkeiten für die Festlegung von WOIs: (a) jede Unterhierarchie der Tiefe drei bildet ein WOI; (b) alle Dateien bzw. Blattknoten bilden jeweils ein WOI; (c) mehrere Unterhierarchien an beliebiger Position bilden ein WOI; (d) Aufziehen eines Rechtecks um mehrere benachbarte Unterhierarchien zu einem WOI hinzuzufügen; (e) das Ergebnis von (d): mehrere benachbarte Unterhierarchien bilden ein WOI.

Unter einem WOI sind eine oder mehrere Bereiche (Unterhierarchien) der gesamten Projekthierarchie zu verstehen. Ein WOI kann zwar aus mehreren Unterhierarchien bestehen, mehrere WOIs können sich allerdings nicht überlappen. Diese WOIs können interaktiv in der Hierarchievisualisierung festgelegt werden. Alle betroffenen Knoten der Hierarchiedarstellung in den Unterhierarchien eines festgelegten WOIs werden daraufhin in einer veränderten Darstellungsform (durch Verwendung der zugewiesenen WOI-Farbe) angezeigt.

Für die Festlegung von WOIs stehen mehrere Alternativen zur Verfügung. In Abbildung 4.3 sind einige Möglichkeiten als Beispiele dargestellt. Zum einen ist die Auswahl einzelner Unterhierarchien als WOIs möglich, zum anderen können Funktionalitäten verwendet werden, um einen WOI aus mehreren Unterhierarchien zu erzeugen bzw. um mehrere WOIs gleichzeitig zu bilden.

Per Doppelklick auf einen Knoten in der Hierarchiedarstellung können einzelne Unterhierarchien als WOIs festgelegt oder wieder gelöscht werden. Über das Kontextmenü eines Knotens stehen weitere Funktionen zur Verfügung. Dort können alle Unterhierarchien einer bestimmten Tiefe oder alle Kinderknoten eines ausgewählten Knotens als einzelner WOI festgelegt werden. Außerdem ist es möglich, Unterhierarchien zum zuletzt festgelegten WOI hinzuzufügen oder alle Blattknoten der gesamten Hierarchie als einzelne WOIs zu definieren.

Zusätzlich kann durch Aufziehen eines Bereiches mit der Maus in der Hierarchievisualisierung (vgl. Abbildung 4.3d) ein Teil der Hierarchie als ein neuer WOI festgelegt werden. Hierbei werden nur die Knoten berücksichtigt, die noch zu keinem weiteren WOI gehören. Es wird jeweils dafür gesorgt, dass alle Knoten, die den Bereich schneiden zu dem neuen WOI hinzugefügt werden.

Eine weitere Funktion ist, einen WOI aus allen Dateien und/oder Ordnern zu erstellen, die eine bestimmte Zeichenkette enthalten. Hierfür wird bei allen Knoten geprüft, ob die Zeichenkette in deren Namen enthalten ist. Falls dies zutrifft wird die Teilhierarchie, die durch den jeweiligen Knoten festgelegt ist, zu einem neuen WOI hinzugefügt und in der Unterhierarchie des Knotens muss keine weitere Prüfung erfolgen. Die Festlegung kann entweder nur für Dateien oder zusätzlich für Verzeichnisse durchgeführt werden.

Außerdem können für eine festgelegte Anzahl an vorkommenden Dateiendungen neue WOIs erzeugt werden. Da in der Verzeichnisstruktur des Projektes verschiedene Dateiarten vorkommen, könnte eine Festlegung der WOIs in Abhängigkeit von Dateiendungen interessant sein, um herauszufinden, welche Arten von Dateien über die Zeit am meisten und von welchen Entwicklern bearbeitet wurden. Hierfür werden zunächst alle Dateiendungen, die in der gesamten Struktur vorkommen, ermittelt. Indem bei jedem Blattknoten geprüft wird, ob eine Unterteilung des Namens in einen Teil vor und nach einem Punkt möglich ist, kann ggf. der Teil nach dem letzten Punkt als Dateiendung verwendet werden. Nachfolgend wird ermittelt, welches die am häufigsten vorkommenden Dateiendungen sind und es werden für die M häufigsten Dateiendungen neue WOIs angelegt.

4.4.2 Entwicklergruppen festlegen

In der Grundeinstellung der Anwendung werden alle Entwickler, die mindestens einen Commit-Beitrag geleistet haben, bereits in eine aktive Entwicklergruppe zusammengefasst und werden für die Visualisierung des WOI Rivers berücksichtigt. Häufig interessieren allerdings nicht alle Entwickler, die an einem Softwareprojekt beteiligt sind, sondern nur bestimmte Entwickler oder Entwicklergruppen, z.B. die Entwickler, die am meisten zum Projekt beigetragen haben bzw. die an einem bestimmten WOI arbeiten oder auch spezielle einzelne Entwickler. Bei großen Projekten gibt es teilweise Entwickler, die nur ein Mal oder sehr selten Änderungen bewirkt haben; diese können für Rauschen sorgen und sollten unter Umständen herausgefiltert werden.

Zu diesem Zweck bietet die Anwendung die Möglichkeit, einzelne Entwicklergruppen festzulegen und nur diese für die Visualisierung zu berücksichtigen. In einem Dock Widget wird eine Liste aller Entwickler, die Dateiänderungen für das Projekt committet haben, bereitgestellt. Diese Liste kann alphabetisch oder in Abhängigkeit der Anzahl der Änderungen der einzelnen Entwickler sortiert werden. Im ersten Fall können bestimmte Entwickler schnell gefunden werden; im zweiten Fall kann leicht herausgefunden werden, wer ein Hauptentwickler ist und wer eher weniger zum Projekt beigetragen hat. Aus dieser Liste können Entwickler ausgewählt werden und als Gruppe mit einem benutzerdefinierten Namen abgespeichert werden.

Außerdem besteht die Möglichkeit, dass alle Entwickler, die an einem Fluss beteiligt sind einer neuen Gruppe zugeordnet werden (über das Kontextmenü eines Flusses erreichbar).

Anschließend besteht die Möglichkeit, eine oder mehrere Entwicklergruppen zu selektieren, um alle darin enthaltenen Entwickler für die WOI River-Visualisierung zu berücksichtigen.

4.4.3 Zeitbereich und Intervalle festlegen

In der Benutzeroberfläche kann ein Zeitbereich für die zu visualisierenden Daten angegeben werden sowie die Art der Intervallerzeugung.

Zeitbereich

Standardmäßig ist für die Visualisierung der Zeitbereich vom ersten bis zum letzten Commit ausgewählt, sodass alle Commits der Eingabedatei in der Visualisierung berücksichtigt werden können. Falls nur ein bestimmter Zeitabschnitt interessiert, kann dieser individuell festgelegt werden. Hierdurch kann beispielsweise bei einem Entwicklungsprozess, der schon länger andauert, die Visualisierung auf die ersten Jahre beschränkt werden oder es kann bei mehreren Projekten, die parallel laufen, der gleiche Zeitbereich gewählt werden, um die Ergebnisse besser miteinander vergleichen zu können. Zudem ist es dadurch möglich, ohne eine Erhöhung der Anzahl der verwendeten Intervalle trotzdem eine feinere Aufteilung des Zeitbereichs in einzelne Intervalle zu erhalten.

Intervalle

Es gibt zwei verschiedene Arten Intervalle festzulegen: Entweder durch Zeitintervalle, die jeweils eine bestimmte Zeitspanne abdecken oder durch Commit-Intervalle, die eine bestimmte Anzahl an Commits beinhalten.

Durch Angabe der Anzahl an Zeitintervallen, in die der angegebene Zeitraum eingeteilt werden soll bzw. welche Zeitspanne durch ein Intervall repräsentiert werden soll, können Intervalle in Bezug auf die Zeit festgelegt werden. Für die Bildung der Zeitintervalle werden alle Commits verwendet, die in die entsprechende Zeitspanne fallen.

Des Weiteren ist es möglich, Intervalle unabhängig von der Zeit zu bilden, indem nur Commits berücksichtigt werden. Gibt es beispielsweise eine größere Pause in einem Entwicklungsprozess, hat dies bei einer Unterteilung in Zeitintervalle zuerst viele Abflüsse und danach viele Zuflüsse zur Folge. Wenn die Intervalle zeitunabhängig gebildet werden, kann der angegebene Zeitbereich normalisiert werden. Durch Angabe der Anzahl an Commits, die in ein Intervall fallen sollen bzw. die Anzahl der Commit-Intervalle, die gebildet werden sollen, werden Intervalle erzeugt, die nach Möglichkeit jeweils die gleiche Anzahl an Commits beinhalten.

4.5 Weitere Funktionen und Interaktion

Sowohl die Hierarchiedarstellung als auch die WOI River-Visualisierung bietet verschiedene Einstellungs- und Interaktionsmöglichkeiten. Im Folgenden wird zuerst auf die der Hierarchiedarstellung eingegangen, danach auf die der WOI River-Visualisierung.

4.5.1 Hierarchie

Die Hierarchie bietet die Grundlage zur Festlegung von WOIs. Zusätzlich bietet sie verschiedene Interaktionsmöglichkeiten; in Abbildung 4.4 sind einige anhand von Beispielen dargestellt.

Allgemeine Informationen

In einem Dock Widget werden allgemeine Informationen bezüglich der Hierarchie angezeigt. Diese beinhalten die Anzahl der Knoten der Hierarchie, die Anzahl der Blattknoten (d.h. der Dateien/leeren Verzeichnisse) und die maximale Tiefe.

Selektion

Ein einzelner Knoten der Hierarchie kann ausgewählt werden, wodurch in einem Dock Widget weitere Informationen über seine Position innerhalb der Hierarchie angezeigt werden. Dies beinhaltet seinen Pfad, seine Tiefe, die Anzahl seiner Kinder-, Unter- und Blattknoten und den beinhaltenden WOI. Selektierte Knoten werden zusätzlich etwas dunkler visualisiert als andere Knoten mit ähnlichen Eigenschaften (d.h. Knoten gleicher Tiefe bzw. Knoten die zum gleichen WOI gehören) innerhalb der Hierarchie.

Außerdem gibt es, falls der Knoten zu einem WOI gehört, eine Verbindung zum WOI River (mehr dazu in Abschnitt 4.5.2).

Zoomen

Die Hierarchie unterstützt sowohl geometrisches als auch semantisches Zoomen [Spe07]. Beim geometrischen Zoomen wird die Darstellung der Hierarchie nicht verändert, sondern nur der angezeigte Ausschnitt vergrößert oder verkleinert. Beim semantischen Zoomen können auch angezeigte Formen und Details verändert werden oder Objekte verschwinden bzw. hinzukommen.

Der erste Fall wird hier durch Veränderung des dargestellten Ausschnittes der Hierarchie umgesetzt (Abbildung 4.4d). Es ist somit möglich, einzelne Bereiche vergrößert anzuzeigen oder eine Übersicht über einen größeren Bereich der Darstellung zu erhalten. Zusätzlich kann die Höhe bzw. Breite der Abbildung beliebig in x- und y-Richtung gestreckt werden, um beispielsweise weniger Platz für die Darstellung zu benötigen oder um Labels besser anzeigen zu können.

