- Diplomarbeit -

Zur Dualität von Modularität und Open Source Geschäftsmodell: Eine Analyse durch Call Graph Extraktoren und Design Structure Matrices

vorgelegt von

Matthias Gerz

Matr.-Nr.: 204 210 293

Fachbereich 4: Informatik
Institut für Management

Betreuer: Prof. Dr. Harald F. O. von Kortzfleisch
Diplom Informatiker Mario Schaarschmidt

November 2009

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Bendorf, den 27.11.2009

Abstract

Einige akademische Arbeiten behaupten, dass Software, die in einem offenen Prozess erstellt wurde, strukturierter ist, als Programme, die im geschlossenen Umfeld entstanden sind. Die Rede ist hier von Open Source und Closed Source. Der Nachweis dieser Annahme wird mit der Technik der so genannten Design Structure Matrix (DSM) erbracht. DSM erlauben die Visualisierung der Struktur einer Software und ermöglichen das Berechnen von Metriken, mit denen dann der Level an Modularität der einzelnen Open Source Software (OSS) verglichen werden kann. Unter zu Hilfename dieser Technik geht die Diplomarbeit der Frage nach, ob ein Zusammenhang zwischen Geschäftsmodell und OSS existiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CEO	Chief Executive Officer
CFS	Cluster-File-System
CRM	Customer Relationship Systeme
CSV	Character Separated Values
DSM	Design Structure Matrix
FSE	Free Software Foundation
GNU	Gnu's Not Unix
GPL	General Public License
GUPRO	Generische Umgebung zum Programmverstehen
JVM	Java Virtual Machine
JPG	Dateiendung für JPEG-Dateien
JPEG	Joint Photographic Experts Group
MIT	Massachusetts Institute of Technology
OSI	Open Software Initiative
OSS	Open Source Software
PSM	Problem Solving Matrix
SVG	Scalable Vector Graphic

Vorwort

Die vorliegende Diplomarbeit beschäftigt sich mit der Analyse von Open Source Software. Sie wurde in der Zeit von Juni bis November in der Arbeitsgruppe für Management von Information, Innovation, Entrepreneurship und Organisation (MI^2EO) unter der Leitung von Herrn Prof. Dr. Harald F. O. von Kortzfleisch angefertigt. Der Diplomarbeit vorangegangen ist die Mitarbeit in der Arbeitsgruppe MI^2EO, in deren Folge durch Gespräche das Thema und die Aufgabenstellung entstanden sind.

Mein besonderer Dank gilt Herrn Mario Schaarschmidt, Wissenschaftlicher Mitarbeiter in der Arbeitsgruppe MI^2EO, für die umfangreiche Betreuung der Arbeit, sowie die Bereitstellung von ersten Materialien. Während er mich durch die Erstellungsphase der Diplomarbeit begleitete, habe ich viel über wissenschaftliches Arbeiten gelernt. In Gesprächen fanden wir zusammen immer anregende Ideen und Ansätze. Auf seine Hilfe konnte ich mich jederzeit verlassen. Bei Herrn Prof. Dr. Harald F. O. von Kortzfleisch bedanke ich mich für die Erstkorrektur und die gute Zusammenarbeit in der Arbeitsgruppe MI^2EO. Nicht unerwähnt sollen auch die Firmen Latix und SciTools bleiben, die mir für den Zeitraum der Diplomarbeit eine Forschungslizenz für die Produkte „Latix“ und „Understand 2.0“ eingeräumt haben.

Die Diplomarbeit besteht aus drei Hauptteilen. Der erste Teil der Arbeit beschäftigt sich mit den Bausteinen, die für das Verstehen dieser Arbeit notwendig sind. Hierzu zählt eine Einführung und Definition von Open Source Software, Geschäftsmodellen, sowie der Programmanalyse. Im zweiten Teil wird auf die eigentliche Analyse eingegangen. Ein Bestandteil dieser Arbeit ist das Finden von geeigneter Analysesoftware und Programmen für das Darstellen und Berechnen von Design Structure Matrices (DSM). Ein etwas intensiverer und zeitaufwändiger Anteil dieser Arbeit ist das Entwickeln eines

eigenen DSM Viewers, der auch das Berechnen von Metriken ermöglicht. Zum mittleren Teil der Diplomarbeit gehört auch das Finden von geeigneter Open Source Software für die Analyse, sowie die Zuordnung der Open Source Software zu Geschäftsmodellen. Die Auswahl der Open Source Software wurde von Herrn Mario Schaarschmidt mit geprägt. Im letzten Abschnitt setzt sich die vorliegende Arbeit mit den Ergebnissen und den daraus möglichen Schlussfolgerungen auseinander.