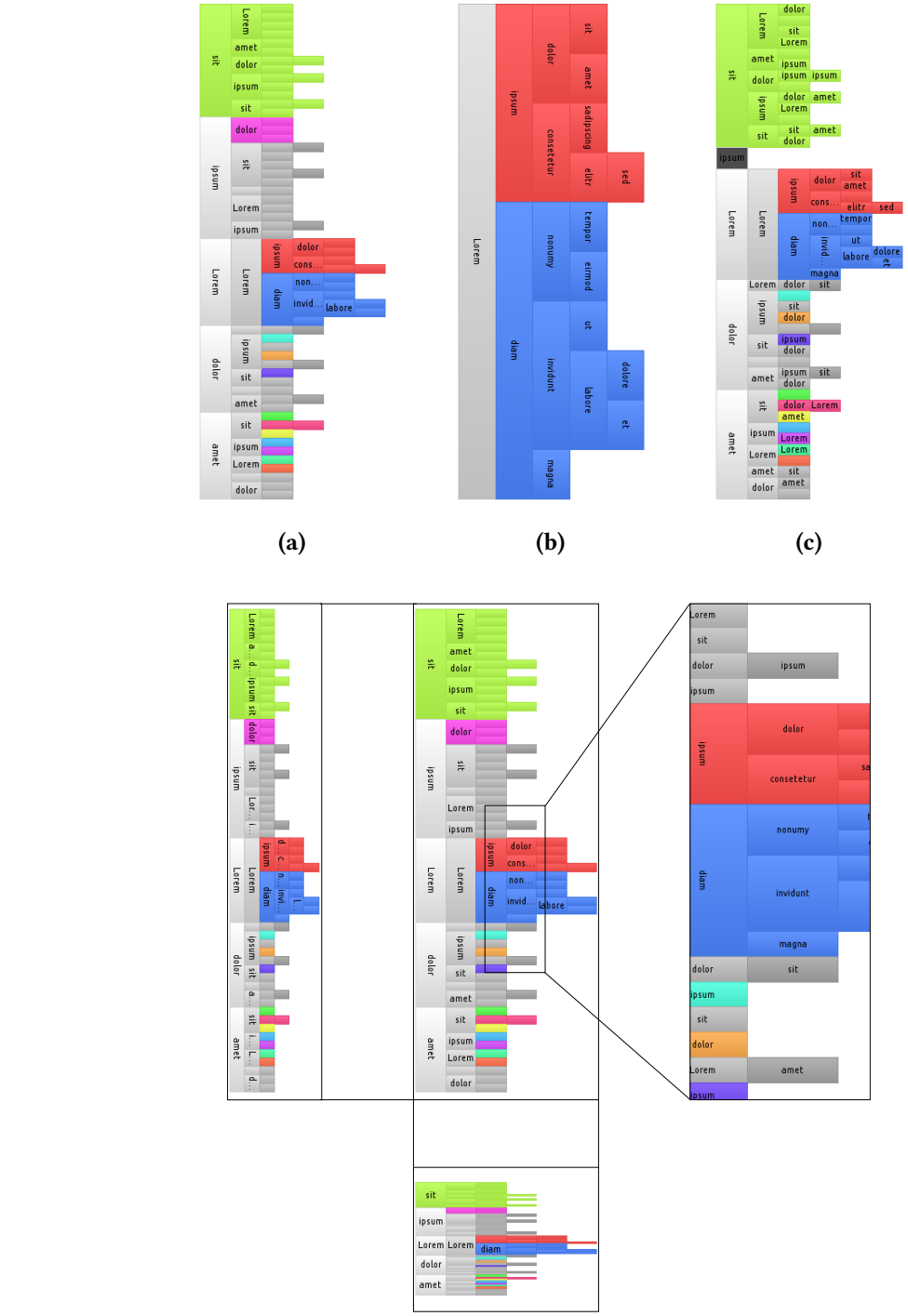


Abbildung 4.4: Interaktionsmöglichkeiten in der Hierarchiedarstellung am Beispiel der Layered Icicle-Darstellung: (a) Hierarchiedarstellung ohne Interaktion; (b) semantisches Zoomen; (c) Aggregation einer Unterhierarchie; (d) geometrisches Zoomen (rechts) sowie veränderte Skalierung in x-Richtung (links) und y-Richtung (unten).

Der zweite Fall wird durch das Anzeigen einer Teilhierarchie realisiert, indem die restlichen Knoten ausgeblendet werden (Abbildung 4.4b). Besonders bei sehr großen Hierarchien ist dadurch eine Konzentration auf bestimmte Bereiche in der Hierarchie möglich.

Navigation

Falls nur eine Teilhierarchie angezeigt wird, kann innerhalb der Hierarchie navigiert werden. Zum einen kann eine Ebene aufgestiegen werden, d.h. der zugehörige Teilbaum des Elternknotens wird dann angezeigt, zum anderen kann zu bereits angezeigten Teilhierarchien zurück navigieren werden.

Aggregation

Außerdem können Teile der Hierarchie aggregiert werden, wodurch Kinderknoten eines ausgewählten Knotens ausgeblendet werden und der Wurzelknoten der aggregierten Teilhierarchie evtl. kleiner dargestellt wird; weniger relevante Bereiche der Hierarchie können so ausgeblendet werden. Ein Beispiel mit einer aggregierten Teilhierarchie ist in 4.4c dargestellt.

Labels

Labels können optional angezeigt werden und enthalten den Namen einzelner Knoten, d.h. des Datei- bzw. Ordnernamens; ihre Schriftgröße ist individuell einstellbar. Beim Zoomen können teilweise weitere Labels sichtbar werden, da sich die Größe der Knotenrepräsentationen in der Ansicht verändert und somit unter Umständen zusätzlicher Platz verfügbar wird (vgl. Abbildung 4.4d).

Tooltips

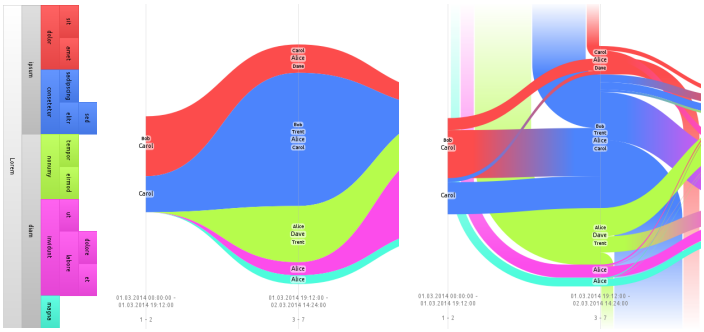
Der Tooltip eines Hierarchieknotens enthält den Namen der zugehörigen Datei bzw. des zugehörigen Ordners. Falls nicht genügend Platz für die Anzeige eines Labels zu Verfügung steht, können Tooltips verwendet werden, um dennoch herauszufinden, welcher Knoten an einer bestimmten Position abgebildet ist.

4.5.2 WOI River

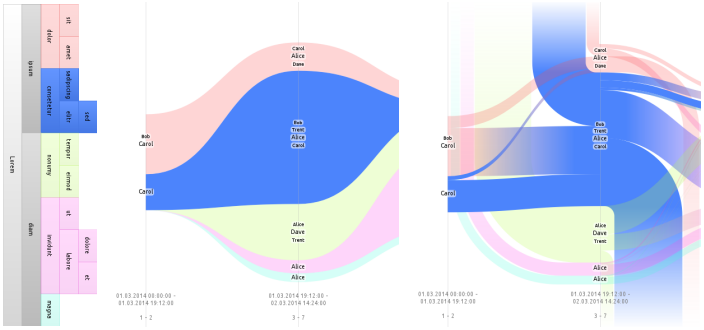
Eine Übersicht verschiedener Funktionen und Interaktionsmöglichkeiten der WOI River-Ansicht sind in den Abbildungen 4.5 und 4.6 dargestellt.

Allgemeine Informationen

Auch zum WOI River werden allgemeine Informationen in einem Dock Widget angezeigt. Diese beinhalten Informationen zu der Anzahl angezeigter Flüsse, Intervalle, beteiligter Entwickler, verwendeter Commits und Dateiänderungen sowie dem dargestellten Zeitbereich.



(a)



(b)

Abbildung 4.5: Darstellung eines Beispieldatensatzes; jeweils Hierarchiedarstellungen und sichtbare Ausschnitte in der Benutzeroberfläche in ThemeRiver-Ansicht und in Detailansicht: (a) Normalansicht; (b) mit hervorgehobenem WOI.

Detailansicht

Die Ansicht der einzelnen Übergangsbereiche kann zwischen einer ThemeRiver-Darstellung und einer Detaildarstellung gewechselt werden (Abbildung 4.5 in der Mitte und rechts). Der Wechsel wird per Doppelklick auf einen Übergangsbereich erreicht. Dies hat den Effekt des „Auf- bzw. Zuklappens“; in die Detailansicht sind Details bezüglich des Verhaltens der Entwickler durch Transitionen und Zu- bzw. Abflüsse sichtbar; in der ThemeRiver-Ansicht ist nur eine Übersicht zu erkennen.

Außerdem besteht die Möglichkeit, dass alle Bereiche gleichzeitig auf- bzw. zugeklappt werden, wodurch der gesamte Fluss als ThemeRiver bzw. in der Detailansicht angezeigt wird.

Hervorhebung

Ein WOI kann in der Visualisierung hervorgehoben werden, indem der Mauszeiger auf einen Fluss bewegt wird oder indem ein Fluss, durch Markieren mit der Maus, ausgewählt wird. In beiden Fällen werden alle WOIs transparent dargestellt, bis auf den zugehörigen WOI des ausgewählten Flusses (Abbildung 4.5b). Diese Änderung in der Visualisierung hat sowohl Auswirkungen auf den WOI River als auch auf die Hierarchiedarstellung. Hiermit ist es für den Benutzer möglich, per „brushing“ und „linking“ [BMMS91] eine Verbindung zwischen der WOI River- und Hierarchievisualisierung

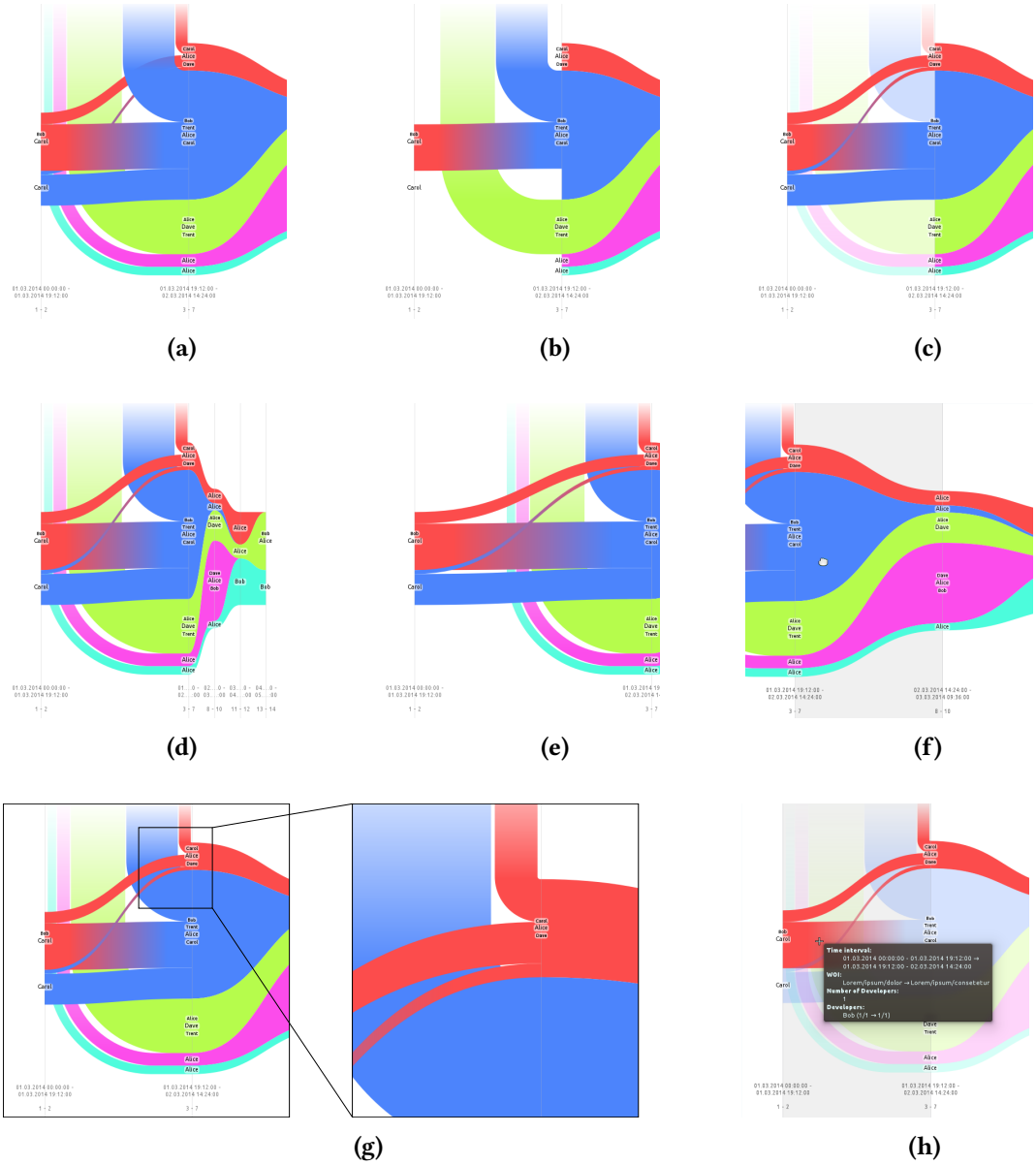
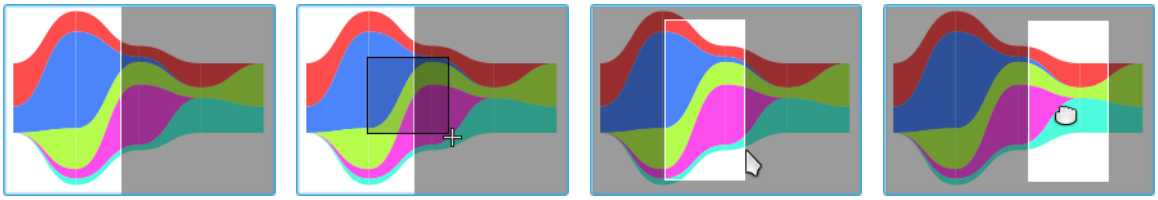


Abbildung 4.6: Veranschaulichung ausgewählter Funktionen bzw. Interaktionsmöglichkeiten in der WOI River-Ansicht: (a) Fluss im Vordergrund, (b) Ausblenden verschiedener Übergänge bis zu einem festgelegten Schwellwert, (c) transparente Zuflüsse, (d) Breite der Übergangsbereiche der ThemeRiver-Ansicht verkleinert, (e) Breite der Übergangsbereiche der Detailansichten vergrößert, (f) Panning, (f) Zoomen und (h) Anzeige eines Tooltips



(a) (b) (c) (d)

Abbildung 4.7: Veranschaulichung der Funktionen der Übersicht: (a) Übersicht des sichtbaren Ausschnitts der Hauptdarstellung; (b) Bereich festlegen; (c) sichtbarer Bereich nach Festlegen eines Bereichs und (d) Verschieben des sichtbaren Bereichs.

aufzubauen, da in beiden Ansichten Elemente, die zu dem entsprechenden WOI gehören, hervorgehoben werden. Hiermit kann eine schnelle Verknüpfung zwischen dem dargestellten Fluss und den Arbeitsbereichen in der Hierarchie erfolgen. Optional kann die Hervorhebung für den Fall, dass der Mauszeiger auf einen Fluss bewegt und das entsprechende WOI hervorgehoben wird, deaktiviert werden.