Bendorf den, 27.11.2009

1. Einleitung

In den letzten Jahren begann im Bereich der Softwareentwicklung ein Trend, der bis heute anhält. Entwickler veröffentlichen ihren Quellcode und lassen dadurch andere an der Software teilhaben und davon profitieren. Durch die Veröffentlichung des Quellcodes spricht man hier von Open Source Software (OSS). Im ersten Teil dieses Kapitels wird die Historie der OSS vorgestellt. Der zweite Abschnitt erläutert deren wirtschaftliche Bedeutung und motiviert für das Thema der Diplomarbeit, welches abschließend vorgestellt wird.

1.1. Historie

Im Gegensatz zu proprietärer[1] Software bietet OSS dem Benutzer die Möglichkeit, den Quellcode (engl. source code) einzusehen [Kri05]. Die Ursprünge der OSS gehen auf das Jahr 1983 zurück. Richard Stallman, zu diesem Zeitpunkt noch Mitarbeiter am Massachusetts Institute of Technology (MIT), schreibt in einem Diskussionsforum (engl. Newsgroup), dass er ein neues, für jedermann frei zugängliches, Unix ähnliches Betriebssystem entwickeln wird. Für den Projektnamen wählt Stallman das Akronym *GNU*(GNUś Not Unix) [Intl]. Zwei Jahre später veröffentlicht Stallman das sog. *Gnu Manefesto*, in dem er seine Motive und den Zweck dieses Projektes vorstellt. Ziel ist das Wiederbeleben der Kooperation in den frühen Jahren der Computergemeinschaft [Inth]. Um die Idee der Freien Software *(Free Software)* zu verstehen, prägte Stallman folgenden Satz:

[1] proprietär stammt vom lateinischen Wort proprietarius „der Eigentümer“ ab

> To understand the concept, you should think of free as in free speech, not as in free beer [Intg].

Um dem GNU Projekt einen finanziellen und juristischen Rahmen zu geben, gründet Stallman 1985 die Free Software Foundation (FSE) [Intf] [Intk]. Damit Software auch freie Software bleibt, werden im Rahmen des GNU Projektes verschiedene Lizenzen entwickelt. Die Bekannteste ist die GNU General Public License (GNU GPL) [Webg].

1998 geht eine Schockwelle durch die Geschäftswelt und die OSS Communities. Um zu verhindern, dass Microsoft mit dem Internet Explorer eine Monoplostellung im Bereich der Internet Browser erlangt, veröffentlicht Netscape den Quellcode des Netscape Navigators. Mit zu dieser Entscheidung hat der Artikel von Raymond *„The Cathedral & the Bazaar“*[2] beigetragen [Ray01]. Im gleichen Jahr wird die Open Software Initiative (OSI) gegründet. Ziel dieser Organisation ist es im Wettbewerb mit proprietärer Software einen geschäftsfreundlicheren und weniger ideologischen Begriff zu definieren. Aus „Free Software“ wird „Open Source“ [Inti] [Inth].

Seit Ende der 90er Jahre lässt sich in der Wissenschaft und auch in der Wirtschaftspraxis ein steigendes Interesse an OSS verzeichnen. Der Marktanteil im Serverbereich ist beachtlich. Auch wenn im Desktopbereich proprietäre Lösungen vorherrschend sind, finden in jüngster Zeit immer mehr OSS Lösungen Anwendung. Eine genaue Anzahl an Nutzern, Entwicklern und OSS Projekten lässt sich nur schwer abschätzen. Eine Annäherung erlauben die Webseiten bei denen OSS Entwickler ihre Projekte verwalten, um Gleichgesinnte daran teilhaben zu lassen. Auf den Webseiten SourceForge, Freshmeat und Savannah sind über 100.000 Projekte mit knapp 1.000.000 Nutzern und Entwicklern vertreten. Selbst Firmen wie IBM beteiligen sich an OSS. IBM ersetzte die eigene Webserversoftware gegen Apache und investierte hohe Summen in die Entwicklung von GNU/Linux [BAPF04].

[2] Dieses Essay über Open Source beschreibt die Vor- und Nachteile der Entwicklungsmethode im OSS Bereich *„Basar“* gegenüber der zuvor gebräuchlichen Methode, die Raymond *„Kathedrale“* nennt.