Entsprechende Auswirkungen werden auch bei der Hierarchiedarstellung erreicht, wenn ein Knoten, der zu einem WOI gehört, ausgewählt wird oder sich die Maus über einem solchen Knoten befindet.

Selektion

Zusätzlich zu der Veränderung in der Darstellung, werden nach einer Markierung eines Flusses (bzw. eines Knotens in der Hierarchievisualisierung) mit der Maus in einem Dock Widget Informationen bezüglich des zugehörigen WOIs angezeigt.

WOI in Vordergrund

Falls ein einzelner Fluss genauer untersucht werden soll, kann dieser auch in den Vordergrund gesetzt werden. Dies bewirkt, dass er von keinen dünneren Übergängen überdeckt wird und sein Verlauf besser untersucht werden kann (Abbildung 4.6a).

Transitionen, Zuflüsse und Abflüsse filtern

Die angezeigten Transitionen und Zu- bzw. Abflüsse können nach verschiedenen Kriterien gefiltert dargestellt werden. Hierbei gibt es die Möglichkeit, sie entweder ganz oder bis zu einen bestimmten Schwellwert auszublenden. Indem sie nur ab einem bestimmten Wert angezeigt werden, können dünnere, evtl. störende Transitionen ausgeblendet werden. Zudem ist es möglich Transitionen transparent darzustellen; weniger interessante Transitionen verschwinden dadurch nicht ganz, gelangen aber etwas in den Hintergrund. Da die Visualisierungen teilweise sehr stark von Zu- und Abflüssen geprägt sind, könnte es hier sinnvoll sein, diese transparent darzustellen um sie nicht ganz aus dem

Gesichtsfeld zu verlieren und den Schwerpunkt auf die restlichen Übergänge zu legen. Beispiele sind in Abbildung 4.6b und 4.6c abgebildet.

Die beschriebenen Einstellungen können getrennt für Transitionen zwischen gleichen WOIs, Transitionen zwischen verschiedenen WOIs und Zu- und Abflüssen vorgenommen werden.

Breite der Übergangsbereiche

Da die Zu- und Abflüsse horizontal und vertikal mindestens die gleiche Breite besitzen müssen, können diese nicht getrennt in x- und y-Richtung skaliert werden; bei der ThemeRiver-Visualisierung spielen die Proportionen in x- und y-Richtung keine Rolle. Um den dargestellten Bereich dennoch in x-Richtung zu skalieren, kann die Breite der Übergangsbereiche sowohl in der ThemeRiver-Ansicht als auch in der Detailansicht unabhängig voneinander eingestellt werden (Abbildungen 4.6d und 4.6e). Hierdurch kann eine Visualisierung mit vielen Intervallen verkleinert werden. Allerdings benötigen Zu- und Abflüsse teilweise eine minimale Breite (maximal die Höhe des Flusses und einen geringen Abstand zwischen jedem Zu-/Abfluss) wodurch diese Funktion nur eingeschränkt verwendet werden kann.

Zoomen

Wie auch schon bei der Hierarchiedarstellung, kann in der Ansicht gezoomt werden (Abbildung 4.6g), wodurch mehr Details erkennbar werden und evtl. weitere Labels sichtbar werden können oder der gesamte Verlauf des Flusses angezeigt werden kann. Die Visualisierung kann hierbei jedoch nicht getrennt in x- und y-Richtung skaliert werden, da, wie bereits beschrieben, bei Zu- und Abflüssen sowohl die Höhe als auch Breite entscheidend ist.

Panning

In dem sichtbaren Ausschnitt der WOI River-Ansicht kann meist nicht der gesamte WOI River mit den gewünschten Einstellungen angezeigt werden. Während die ThemeRiver-Darstellung in x-Richtung beliebig skaliert werden kann, benötigt die Detailansicht häufig eine Mindestbreite. Durch zoomen kann zwar die gesamte Visualisierung angezeigt werden, allerdings wird der Fluss so klein, dass keine Details mehr zu erkennen sind. Durch Verschieben des sichtbaren Bereichs („Panning“), können die Bereiche der Visualisierung sichtbar gemacht werden, die sich momentan außerhalb des dargestellten Bereichs befinden um diese weiter zu untersuchen (Abbildung 4.6h).

Übersicht

In einem Dock Widget befindet sich eine Miniatur-Übersicht des dargestellten WOI Rivers in Form eines ThemeRivers. In ihr ist immer erkennbar, welcher Ausschnitt momentan in der Hauptdarstellung sichtbar ist. Wenn gezoomt wird oder ein anderer Ausschnitt in der WOI River-Ansicht angezeigt wird, wird die Übersicht entsprechend angepasst. Auch wenn Bereiche der Hauptansicht aufgeklappt sind,

wird der entsprechende Bereich in der zugehörigen ThemeRiver-Visualisierung erkennbar. Hierdurch ist ein Rückschluss vom sichtbaren Ausschnitt auf die gesamte Darstellung möglich und es wird dadurch den Erhalt der Mental Map unterstützt.
Zusätzlich ist es möglich, in der Übersicht den sichtbaren Bereich zu verschieben oder durch Auswahl eines Bereichs neu festzulegen, um schnell an eine gewünschte Stelle in der Visualisierung zu gelangen. Die Funktionen der Übersicht sind in Abbildung 4.7 veranschaulicht.

Tooltips

Für die einzelnen Elemente, aus denen der WOI River aufgebaut ist (Transitionen, Zuflüsse, Abflüsse und Elemente des ThemeRivers zwischen zwei Intervallen) können jeweils Tooltips angezeigt werden (vgl. Abbildung 4.6f). Je nach Typ, können der zugehörige Zeitbereich, der Name des (bzw. der) zugehörigen WOIs, die Anzahl der Entwickler und zehn Entwickler mit der größten Beteiligung angezeigt werden. Bei der Anzahl der Entwickler wird zusätzlich angezeigt, wie viele Entwickler den entsprechende WOI/die entsprechenden WOIs in beiden angrenzenden Intervallen bearbeitet haben und bei den Entwicklern ist angegeben, wie viele Dateien sie in beiden Intervallen verändert haben.

Außerdem besitzen die Intervall-Linien Tooltips, in denen Informationen über das Intervall, die Anzahl der Bearbeiter und der veränderten Dateien sowie zehn Entwickler, welche die meisten Änderungen in dem Intervall bewirkt haben, angegeben sind.

Labels

Es gibt verschiedene Arten von Labels, die optional angezeigt werden können (die verschiedenen Varianten sind im Kapitel 3.3.5 beschrieben); die Schriftgröße kann jeweils variiert werden.

Hilfslinien

Ferner können in der Visualisierung Intervall-Linien und eine Mittellinie angezeigt werden.

4.5.3 Weitere Visualisierungsparameter

Bei jedem Festlegen eines neuen WOIs bekommt dieser eine neue Farbe zugewiesen. Wenn zwischendurch andere WOIs wieder gelöscht werden, wird deren Farbe in den Visualisierungen nicht mehr verwendet. Daher besteht die Möglichkeit, alle Farben neu zu vergeben, wodurch alle WOIs neue Farben erhalten, auch wenn sie manuell festgelegt wurden.

Die Anordnung, in der WOIs im WOI River angezeigt werden, kann entweder in der Reihenfolge erfolgen, in der die einzelnen WOIs ausgewählt wurden oder so wie sie in der Hierarchie vorkommen. Beim zweiten Fall werden die Flüsse so angeordnet, dass WOIs, die weiter oben in der Visualisierung der Hierarchie vorkommen auch weiter oben in der WOI River-Visualisierung gezeichnet werden. Hiermit kann eine einfache Verknüpfung zwischen den beiden Visualisierungen für den Benutzer erfolgen, auch wenn gleiche oder ähnliche Farben bei verschiedenen WOIs verwendet werden. Besteht

ein WOI aus mehreren Teilhierarchien, wird jeweils die Position der Teilhierarchie verwendet, die am weitesten oben in der Visualisierung gezeichnet wird.

4.6 Export

Es gibt mehrere Möglichkeiten zum Export der Visualisierungen und der zugehörigen Daten. Zum einen können die Visualisierungen im PNG-Format gespeichert werden, zum anderen besteht die Möglichkeit Informationen bezüglich der Einstellungen und der visualisierten Daten zu speichern sowie benutzerdefinierte Ausschnitte der Daten abzuspeichern.

4.6.1 PNG-Export

Die dargestellten Visualisierungen können als PNG-Datei exportiert werden. Dabei gibt es sowohl für die Hierarchie als auch den WOI River die Möglichkeit die gesamte Visualisierung zu exportieren oder nur den aktuell sichtbaren Bereich. Hierbei werden in den Dateien Metadaten bezüglich der verwendeten Einstellungen der Anwendung, die für die Visualisierung relevant sind, gespeichert.

4.6.2 Einstellungen und allgemeine Informationen

Weiterhin gibt es die Möglichkeit eine Textdatei mit Informationen bezüglich des aktuell geladenen Datensatzes und der Visualisierungen zu speichern. Diese enthalten allgemeine Informationen bezüglich der geladenen Daten (Name der geladenen Datei, Anzahl der Commits, Anzahl der Entwickler, Zeitpunkt des ersten und letzten Commits, Anzahl der Knoten und Blattknoten in der Hierarchie sowie die maximale Tiefe der Hierarchie) und des dargestellten WOI Rivers (Zeitintervall der dargestellten Daten, Anzahl der Flüsse sowie Anzahl der berücksichtigten Commits und Entwickler). Zusätzlich kann eine Datei mit den aktuellen Einstellungen bzw. Festlegungen (Einstellungen in den Dock Widgets und festgelegte WOIs) der angezeigten Visualisierungen exportiert werden um die Visualisierungen rekonstruieren zu können.

4.6.3 Benutzerdefinierte Ausschnitte der Daten

Darüber hinaus können benutzerdefinierte Ausschnitte der Daten für eine spätere Analyse abgespeichert werden. Teilweise sind die zu visualisierenden Daten recht groß und enthalten Informationen über einen längeren Zeitraum von vielen Entwicklern und einer großen Projekthierarchie. Interessiert für einen langen Zeitraum, zu vielen Entwicklern und einer großen Projekthierarchie nur ein bestimmter Zeitabschnitt, bestimmte Entwickler oder nur ein Teil der Projekthierarchie, können die entsprechenden relevanten Daten exportiert werden, die dann wiederum in der Anwendung geladen werden können um nur die gefilterten Daten zu visualisieren.

5 Fallstudien

In diesem Kapitel werden drei Fallstudien anhand verschiedener Open-Source-Softwareprojekte durchgeführt. Ziel ist dabei jeweils, einen guten Einblick in den Softwareentwicklungsprozess zu erhalten und Besonderheiten oder Trends zu erkennen.

Mehrere Fragestellungen können hierbei untersucht werden – zum einen in Bezug auf die Anzahl der Entwickler, die an den Projekten arbeiten und zum anderen in Bezug auf die Anzahl der Änderungen (im Folgenden in Form von Dateiänderungen berücksichtigt), die von den Entwicklern bewirkt werden.

Als erstes Projekt wird die Programmiersprache Python untersucht. Hierbei wird zusätzlich mit den Ergebnissen der Evolution Storylines-Visualisierung verglichen. Danach wird die Entwicklung der Bibliothek libvpx für die Videocodecs VP8/VP9 betrachtet sowie die des Linux-Kernels. Der Linux-Kernel zählt als recht großes Projekt; es wird geprüft, ob die Visualisierungstechnik auch für sehr große Projekte skalierbar ist und es noch möglich ist, durch die Visualisierung Einsichten zu erhalten.

Ausschlaggebend für eine sinnvolle Untersuchung des Entwicklungsverhaltens ist eine geeignete Wahl der WOIs. Zum einen sollten nicht zu viele WOIs gewählt werden, da die einzelnen Flüsse so immer schmaler werden, ähnliche Farben bei unterschiedlichen Flüssen verwendet werden und es sehr starke Wechsel zwischen den WOIs geben kann. Zum anderen sollte aber auch versucht werden, möglichst viele Bereiche zu berücksichtigen um keine wichtigen Veränderungen zu übersehen. Bei den folgenden Analysen wurde versucht, Hauptordner der Projekthierarchie als WOIs zu verwenden und die restlichen Dateien in einem zusätzlichen WOI zusammenzufassen, um jeden Entwickler und jede Änderung an dem jeweiligen Projekt zu berücksichtigen.

5.1 Python

Als erstes Open-Source-Projekt wird die Programmiersprache Python [Pyta] gewählt. Die Entwicklung von Python begann 1990 und dauert bis heute an, es liegt somit die Geschichte des Entwicklungsprozesses aus etwa 24 Jahren vor. Zum Zeitpunkt der Untersuchung (Februar 2014) besteht der verwendete Datensatz aus 88950 Commits von insgesamt 184 Entwicklern. Die Projekthierarchie umfasst 10865 Knoten mit 10108 Dateien und einer maximalen Tiefe von zehn. Die Eingabedaten wurden aus dem Mercurial-Repository [Pytb] erzeugt.