1.2. Motivation

Das im vorherigen Abschnitt beschriebene Beispiel von IBM zeigt, das Unternehmen Interesse an OSS haben. Das spiegelt sich nicht nur in der von Unternehmen geförderten Entwicklung von OSS wieder, sondern auch bei Übernahmen von Unternehmen, die im Bereich OSS tätig sind. In den vergangen zwei Jahren wurden einige Unternehmen aus dem OSS Umfeld verkauft. Die folgende Auflistung von Diedrich [Die08] zeigt einige Beispiele:

- Citrix kaufte für 500 Millionen US-Dollar XenSource.
- Yahoo kaufte den Groupwareanbieter Zimba.
- Sun übernahm den Hersteller Lustre.
- Der Linux-Distributor Xandros kaufte den Groupware Hersteller Scalix.
- Oracle Corporation[3] kaufte Sun Microsystems[4]. Am 20. April diesen Jahres gaben die beiden Unternehmen die 7,4 Milliarden US-Dollar teure Übernahme bekannt [Into].

Ein Grund für die Übernahmen sind die Rechte am Quellcode eines Produktes. Obwohl jeder den Quellcode nutzen kann, haben Firmen enormes Interesse an den Rechten der OSS [Die08]. Dann gibt es Unternehmen, die ihre vormals proprietären Softwareprodukte unter eine Open Source Lizenz veröffentlichen [Ale08]. Alexy [Ale08] spricht von „Business Transformation“ . Ein bekanntes Beispiel ist die Veröffentlichung des Netscape Navigators (heute Mozilla Firefox) oder die von IBM veröffentliche Entwicklungsumgebung Eclipse [BAPF04]. Die Gründe von Netscape und damit auch die Motivation für diesen Schritt, wurden in der Einleitung genannt. Knoblich [Kno06] macht auf die Vorteile des Open Source Entwicklungsmodell aufmerksam. Unter anderem erwähnt er eine hohe Innovationsgeschwindigkeit, Qualität und Sicherheit von OSS, ermöglicht durch

[3] `http://www.oracle.com`
[4] `http://www.sun.com`

eine Vielzahl an Entwicklern. Der Quellcode wird von mehr Entwicklern getestet und auf Fehler untersucht, als es bei proprietärer Software der Fall ist. Nicht nur die Innovationsgeschwindigkeit sowie Qualität und Sicherheit sind für Unternehmen interessant. Unternehmen können mit OSS die Entwicklungskosten reduzieren. Dies liegt zum Beispiel daran, dass im Gegensatz zur proprietären Software Basisfunktionalitäten aus anderen OSS verwendet werden können und es eine Vielzahl an freiwilligen Entwicklern gibt [Ale08]. Die Gründe sich mit OSS zu befassen sind vielfältig.

Wie kann man aber mit OSS Geld verdienen? Knyphausen-Aufseß [KA04] zeigt in seinem Artikel, dass nur wenige Firmen in der Lage sind, mit OSS Gewinne zu erzielen. Dennoch gibt es Unternehmen, die im Bereich OSS tätig sind. Aufschluss über die Art und Weise wie ein Unternehmen Geld verdient, gibt das Geschäftsmodell. Im Fall von MySQL ist OSS ein Erfolg. Marten Mickos, Geschäftsführer bei MySQL, sagt: *„Wir verfolgen das Ziel, dass die Datenbank MySQL Bestandteil eines jeden Rechners ist“ [Webh].* MySQL hat eine große, unterstützende Gemeinde mit über vier Millionen Installationen. Nach Angaben von Mickos sei die potentielle Reichweite noch größer. Durch die Bündelung mit Produkten von Unternehmen wie Sun und Apple, erzielt MySQL eine weitere Bekanntmachung. Durch diese Bündelung wird das kommerzielle Geschäft gestärkt [Webh]. Das Geschäftsmodell von MySQL ist das sogennante „Dual Licensing“. Das bedeutet das MySQL unter verschiedenen Lizenzen erhältlich ist. MySQL steht einmal unter GPL und einmal unter einer proprietären Lizenz. Letztere ist für Unternehmen interessant, die MySQL in ihre Produkte einbauen wollen, ohne Gefahr zu laufen, dass durch die GPL auch das eigene Produkt unter die GPL fällt [Ale08]. Dabei ist Open Source ausschlaggebend für das erfolgreiche Geschäftsmodell von MySQL. Der OSS Gedanke wird von Beginn an von MySQL leidenschaftlich unterstützt und gefördert. Der Quellcode kann von jedem gelesen, weiterverbreitet und verändert werden. Dies fördert die schnelle Verbesserung und Entwicklung der Software [Webh]. Die Studie von Yuhanna [2004] zum Beispiel zeigt, dass der Code von MySQL viermal besser ist, als bei kommerzieller Software [Yuh04] [Intb].

1.3. Ziel der Diplomarbeit

Falls ein Zusammenhang zwischen Geschäftsmodell und Quellcode existiert, so könnten einige interessante Fragestellungen beantwortet werden. Unternehmen, die sich zum Beispiel mit der Frage beschäftigen, ob sie ihre proprietäre Software unter einer Open Source Software Lizenz veröffentlichen sollen, könnten prüfen, ob ihr Produkt zu dem angestrebten Geschäftsmodell passt oder angepasst werden muss. Statt der Software könnte man auch das Geschäftsmodell anpassen. Eine Software, die bereits OSS ist, kann in der Weise untersucht werden, ob das aktuelle Geschäftsmodell zur Software passt oder ob es nicht bessere wäre, entweder die Software oder das Geschäftsmodell anzupassen. Vor diesem Hintergrund ist ein wichtiger Bestandteil dieser Diplomarbeit das Finden von OSS und den zugehörigen Geschäftsmodellen des OSS Lizenz innehabenden Unternehmens. Das Ziel dieser Arbeit ist es zu prüfen, ob es einen Zusammenhang zwischen der OSS und Geschäftsmodellen gibt.

2. Grundlagen

Für einige Begriffe, die im Rahmen dieser Arbeit verwendet werden, gibt es keine einheitlichen Definitionen. In diesem Kapitel werden die Begriffe OSS und Geschäftsmodell definiert. Eine Einführung in die Programmanalyse runden die Grundlagen ab.

2.1. Open Source Software

Per Definition ist OSS eine Software, die unter einer Open Source Lizenz steht. Jeder Empfänger einer OSS hat das Recht den dazugehörigen Quellcode zu verändern oder anzupassen. Änderungen am Quellcode müssen nur dann veröffentlicht werden, wenn das modifizierte Programm weitergegeben wird [Hen04].

Der Begriff Open Source Software wurde 1997 von Erik Raymond eingeführt. Im Rahmen der Gründung der OSI 1998 präzisierte Bruce Perens den Begriff OSS durch eine Open Source Definition. Diese Definition beruht auf den Debian Free Software Guidelines, die wiederum von der Free Software Foundation[1] abgeleitet wurden [Eve08, S.18]. Für die OSI bedeutet OSS nicht nur Zugriff auf den Sourcecode.

> „Open source doesn't just mean access to the source code.“ [Intj]

Eine OSS muss auch den folgenden Kriterien entsprechen [Intj]:

[1] `http://www.fsfe.org`

1. Freie Weitergabe der Software ohne Lizenzgebühren.
2. Programm muss Quellcode beinhalten.
3. Lizenz muss Veränderungen und Derivate zulassen.
4. Unversehrtheit des Quellcodes des Autors.
5. Keine Diskriminierung von Personen oder Gruppen.
6. Keine Einschränkungen bezüglich des Einsatzfeldes.
7. Weitergabe der Lizenz.
8. Die Lizenz darf die Weitergabe zusammen mit anderer Software nicht einschränken.
9. Lizenz darf nicht auf ein bestimmtes Produktpaket beschränkt sein.
10. Lizenz muss technologieneutral sein.

Evers [2008] führt weitere Begriffe ein, um den überladenen Begriff Open Source zu differenzieren und zu präzisieren. Die Tabelle 2.1 zeigt die einzelnen Begriffsdefinitionen. Im Hinblick auf den Sourcecode ist für die vorliegende Ausarbeitung der Begriff **open-software** wichtig. Im folgenden schließt der Begriff OSS open-software mit ein. OSS bedeutet daher nicht zwingend, dass alle Kriterien der OSI erfüllt sein müssen.

Durch die Veröffentlichung des Quellcodes bietet OSS der Konkurrenz eine Möglichkeit ähnliche oder gleiche Dienstleistungen anzubieten. Damit könnten konkurrierende Firmen direkt im Wettbewerb mit den Urhebern stehen. Im proprietären Bereich lässt sich dies weiterhin durch Lizenzen verhindern [Eve08, S. 13]. Wie kann man aber mit OSS Geld verdienen? Im nächsten Abschnitt werden Geschäftsmodelle vorgestellt, die dies ermöglichen.