Dieses Projekt wurde bereits in den Evolution Storylines [OM10] und in code_swarm [OM09] analysiert; das Ergebnis von der Evolution Storylines-Visualisierung kann in Abbildung 5.1 zum Vergleich betrachtet werden. Dort wurde der Entwicklungsprozess nur bis zum Jahr 2006 berücksichtigt. Unsere

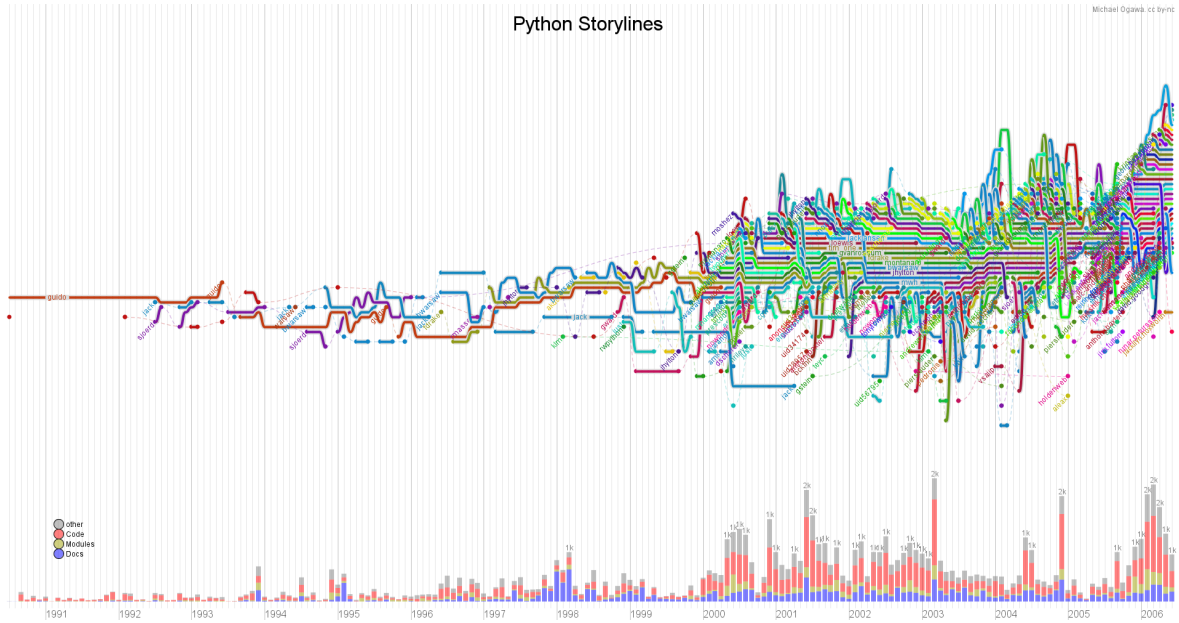


Abbildung 5.1: Visualisierung des Entwicklungsprozesses der Programmiersprache Python von 1990 bis 2006 mittels der Evolution Storylines [OM10].

Visualisierung wird mit dem Ergebnis der Evolution Storylines-Visualisierung verglichen, um zu sehen, ob die gleichen Besonderheiten erkennbar sind.

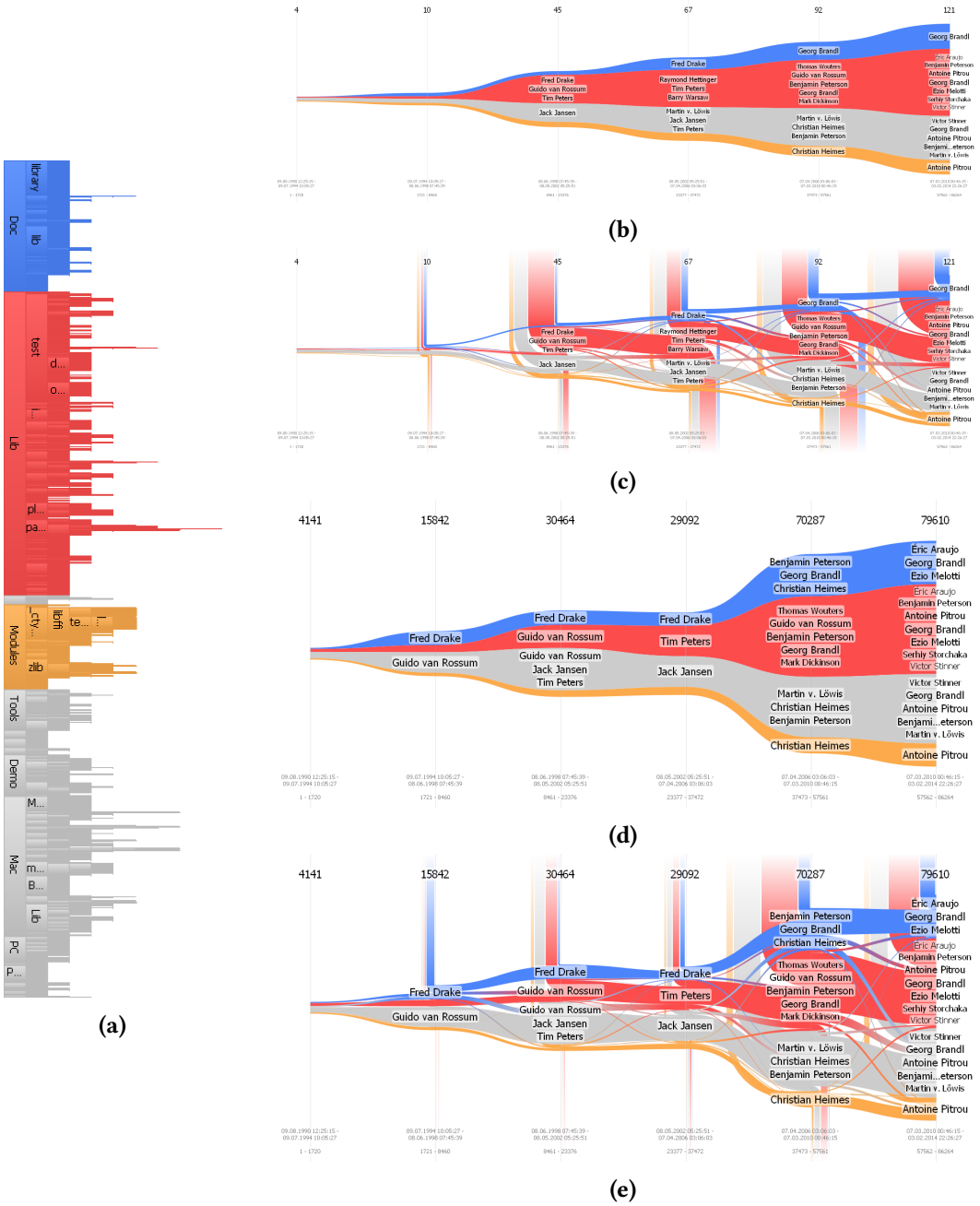
Im Folgenden werden vier WOIs für die Betrachtung gewählt: Die Ordner *Doc* (blau), *Lib* (rot), *Modules* (orange) und alle restlichen Dateien (grau). Hierbei wird versucht, die gleichen Bereiche festzulegen, die auch bei den Evolution Storylines gewählt wurden.

Bei den Evolution Storylines wurde unter anderem festgestellt, dass Guido van Rossum das Projekt 1990 anfängt und erst zwei Jahre später vereinzelt weitere Entwickler hinzukommen. Hierbei ist erkennbar, wie Entwickler teilweise zusammenarbeiten. Außerdem kommen in den Jahren 2000 und 2005 viele weitere Entwickler hinzu.

Bei den Evolution Storylines erfolgt die Aufteilung monatsweise; dies würde bei dem hier betrachteten Zeitraum von etwa 24 Jahren zu einer Aufteilung in ca. 288 Intervalle führen. Da viele der Transitionsbereiche aufgrund der Zu- und Abflüsse eine Mindestbreite besitzen müssen, würden hierdurch sehr breite Visualisierungen entstehen, welche zur Untersuchung eher weniger geeignet wären, da der vollständige Zeitbereich nicht auf einmal betrachtet werden kann. Im Folgenden werden nur sechs Intervalle gebildet, deshalb werden die Daten innerhalb der einzelnen Abschnitte stark aggregiert.

In Abbildung 5.2 ist die Hierarchie zusammen mit der Übersicht und der Detailansicht des WOI Rivers zu sehen, jeweils für die Darstellung in Abhängigkeit der Anzahl an Entwicklern (5.2b und 5.2c) und in Abhängigkeit der veränderten Dateien (5.2d und 5.2e).

Sowohl bei den Darstellungen in Abhängigkeit der Entwicklerzahl als auch bei den Visualisierungen in Abhängigkeit der Anzahl bearbeiteter Dateien ist erkennbar, dass die Beteiligung am Projekt monoton wächst. Es gibt in jedem Intervall recht viele neue Entwickler, die hinzukommen; etwas



(a) (b) (c) (d) (e)

Abbildung 5.2: Softwareentwicklungsverlauf von Python: (a) Projekthierarchie; (b) und (c): Darstellung der Anzahl der beteiligten Entwickler als Übersicht und Detailansicht; (d) und (e): Darstellung der Anzahl an veränderten Dateien als Übersicht und Detailansicht.

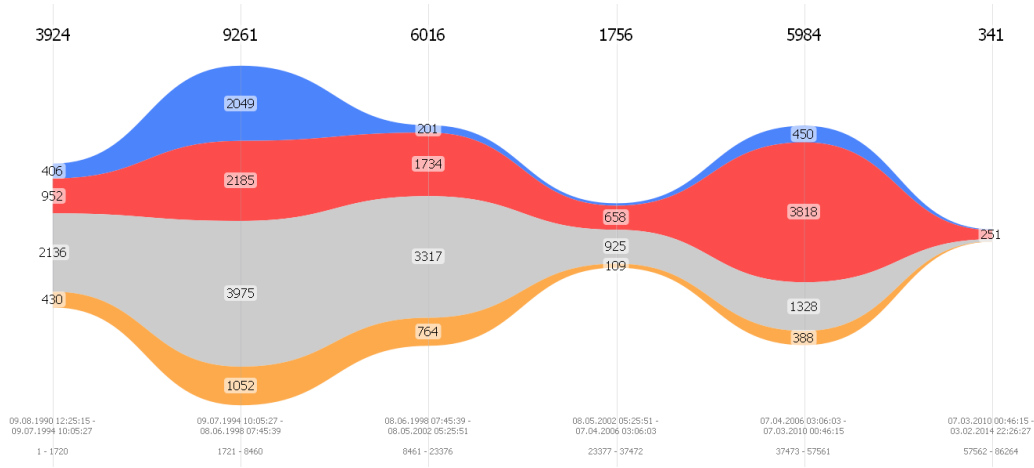


Abbildung 5.3: WOI River des Python-Entwicklungsprozesses für Guido van Rossum.

weniger verlassen das Projekt wieder. Die Entwickler, die das Projekt verlassen, haben nicht viele Dateien verändert, da in der unteren Teilabbildung nur sehr wenige Abflüsse zu sehen sind.
Am Anfang bearbeiten nur sehr wenige Entwickler das Projekt. Aufgrund der hier gewählten Intervalleinteilung kann allerdings nicht festgestellt werden, ob dies nur Guido van Rossum ist; bei einer feineren Einteilung kann dies jedoch auch herausgefunden werden. Das gleiche gilt für die Feststellung bei den Evolution Storylines, dass viele Entwickler in den Jahren 2000 und 2005 hinzukamen; die hier gezeigten Visualisierungen erwecken eher den Eindruck, dass in jedem Intervall jeweils viele neue Entwickler hinzukommen. Nur bei Betrachtung der Abbildungen in Abhängigkeit der veränderten Dateien kann erkannt werden, dass im zweiten Drittel die Beteiligung durch sehr viele Zuflüsse stark zunimmt.

Bis jetzt wurden die Beiträge aller Entwickler untersucht. Da Guido van Rossum das Projekt vor vielen Jahren gestartet hat, könnte interessant sein, herauszufinden, wie er im Laufe dieser Zeit an dem Projekt mitgewirkt hat. In Abbildung 5.3 ist eine ThemeRiver-Ansicht für die Anzahl der durch ihn veränderten Dateien dargestellt. In dieser Visualisierung fällt auf, dass Guido van Rossum zunächst immer mehr zum Projekt beigetragen hat, danach wieder weniger. Im letzten Intervall hat er kaum noch Dateien verändert. Er beteiligt sich in jedem der ausgewählten Arbeitsbereiche, wobei er bei *Doc* und *Lib* eher weniger Änderungen vornimmt.

Als nächstes wird noch eine weitere Festlegung der WOIs gewählt: die zehn am häufigsten vorkommenden Dateiendungen; in Abbildung 5.4 sind die zugehörigen WOI Rivers dargestellt: die Höhe des WOI Rivers bezieht sich auf die Anzahl der veränderten Dateien.

In den Abbildungen 5.4b und 5.4c sieht man, dass am häufigsten Python-Dateien (*.py) bearbeitet wurden, gefolgt von C-Dateien (*.c), TEX-Dateien (*.tex), RST-Dateien (*.rst) und H-Dateien (*.h). Im zweiten Drittel des Entwicklungsprozesses ist in der Übersichts-Darstellung hierbei etwas Interessantes zu sehen: der rosarote Bereich (TEX-Dateien) löst sich auf, während der grüne Bereich (RST-Dateien) entsteht.
Wenn hierzu die Detailansicht angeschaut wird, ist erkennbar, dass zum einen neue Entwickler zum RST-Bereich hinzukommen und zum anderen ein Wechsel der Entwickler vom TEX-Bereich zum

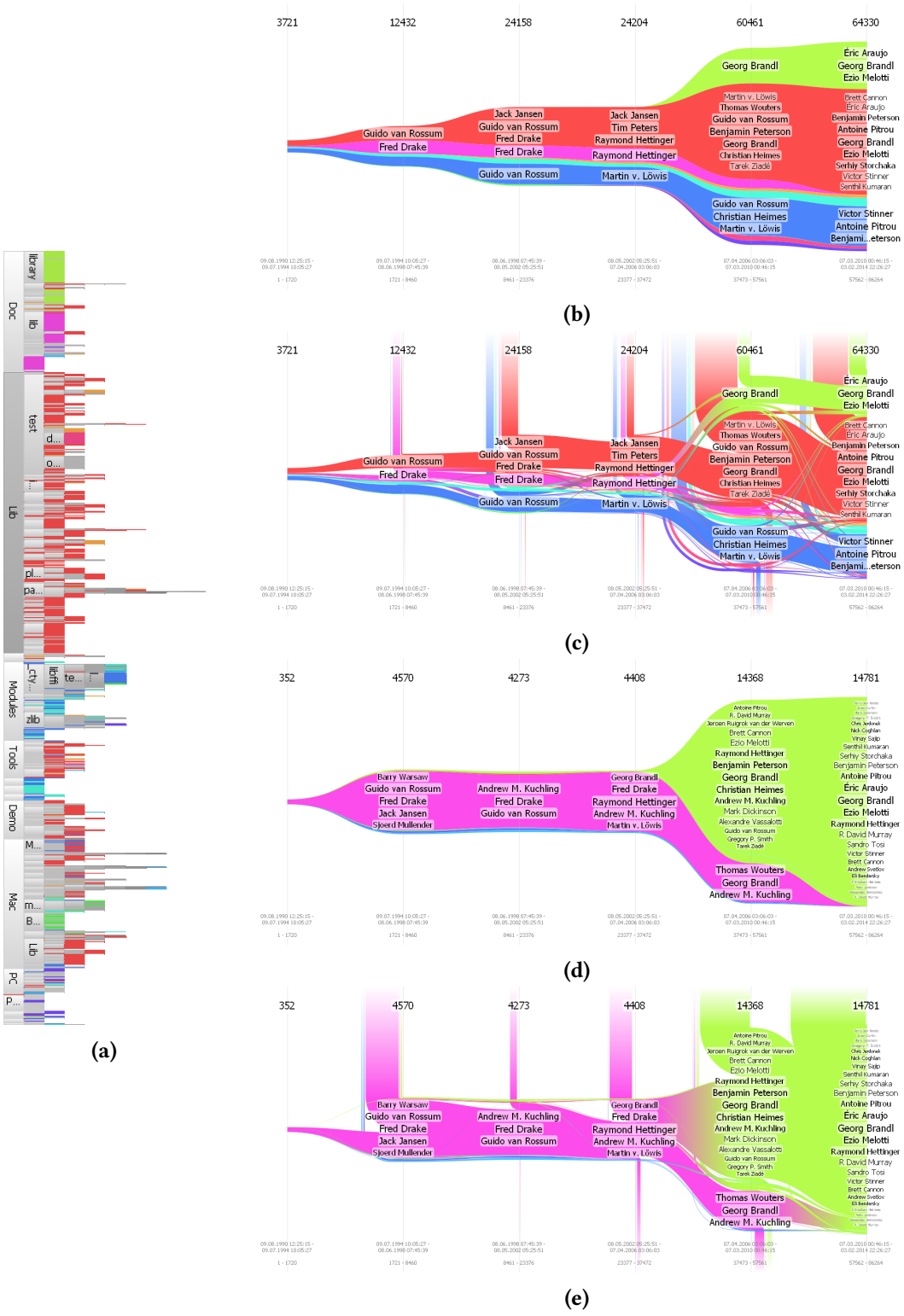


Abbildung 5.4: Darstellungen nach Anzahl der veränderten Dateien mit WOIs für Dateiendungen: (a) Hierarchiedarstellung; (b) und (c): zugehöriger WOI River; (d) und (e): WOI River für den *Doc*-Ordner.

RST-Bereich stattfindet.
Werden als nächstes nur die Änderungen des Dokumentations-Ordners für den WOI River berücksichtigt (Abbildungen 5.4d und 5.4e), ist dieser Wechsel sehr deutlich erkennbar. Es findet anscheinend ein Wechsel vom Ordner *Doc/Lib* (enthält sehr viele TEX-Dateien) zu *Doc/Libraries* (enthält die RST-Dateien) statt. Wahrscheinlich wurde das Dokumentationsformat von TEX auf RST umgestellt.

5.2 libvpx

Als nächstes wird die Entwicklung der Bibliothek *libvpx* der Videocodecs VP8/VP9 für verlustbehaftete Komprimierung von Videodaten untersucht.

Der Codec VP8 wurde zunächst von dem Unternehmen On2 Technologies entwickelt [web]. Seit dem Jahr 2010 besitzt Google durch die Übernahme von On2 Technologies die Rechte an dem Projekt und stellt es als Open-Source-Software zur freien Verfügung. In den darauf folgenden Jahren begann die Entwicklung des Nachfolgers VP9.

Die zur Verfügung stehende History stammt aus einem Git-Repository [lib]. Die Daten reichen von Mitte 2010 (als das Projekt als freie Software freigegeben wurde) bis März 2014, also über einem Zeitraum von etwa vier Jahren. Es wurden alle 5222 Commits von 106 Entwicklern berücksichtigt. Die Projekthierarchie umfasst 2364 Knoten mit 2196 Dateien und besitzt eine maximale Tiefe von acht. Für die Analyse wurden drei WOIs festgelegt und im Nachfolgenden durch die jeweils angegebenen Farben dargestellt: die Verzeichnisse *vp8* (grün) und *vp9* (rot) sowie alle restlichen Ordner/Dateien, die unter anderem Tests und Beispiele enthalten (blau). In Abbildung 5.5a sind die WOIs innerhalb der Projekthierarchie erkennbar.

Für die Untersuchung des Entwicklungsprozesses wird zunächst der zeitliche Verlauf der Entwickleranzahlen betrachtet. Die Abbildungen 5.5b und 5.5c zeigen die Übersicht und Detailansicht dieser Entwicklung.

In Abbildung 5.5b sieht man, dass an dem Projekt meistens etwa 31–53 Entwickler arbeiten; gegen Anfang ist die Anzahl etwas erhöht, dann sinkt sie leicht und gegen Ende scheint sie stärker zu wachsen. Das Projekt wird im Vergleich zu Python schon von Beginn an von mehreren Entwicklern bearbeitet. Dies liegt wahrscheinlich daran, dass libvpx schon bevor es als freie Software zur Verfügung stand, entwickelt wurde und vielleicht noch die gleichen Entwickler wie zuvor daran arbeiten. Man kann feststellen, dass in den ersten beiden Jahren, bis 2012, nur im Ordner *vp8* und an den weiteren Dateien gearbeitet wird. Ende 2012 dominiert die Entwicklung im Verzeichnis *vp9* stark, bis sie 2014 die Entwicklung in *vp8* fast vollständig abgelöst hat.

In der zugehörigen Detailansicht (Abbildung 5.5c) fallen zunächst recht viele Zu- und Abflüsse auf, es scheint einen ständigen Wechsel der Entwickler zu geben bzw. viele Entwickler, die sich vorübergehend an dem Projekt beteiligen und es wieder verlassen. Gegen Ende ist allerdings ein deutlicher Zufluss zu erkennen. Um zu unserer vorherigen Feststellung zurück zu kommen: Auch hier ist der Wechsel (als ein Entwicklerstrom von *vp8* zu *vp9*) erkennbar. Zunächst bleibt einige Beteiligung bei dem Verzeichnis *vp8* (evtl. arbeiten Entwickler noch an beiden Versionen, aber verstärkt an der neueren, oder es haben einige gewechselt, während andere noch an der alten Version arbeiten). Im

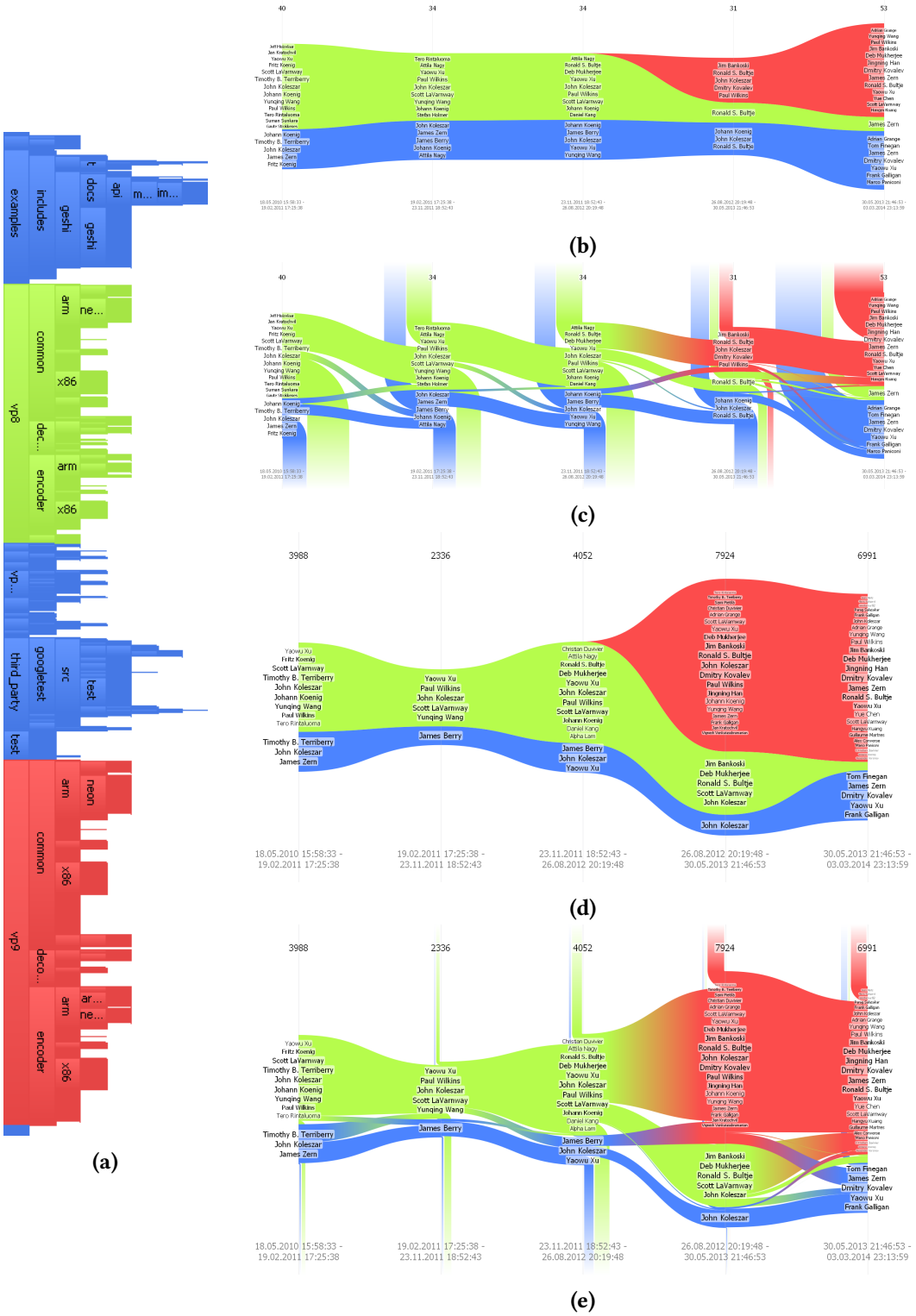


Abbildung 5.5: Softwareentwicklungsverlauf von libvpx: (a) Projekthierarchie; (b) und (c): Darstellung der Anzahl der beteiligten Entwickler als Übersicht und Detailansicht; (d) und (e): Darstellung der Anzahl an veränderten Dateien als Übersicht und Detailansicht.