Begriff	Definition
open-software	Veröffentlichung der Software unter einer Open Source Lizenz inklusive Quellcode.
open-collaboration	Internetbasierte Kommunikation und Koordination.
open-process	Extern sichtbarer Entwicklungs- und Veröffentlichungsprozess.
open-releases	Häufige Veröffentlichung von Arbeitsergebnissen, d.h. es gibt keine großen Sprünge zwischen den freigegebenen Versionen
open-deployment	Ausrichtung neuer Produkt-Releases auf Plattformen, die hauptsächlich aus Open Source Produkten bestehen.
open-environment	Entwicklung von Systemen mit Werkzeugen die aus Open Source Produkten bestehen [Eve08, Vgl. 19].

Tabelle 2.1.: Weitere OSS Begriffe nach [Eve08, Vgl. 19]

2.2. Geschäftsmodelle

Zu Beginn werden in diesem Abschnitt die Ursprünge des Begriffes Geschäftsmodell aufgezeigt, denen dann anschließend eine Begriffsdefinition folgt. Abschließend werden einige Geschäftsmodelle vorgestellt und ein Zuordnungsverfahren eingeführt. Mit Hilfe dieses Zuordnungsverfahrens können dann Geschäftsmodelle identifiziert werden.

2.2.1. Die Ursprünge des Begriffs Geschäftsmodell

Die Ursprünge des Begriffs **Geschäftsmodell** (engl. Business Model) stammen aus dem Informationsmanagement. Mit dem Begriff Geschäftsmodell versuchte man die Prozesse, Aufgaben und Kommunikationsbeziehungen eines Unternehmens auf ein einheitliches System abzubilden. Die Bedeutung bzw. die Auffassung des Geschäftsmodells hat sich in den vergangenen Jahrzehnten verändert. Heute versteht man unter einem Geschäftsmodell einen Plan, mit dem Unternehmen gewisse Kundenbedürfnisse befriedigen. Die ursprüngliche Sichtweise auf Koordination und den Umgang mit Informationen und Kommunikationstechnologie ging verloren [Stä01, S. 38ff].

2.2.2. Geschäftsmodelldefinition

Es gibt eine Vielzahl an Definitionen für den Begriff Geschäftsmodell. In den letzten Jahren haben sich diverse Definitionen mit verschiedenen Schwerpunkten angesammelt. Eine sehr frühe Definition des Begriffes Geschäftsmodell stammt von Baatz.

> „Baatz (1996) [...] how to make money [...]“ [SDL03]

Der wesentliche Inhalt bei Baatz ist das „wie“ des Geldverdienens. Timmers hingegen stellt klar heraus, dass zu einem Geschäftsmodell Produkt-, Dienstleistungs-, und Informationsflüsse gehören. Diese werden von einem Akteur gesandt und von einem anderen Akteuer empfangen [SDL03].

> Timmers (1998) „An architecture for the product, service and information flows, including a description of the various business actors and their roles; and a description of the potential benefits for the various business actors; and a description of the sources of revenues [SDL03].“

Damit liefert Timmers ein Umfassendes Bild eines Unternehmens. Scheer führt viele weitere Definitionen an. Die Abbildung 2.1 zeigt die einzelnen Autoren und die von ihnen in den Definitionen angesprochenen Merkmale [SDL03].

Auf Basis der Literaturübersicht erstellt Scheer eine eigene Geschäftsmodelldefinition:

> Ein Geschäftsmodell kann als eine abstrahierende Beschreibung der ordentlichen Geschäftstätigkeit einer Organisationseinheit angesehen werden. Diese Abstraktion basiert auf einer Abbildung von Organisationseinheiten, Transformationsprozessen, Transferflüssen, Einflussfaktoren sowie Hilfsmitteln oder einer Auswahl hieraus [SDL03].