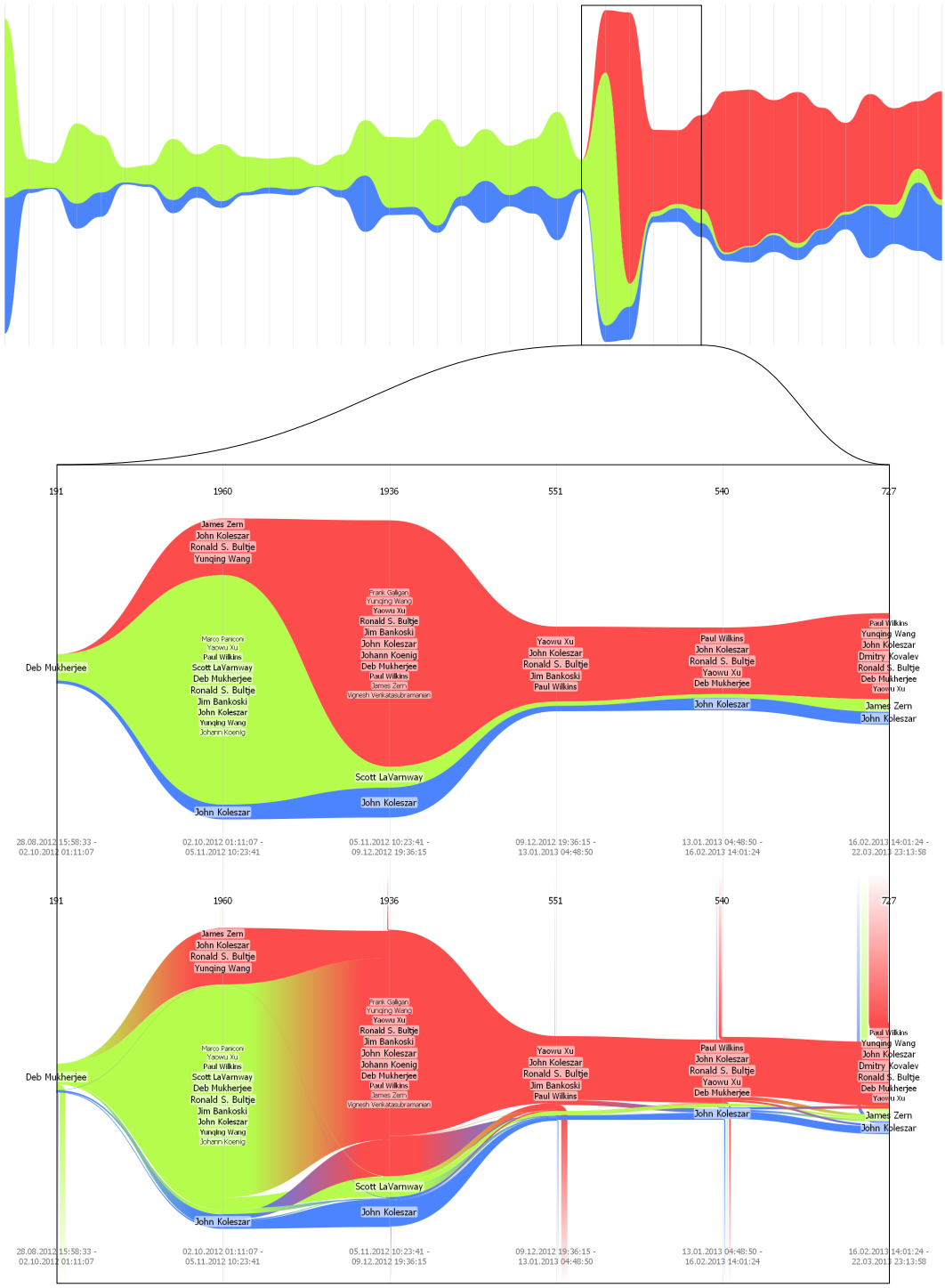
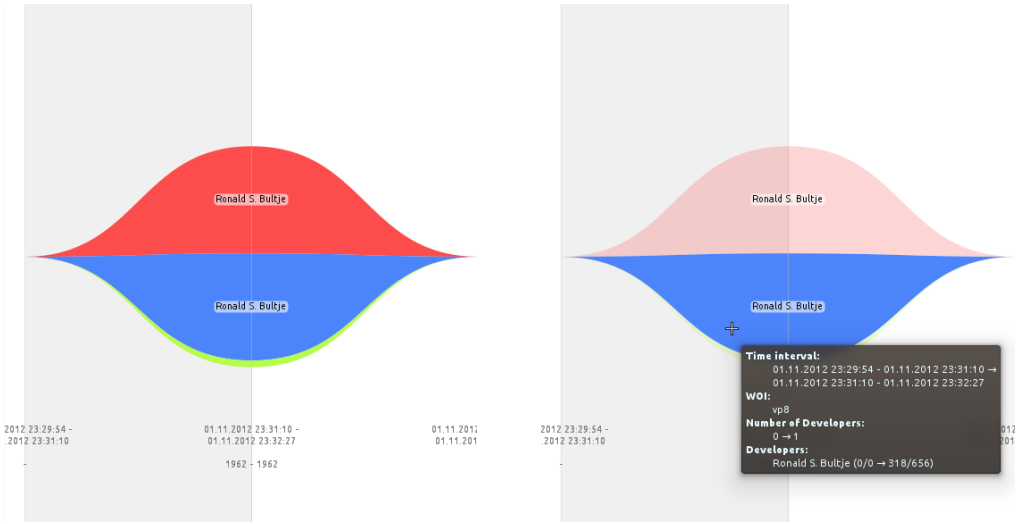


Abbildung 5.6: Softwareentwicklungsverlauf von libvpx in Bezug auf die Anzahl der veränderten Dateien. Oben: gesamter Verlauf; unten: Übersicht und Detailansicht eines ausgewählten Zeitabschnitts.



(a) (b)

Abbildung 5.7: Softwareentwicklungsverlauf von libvpx: Erster Commit in *vp9*.

nächsten Zeitschritt geht dann weitere Entwicklerkapazität zu *vp9* über, bis die Entwicklung von *vp8* fast aufzuhören scheint.

Bei der Betrachtung der Visualisierungen in Abhängigkeit der Änderungen (Abbildungen 5.5d und 5.5e) werden die bereits festgestellten Besonderheiten nochmals verstärkt erkennbar. Zusätzlich ist zu sehen, dass die Anzahl der veränderten Dateien leicht sinkt, dann immer stärker wächst, bis sie ihr Maximum bei dem erwähnten Wechsel erreicht hat und dann wieder leicht abnimmt. Zu- und Abflüsse fallen fast nicht auf – wahrscheinlich bewirken die Entwickler, die in den vorherigen Visualisierungen durch Zu- und Abflüsse sehr stark dominiert haben, nur sehr wenige Änderungen an den Dateien und fallen daher kaum ins Gewicht. Bei dem Wechsel ist erkennbar, dass die Entwickler, die bereits an dem Projekt arbeiten, immer mehr Änderungen zum Projekt beitragen. Dies sieht man daran, dass die Übergänge von grün nach rot und grün nach grün im dritten Abschnitt breiter werden. Hier wechseln zudem Entwickler von den weiteren Dateien zu *vp8*. Im nächsten Schritt geht die Beteiligung wieder etwas zurück und es wirkt so, als ob Entwickler, die vorübergehend bei Änderungen im Ordner *vp9* mitgewirkt haben, wieder zurück zu den weiteren Dateien gehen.

Da die Aufteilung der vier Jahre Entwicklungsprozess in nur fünf Zeitintervalle etwas ungenau ist, betrachten wir nun noch eine feinere Aufteilung. In Abbildung 5.6 wurde der Zeitbereich in 40 Intervalle aufgeteilt. Auch hier fällt der Übergang zwischen der Beteiligung bei *vp8* und *vp9* gleich auf. Die allgemeine Beteiligung der Entwickler ist hier sehr groß. Davor ist sie eher gering, danach sinkt sie auch wieder stark, bleibt aber etwas höher als zuvor. Wenn wir uns nur den Zeitbereich der Änderung genauer anschauen (der Bereich darunter), sticht heraus, dass der Übergang recht schnell vollzogen wird. Aus den vorherigen Bildern könnte entnommen werden, dass der Wechsel ein länger andauernder Prozess ist, der über ein Jahr anhält; dies liegt an der Aufteilung des Zeitbereichs in sehr wenige Intervalle, hierdurch werden die Daten sehr stark aggregiert und ungenau. In dieser Ansicht wird nun klar, dass der Wechsel in sehr kurzer Zeit durchgeführt wird, bis fast alle Entwickler zwischen den beiden Bereichen gewechselt haben.

Bei einer noch feineren Aufteilung der Intervalle bzw. bei einer Einschränkung des dargestellten Zeitbereichs kann sogar herausgefunden werden, wann das erste Mal im Ordner *vp9* committet wurde: Am 1. 11. 2012 hat Ronald S. Bultje die erste Änderung im Verzeichnis *vp9* vorgenommen. Durch Verwendung der Tooltips (sowohl die der Intervalle als auch die der Fluss-Elemente; in Abbildungen 5.7b ist einer dieser Tooltips abgebildet) kann herausgefunden werden, dass bei diesem Commit die genau gleiche Anzahl an Dateien in *vp8* und *vp9* bearbeitet wurden. Möglicherweise hat er hier Dateien im *vp8*-Verzeichnis gelöscht und bei *vp9* eingefügt; dies kann allerdings in der Anwendung nicht herausgefunden werden, da zwischen Dateien, die erzeugt, gelöscht oder verändert werden nicht unterschieden wird.

Wenn nun wieder die Abbildung 5.6 (unten) betrachtet wird, ist hier zudem sichtbar, dass bei der Anzahl der Änderungen schon vor dem Wechsel zu *vp9* bereits ein starkes Wachstum in *vp8* zu erkennen ist (der grüne Bereich ganz links, der sehr schmal anfängt und sehr breit wird, bevor er sich wieder fast auflöst), ohne dass Zuflüsse hinzukommen; erst danach wechselt der Großteil der Beteiligung von *vp8* zu *vp9*. Darüber hinaus ist auch ein stärkerer Wechsel aus den weiteren Dateien zu *vp9* vorhanden. Gegen Ende des Zeitbereichs wechseln die Entwickler teilweise wieder zurück in diesen Bereich.

In den Darstellungen kann demnach erkannt werden, dass zunächst im Bereich von VP8 und später von VP9 entwickelt wurde. Gleichzeitig wurden Dateien für Tests und Beispiele erstellt oder bearbeitet. Gegen Mitte des betrachteten Zeitraums findet eine kurze Übergangsphase statt. Fast die gesamte Entwicklerleistung wechselt von *vp8* zu *vp9* und nimmt dabei zu. In dieser Übergangsphase wurden deutlich mehr Änderungen vorgenommen als davor oder danach. Die Wartung der alten Version bricht jedoch nie vollständig ab, wird aber deutlich weniger, bis sie noch eine geringe gleichmäßige Beachtung erhält. Die Testdateien werden wahrscheinlich für beide Versionen verwendet und sind daher unabhängig von dem Wechsel.

5.3 Linux-Kernel

Das letzte Projekt, das betrachtet wird, ist der Linux-Kernel; dieser wird bereits seit 1991 entwickelt. Die verwendeten Log-Daten des Git-Repositories [Lin] reichen allerdings nur von Mitte 2005 bis heute. Für diesen Zeitraum konnten 426935 Commits von 11495 Entwicklern berücksichtigt werden. Die Projekthierarchie umfasst 81938 Knoten, wobei 77599 Dateien enthalten sind; sie hat eine maximale Tiefe von zwölf. Es werden folgende Unterhierarchien als WOIs gewählt: *drivers*, *fs*, *arch*, *sound*, *include*, *Documentation*. Alle restlichen Dateien werden einem weiteren WOI zugeordnet.

In Abbildung 5.8a ist die Hierarchie mit den ausgewählten WOIs dargestellt, daneben die Übersicht und Detailansicht zu jeweils den Entwickleranzahlen und der Anzahl an veränderten Dateien.

Auf den ersten Blick ist bereits in Abbildung 5.8b sichtbar, dass die Gesamtzahl der Entwickler in der gesamten Zeit leicht ansteigt. Die einzelnen Bereiche werden gleichmäßig von etwa einem gleichen Anteil der Bearbeiter modifiziert. Aufgeklappt (Abbildung 5.8c) sieht jeder Übergangsbereich etwa gleich aus. Hier gibt es, wie in den beiden Fallstudien zuvor, recht viele Zu- und Abflüsse. Ein Hauptfluss von Entwicklern, die durchgängig am Projekt zu arbeiten scheinen, kann in dem WOI

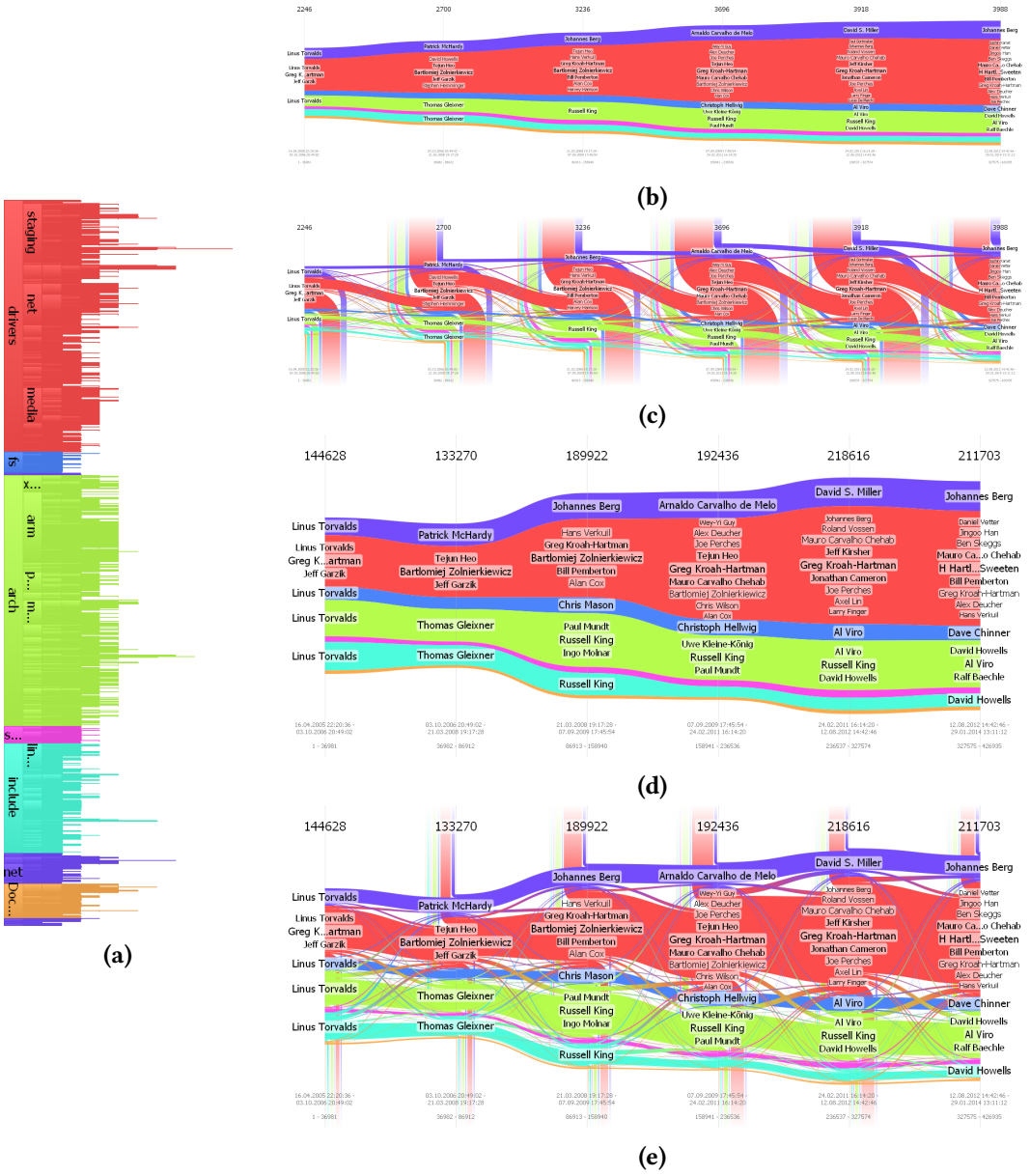
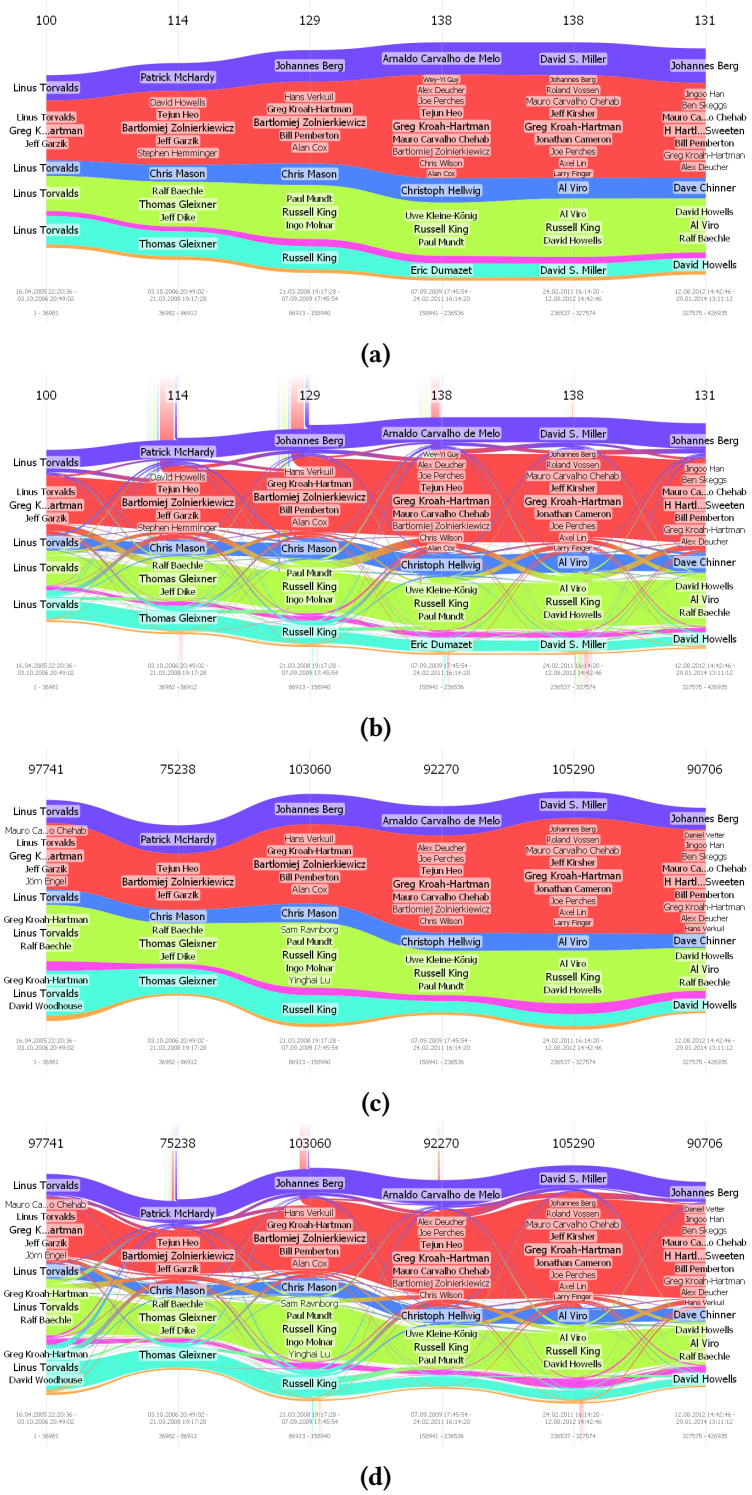


Abbildung 5.8: Softwareentwicklungsverlauf des Linux-Kernels: (a) Projekthierarchie; (b) und (c): Darstellung der Anzahl der beteiligten Entwickler als Übersicht und Detailansicht; (d) und (e): Darstellung der Anzahl an veränderten Dateien als Übersicht und Detailansicht.



(a)

(b)

(c)

(d)

Abbildung 5.9: WOI River des Linux-Kernels für die 140 Entwickler mit den meisten Dateiänderungen: (a) und (b): Darstellung der Anzahl der beteiligten Entwickler als Übersicht und Detailansicht; (c) und (d): Darstellung der Anzahl an veränderten Dateien als Übersicht und Detailansicht.

drivers erkannt werden (der rote Fluss); hier scheinen auch allgemein die meisten Entwickler zu arbeiten.

Bei den Darstellungen in Abhängigkeit der veränderten Dateien (Abbildungen 5.8d und 5.8e) ist eine ähnliche Entwicklung zu erkennen, es gibt allerdings deutlich weniger Zu- und Abflüsse; daher sind hier die einzelnen Übergänge zwischen den verschiedenen WOIs besser erkennbar. Zudem ist sichtbar, dass recht viele Entwickler in aufeinanderfolgenden Zeitintervallen in den gleichen WOIs entwickeln. Wechsel zwischen verschiedenen WOIs kommen eher selten vor – hauptsächlich zwischen den Ordnern *drivers* und *arch*, wobei eine ähnliche Entwicklerkapazität in die eine, aber auch die andere Richtung wechselt.

Da an dem Projekt sehr viele Entwickler beteiligt sind, viele davon jedoch kaum etwas verändern, werden im Folgenden noch einige Darstellungen betrachtet, die nur Hauptentwickler berücksichtigen. Hierfür sind in Abbildung 5.9 Visualisierungen abgebildet, die das Verhalten der 140 Entwickler zeigen, welche die meisten Dateiänderungen bewirkt haben (jeweils mindesten 1500 Dateiänderungen). Bei dem Vergleich dieser Visualisierungen fällt auf, dass die Verteilung der Anzahl an Entwicklern und der Dateiänderungen sich im Wesentlichen nur gering unterscheiden. Es sind kaum Zu- bzw. Abflüsse zu erkennen, was bedeutet, dass fast alle Entwickler in dem dargestellten Zeitbereich an dem Projekt beteiligt waren. Während bei den zuvor betrachteten Visualisierungen sowohl die Anzahl der Entwickler, als auch die der Dateiänderungen monoton gewachsen ist, schwanken hier die Anzahlen teilweise.

Allgemein könnte man vermuten, dass es sich hier um einen gleichmäßigen und stabilen Entwicklungsprozess handelt, bei dem jeder Arbeitsbereich jeweils zu gleichen Anteilen bearbeitet wird. Es gibt zwar, je nach Art der Darstellung, recht viele Abflüsse, aber jeweils noch mehr Zuflüsse.

Obwohl dieses Projekt recht groß ist, sind in der Visualisierung immer noch Trends und Merkmale erkennbar.

6 Zusammenfassung und Ausblick

Ziel dieser Arbeit war die Umfunktionierung des Konzepts der AOI Rivers [BKW13] in ein interaktives Visualisierungswerkzeug zur Analyse dynamischer Entwicklerzahlen in WOIs eines Softwaresystems.

Die AOI River-Visualisierung wurde ursprünglich für Eye-Tracking-Daten verwendet; in dieser Arbeit wurde sie für die Analyse von Softwareentwicklungsprozessen angepasst und erweitert.

Ein WOI River kann die dynamischen Verhaltensweisen von Entwicklergruppen veranschaulichen. Durch ihn kann die Anzahl an Entwicklern bzw. die Anzahl ihrer Änderungen am Softwaresystem für festgelegte WOIs über die Zeit visualisiert werden. Außerdem können Wechsel zwischen verschiedenen WOIs sowie Zu- und Abflüsse zum bzw. aus dem Projekt sichtbar gemacht werden. Es wurden verschiedene Methoden zur Behandlung der Höhe der einzelnen Entwickler für die einzelnen Intervalle und Transitionen untersucht.

Für die Visualisierungstechnik wurde eine Anwendung implementiert, in der die Projekthierarchie und der zugehörige WOI River interaktiv untersucht werden können. Die verschiedenen WOIs, der darzustellende Zeitbereich, die Einteilung in Intervalle und die Entwickler, welche für die Visualisierung berücksichtigt werden sollen, können hier angegeben werden. Für die Festlegung der WOIs können verschiedene Hilfsfunktionen und Filter eingesetzt werden. Weitere Einstellungsmöglichkeiten für die Berechnung des WOI Rivers und für die Anzeige verschiedener Elemente stehen zur Verfügung. Ein WOI River bietet zunächst in Form eines ThemeRivers einen Gesamtüberblick des Entwicklungsprozesses. Auf Wunsch können Bereiche zwischen Intervallen in der Darstellung verändert werden, um Details bezüglich der Zu- und Abflüsse zum bzw. aus dem Projekt sowie den Übergängen zwischen WOIs anzuzeigen.

Die WOI River-Visualisierungstechnik liefert ästhetische Darstellungen eines Softwareentwicklungsprozesses, in denen Entwicklungsstrategien und -verhalten sichtbar gemacht werden können. Es wurde festgestellt, dass die Technik für unterschiedlich große Projekte geeignet ist und auch für größere Projekte wie den Linux-Kernel skalierbar ist. Entwickler, die wenig zum Projekt beitragen, dominieren teilweise in der Visualisierung durch häufige Zu- und Abflüsse. Indem die dargestellten Daten reduziert werden, um weniger relevante Aspekte, wie Commits mit sehr vielen Dateiänderungen oder Entwickler mit geringer Beteiligung auszublenden, können Rauschen und Visual Clutter vermindert und ein Überblick über den Entwicklungsprozess gewonnen werden. Interaktionstechniken können hilfreich sein, das Dargestellte in den WOI Rivers besser zu verstehen und weitere Details herauszufinden.

Anhand dreier Fallstudien wurde untersucht, welche Einsichten in die Entwicklungsprozesse der Projekte mit Hilfe der Visualisierung gewonnen werden können und wie interaktiv weitere nützliche

Details herausgefunden werden können. Zuerst wurde die Entwicklung der Programmiersprache Python analysiert, gefolgt von der Bibliothek libvpx und dem Linux-Kernel.

Ausblick

Die Visualisierung liefert einen guten Überblick über die Anzahl der Entwickler, die in verschiedenen Arbeitsbereichen gearbeitet haben oder über deren Änderungen am Projekt. Es gibt jedoch noch viele Möglichkeiten die Visualisierung anzupassen um noch mehr über den Entwicklungsprozess herauszufinden oder die Analyse zu erleichtern. Im Folgenden werden einige dieser Ideen vorgestellt.

Die Intervalle, die in der Anwendung gebildet werden können, haben jeweils etwa die gleiche Größe; sie decken entweder einen festen Zeitbereich oder eine bestimmte Anzahl an Commits ab. Bei dieser Einteilung könnte nach verschiedenen anderen Kriterien vorgegangen werden. Es wäre möglich wichtige Ereignisse zu berücksichtigen, wie etwa Releases oder das Anlegen von Tags im Versionskontrollsystem. Hiermit könnte das Verhalten zwischen wichtigen Meilensteinen der Entwicklung untersucht werden.