Die Aufzählung und Auswertung für eine neue Begriffsbildung endet bei Scheer [SDL03] im Jahr 2002 mit der Definition von Stähler. Stähler [Stä01] geht der Frage nach wodurch Geld verdient wird. Dabei fokussiert Stähler im Wesentlichen den Nutzen, den Prozess

Merkmale \ Autoren (28 Quellen)	Österle (1996)	Caroll und Trebnick (1997)	Timmers (1998)	Lindström (1999)	Nilsson, Tollis und Nellborn (1999)	Willars (1999)	Bartelt und Lamersdorf (2000)	Heinrich und Leist (2000)	Klueber (2000)	Mahadevan (2000)	Martinez (2000)	Zimmermann (2000)	Alt und Zimmermann (2001)	Amit und Zott (2001)	Gordijn und Akkermans (2001)	Porter (2001)	Rentmeister und Klein (2001)	Robert und Racine (2001)	Weill und Vitale (2001)	Bieger, Bickhof und Knyphausen-Aufseß (2002)	Bieger, Rüegg-Stürm und Rohr (2002)	Magretta (2002)	Mercer (2002)	Osterwalder und Pigneur (2002)	Schögel (2002)	Servatius (2002)	Stähler (2002)	Übereinstimmungen
Abstraktion / Aggregation				■	■	■			■						■		■		■	■						■		9
Akteure / Rollen			■				■	■	■	■		■	■	■	■		■	■	■	■	■	■		■		■	■	18
Betrachtung von Unternehmensteil					■														■		■					■		4
Betrachtung von Unternehmen		■			■	■		■	■					■			■	■	■	■	■	■		■	■		■	15
Betrachtung von Unternehmensnetz									■												■							2
Beziehungen der Akteur												■		■					■		■			■				5
Externes Kommunikationskonzept								■												■	■							3
Finanzen / Umsätze (Flüsse)			■				■		■	■	■	■	■				■		■	■	■			■	■	■	■	15
Gewinn	■										■									■		■	■			■		6
Güter / Dienstleistungen (Flüsse)			■				■		■	■		■	■	■	■		■		■	■			■	■	■			14
Informationen (Flüsse)			■				■		■					■			■		■									6
Kontrollmechanismen														■														1
Kritische Erfolgsfaktoren		■																	■									2
Nutzen			■						■			■		■			■		■								■	7
Organisationsform								■													■		■					3
Produktlebenszyklus															■									■				2
Prozesse / Ablauf						■		■					■	■											■	■		6
Rechtliche Aspekte													■	■											■			3
Ressourcen (Allgemein)														■									■		■			3
Strategie / Vision / Ziel					■			■					■			■			■	■			■		■			8
Technologie						■			■								■							■	■			5
Wachstum		■																		■	■							3
Wertkette / Kernkompetenzen								■							■				■	■	■	■					■	7
Wertschöpfung									■	■			■	■	■	■					■	■		■	■	■	■	12
Wettbewerbsumfeld						■		■	■				■			■	■			■			■		■			9

■ wird erwähnt □ wird nicht erwähnt

Abbildung 2.1.: Geschäftsmodellbegriffe und ihre Komponenten [SDL03]

der Nutzenerstellung und die dadurch möglicherweise vorhandenen Einnahmen auf der Unternehmensseite [Stä01] [SDL03].

Für diese Arbeit gilt für den Begriff Geschäftsmodell die Definition von Alexy. Alexy [2008] greift die seiner Meinung nach aktuelle Definition von Geschäfstmodellen von West und Gallagher auf und stützt damit die Arbeit von Staehler [2001]:

> „Using the least common denominator found in the literature, for this thesis a business model shall be defined as the way in which the firm creates and delivers value for the customer and how the firm appropriates the rents from the business [Ale08, S. 41].“

Das anschließende Kapitel Geschäftsmodellgrundlagen zeigt, welchen Wert Unternehmen für die Kunden generieren und wie sie dafür im Gegenzug einen Wert erhalten.

2.2.3. Geschäftsmodellgrundlagen

In der Vergangenheit fand der Großteil der Softwareentwicklung im kommerziellen Umfeld statt. Diese von Investoren vorfinanzierte Entwicklung diente der Erstellung eines kommerziellen Produktes, welches Gewinn bringend verkauft werden sollte. Diese Form der Entwicklung versteht man unter dem Geschäftsmodell **Software as a Product** [Eve08, S.11]. In den letzten Jahren hat sich die Sichtweise auf Software verändert. Immer häufiger wird Software als Dienstleistung angeboten **Software as a Service** [Eve08, S. 13]. Auf Grund der großzügigen Lizenzen im OSS Bereich, hat das etablierte Geschäftsmodell „Software as a Product“ oder „Software as a Service“ seine Probleme. Firmen sind daher auf der Suche nach anderen Geschäftsmodellen, mit denen sie wirtschaftlich arbeiten können [Eve08, S.17].

Eine Betrachtung der Prozesse im Unternehmen hilft bei dem Verständis der einzelnen Geschäftsmodelle. Anhand einer Wertschöpfungskette, können