Der Zeitbereich für die Visualisierung kann beliebig festgelegt werden. Wenn entsprechende Bereiche in der WOI River-Visualisierung interaktiv selektiert werden könnten, wäre eine direkte Darstellung des entsprechenden Zeitbereichs mit einer feineren Aufteilung möglich. Es müssten nicht zuerst die Grenzen des gewünschten Zeitbereiches herausgefunden werden; der Analyseprozess könnte hiermit vereinfacht werden.

Darüber hinaus wäre das Hervorheben eines einzelnen Entwicklers oder einer Entwicklergruppe im Kontext des gesamten Flusses sehr nützlich um die Beteiligung der entsprechenden Entwickler im Vergleich mit den restlichen Entwicklern zu sehen und besser verfolgen zu können. Je nach Einstellungen könnte erkannt werden, ob die ausgewählten Entwickler in den Bereichen der Hauptentwicklung arbeiten, in einzelnen Bereichen dominieren oder über mehrere Bereiche verteilt sind.

Dadurch, dass Entwickler häufig gleichzeitig an verschiedenen Bereichen des Projekts beteiligt sind wird die Höhe, die einen Entwickler in der Visualisierung repräsentiert, auf möglicherweise mehrere WOIs aufgeteilt. Eventuell müssen die Entwickler manche Dateien, die außerhalb ihres eigenen Arbeitsbereiches liegen sehr selten verändern, wodurch viele dünne Übergänge zwischen den WOIs entstehen können. Würde einem Entwickler in jedem Intervall nur ein WOI zugeordnet werden – der WOI zu dem er am meisten beigetragen hat – könnte Visual Clutter vermieden werden, allerdings würden auch leicht wichtige Informationen verloren gehen.

Gelegentlich wird ein Projekt von Entwicklern nur für eine gewisse Zeit verlassen, bevor sie wieder zum Projekt zurückkehren. Dies hat mehrere Zu- und Abflüsse zur Folge. Hier könnte überlegt werden, ob zwischen den Entwicklern, die das Projekt vollständig und denen, die es nur vorübergehend verlassen unterschieden wird, um die Zu- und Abflüsse in unterschiedlicher Weise zu visualisieren. Beispielsweise könnten „inaktive“ Entwickler in einen zusätzlichen „inaktiven“ Fluss fließen und aus diesem wieder zurück in das Hauptprojekt. Ein anderer Ansatz könnte sein, Zu- und Abflüsse in einer anderen Darstellungsform, z.B. durch eine verkürzte vertikale Länge der Elemente, anzuzeigen.

Die Reihenfolge in der die einzelnen Flüsse vertikal angeordnet sind spielt eine recht große Rolle um Besonderheiten zu erkennen - derzeit entspricht sie der Anordnung in der Hierarchie oder der Reihenfolge in der sie festgelegt wurden. Befinden sich beispielsweise zwei WOIs, die sich gegenseitig ablösen übereinander ist ein Übergang von dem einen zu dem anderen WOI gut erkennbar. Befinden sich jedoch andere WOIs dazwischen, könnte dieser Wechsel möglicherweise übersehen werden. Zusätzlich ist es in einigen Fällen schwierig zu erkennen, welche Arbeitsbereiche die größte Beteiligung besitzen, vor allem wenn mehrere Bereiche gleich stark bearbeitet werden. Außerdem gibt es in der Visualisierung teilweise sehr viele sich überkreuzende Transitionen zwischen verschiedenen WOIs. Besonders störend könnte es sein, wenn viele Transitionen zwischen den obersten und den untersten WOIs verlaufen. Ein geeigneter Algorithmus könnte dafür sorgen, dass eine optimierte Reihenfolge der Flüsse erreicht wird.
Ein einfacher Ansatz hierzu könnte sein, die Flüsse nach ihrer durchschnittlichen oder maximal vorkommenden Höhe zu sortieren. Hierdurch ist schneller erkennbar wo die Beteiligung einzelner Entwickler am größten war. Würde der Fluss mit der höchsten Wichtigkeit in der Mitte platziert werden und nach außen die eher unwichtigeren, wäre eine Reduzierung der Überkreuzungen möglich, da wahrscheinlich häufiger Transitionen zu oder von den wichtigeren WOIs ausgehen. Zu dem wichtigsten Fluss müssten folglich keine Transitionen über die gesamte vertikale Visualisierung verlaufen, sondern nur bis etwa zur Mitte.

Auch die Farbvergabe für die einzelnen WOIs könnte optimiert werden. Während momentan zwar Farben mit einem möglichst hohen Kontrast vergeben werden, könnte zusätzlich der Flussverlauf berücksichtigt werden. Hierbei müsste darauf geachtet werden, dass an benachbarte Flüsse und bei Flüssen zwischen denen Transitionen verlaufen eine möglichst kontrastreiche Farbe vergeben wird.

Zudem ist denkbar, weitere Informationen des Versionskontrollsystems zu verwenden und diese in die Anwendung zu integrieren. Eine Verwendung der Commit-Kommentare oder das Zugreifen auf bestimmte Dateien und deren Veränderungen könnte die Anwendung noch hilfreicher machen.

Literaturverzeichnis

[AMM+07] W. Aigner, S. Miksch, W. Müller, H. Schumann, C. Tominski. Visualizing time-oriented data-A systematic view. *Comput. Graph.*, 31(3):401–409, 2007. doi:10.1016/j.cag.2007.01.030. URL http://dx.doi.org/10.1016/j.cag.2007.01.030. (Zitiert auf Seite 9)

[Ank09] M. Ankerl. How to Generate Random Colors Programmatically. http://martin.ankerl.com/2009/12/09/how-to-create-random-colors-programmatically/, 2009. (Zitiert auf Seite 34)

[BBM+14] F. Beck, M. Burch, T. Munz, L. Di Silvestro, D. Weiskopf. Generalized Pythagoras Trees for Visualizing Hierarchies. In *IVAPP '14: Proceedings of the 5th International Conference on Information Visualization Theory and Application*, S. 17–28. SCITEPRESS, 2014. (Zitiert auf den Seiten 7 und 9)

[BDL05] M. Balzer, O. Deussen, C. Lewerentz. Voronoi Treemaps for the Visualization of Software Metrics. In *Proceedings of the 2005 ACM symposium on Software visualization*, SoftVis '05, S. 165–172. ACM, New York, NY, USA, 2005. doi:10.1145/1056018.1056041. URL http://doi.acm.org/10.1145/1056018.1056041. (Zitiert auf Seite 8)

[BHW00] M. Bruls, K. Huizing, J. Wijk. Squarified Treemaps. In W. Leeuw, R. Liere, Herausgeber, *Data Visualization 2000*, Eurographics, S. 33–42. Springer Vienna, 2000. doi:10.1007/978-3-7091-6783-0_4. URL http://dx.doi.org/10.1007/978-3-7091-6783-0_4. (Zitiert auf Seite 8)

[BKW13] M. Burch, A. Kull, D. Weiskopf. AOI Rivers for Visualizing Dynamic Eye Gaze Frequencies. In *Computer Graphics Forum*, Band 32, S. 281–290. Wiley Online Library, 2013. (Zitiert auf den Seiten 4, 9, 10 und 66)

[BMMS91] A. Buja, J. McDonald, J. Michalak, W. Stuetzle. Interactive data visualization using focusing and linking. In *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on*, S. 156–163, 419. 1991. doi:10.1109/VISUAL.1991.175794. (Zitiert auf Seite 47)

[Bos57] A. E. Bosman. *Het wondere onderzoekingsveld der vlakke meetkunde*. Parcival, Breda, 1957. (Zitiert auf Seite 9)

[BRW10] M. Burch, M. Raschke, D. Weiskopf. Indented Pixel Tree Plots. In *Advances in Visual Computing*, S. 338–349. Springer, 2010. (Zitiert auf Seite 8)

[BW08] L. Byron, M. Wattenberg. Stacked Graphs – Geometry & Aesthetics. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1245–1252, 2008. doi:10.1109/TVCG.2008.166. (Zitiert auf Seite 9)

[Cau10] A. H. Caudwell. Gource: Visualizing Software Version Control History. In *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*, SPLASH '10, S. 73–74. ACM, New York, NY, USA, 2010. doi:10.1145/1869542.1869554. URL http://doi.acm.org/10.1145/1869542.1869554. (Zitiert auf den Seiten 4 und 11)

[Die07] S. Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, 2007. (Zitiert auf den Seiten 10 und 11)

[HHWN02] S. Havre, E. Hetzler, P. Whitney, L. Nowell. ThemeRiver: Visualizing Thematic Changes in Large Document Collections. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):9–20, 2002. doi:10.1109/2945.981848. (Zitiert auf den Seiten 4, 9 und 10)

[iso] Data Elements and Interchange Formats—Information Exchange—Representation of Dates and Times—ISO 8601: 2004. International Standardizaton Organization (ISO). (Zitiert auf Seite 40)

[JS91] B. Johnson, B. Shneiderman. Tree-Maps: a Space-Filling Approach to the Visualization of Hierarchical Information Structures. In *Proceedings of the 2nd conference on Visualization '91*, VIS '91, S. 284–291. IEEE Computer Society Press, Los Alamitos, CA, USA, 1991. URL http://dl.acm.org/citation.cfm?id=949607.949654. (Zitiert auf den Seiten 7 und 8)

[JS10] S. Jürgensmann, H.-J. Schulz. Poster: A Visual Survey of Tree Visualization. In *Proceedings of IEEE Information Visualization*, Band 5. 2010. (Zitiert auf Seite 7)

[KBK11] M. Krstajic, E. Bertini, D. Keim. CloudLines: Compact Display of Event Episodes in Multiple Time-Series. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2432–2439, 2011. doi:10.1109/TVCG.2011.179. (Zitiert auf Seite 9)

[KL83] J. B. Kruskal, J. M. Landwehr. Icicle Plots: Better Displays for Hierarchical Clustering. *The American Statistician*, 37(2):162–168, 1983. (Zitiert auf den Seiten 7 und 9)

[lib] lipvpx-Repository. https://chromium.googlesource.com/webm/libvpx. (Zitiert auf Seite 58)

[Lin] Linux-Kernel-Repository. git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git. (Zitiert auf Seite 62)

[LWW+13] S. Liu, Y. Wu, E. Wei, M. Liu, Y. Liu. StoryFlow: Tracking the Evolution of Stories. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2436–2445, 2013. doi:10.1109/TVCG.2013.196. (Zitiert auf Seite 9)

[Mun13] T. Munz. Hierarchievisualisierung mit verallgemeinerten Pythagoras-Bäumen. Studienarbeit, Universität Stuttgart, VISUS, 2013. (Zitiert auf Seite 8)

[OM09] M. Ogawa, K.-L. Ma. code_swarm: A Design Study in Organic Software Visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1097–1104, 2009. doi:10.1109/TVCG.2009.123. (Zitiert auf den Seiten 4, 11 und 53)

[OM10] M. Ogawa, K.-L. Ma. Software Evolution Storylines. In *Proceedings of the 5th International Symposium on Software Visualization*, SOFTVIS '10, S. 35–42. ACM, New York, NY, USA, 2010. doi:10.1145/1879211.1879219. URL http://doi.acm.org/10.1145/1879211.1879219. (Zitiert auf den Seiten 4, 11, 12, 53 und 54)

[Pyta] Python. http://www.python.org/. (Zitiert auf Seite 53)

[Pytb] Python-Repository. http://hg.python.org/cpython. (Zitiert auf Seite 53)

[RLN07] R. Rosenholtz, Y. Li, L. Nakano. Measuring Visual Clutter. *Journal of Vision*, 7(2):17, 2007. (Zitiert auf Seite 23)

[SČG05] M.-A. D. Storey, D. Čubranić, D. M. German. On the Use of Visualization to Support Awareness of Human Activities in Software Development: A Survey and a Framework. In *Proceedings of the 2005 ACM symposium on Software visualization*, S. 193–202. ACM, 2005. (Zitiert auf Seite 11)

[Shn96] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, S. 336–343. 1996. doi:10.1109/VL.1996.545307. (Zitiert auf Seite 36)

[Spe07] R. Spence. *Information visualization*. Pearson, London u.a., 2007. (Zitiert auf Seite 44)

[SZ00] J. Stasko, E. Zhang. Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, S. 57–65. 2000. doi:10.1109/INFVIS.2000.885091. (Zitiert auf Seite 9)

[TA08] A. Telea, D. Auber. Code Flows: Visualizing Structural Evolution of Source Code. In *Computer Graphics Forum*, Band 27, S. 831–838. Wiley Online Library, 2008. (Zitiert auf den Seiten 4, 11 und 12)

[Tuf83] E. R. Tufte. *The Visual Display of Quantitative Information*, Band 2. Graphics Press Cheshire, CT, 1983. (Zitiert auf den Seiten 4 und 10)

[web] The WebM Project | Frequently Asked Questions. http://www.webmproject.org/about/faq. (Zitiert auf Seite 58)

[Wet03] K. Wetzel. Pebbles – using Circular Treemaps to visualize disk usage. http://lip.sourceforge.net/ctreemap.html, 2003. (Zitiert auf Seite 8)

[ZW04] T. Zimmermann, P. Weißgerber. Preprocessing CVS Data for Fine-Grained Analysis. In *Proceedings International Workshop on Mining Software Repositories, MSR04*. 2004. (Zitiert auf Seite 37)

Alle URLs wurden zuletzt am 14. 03. 2014 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift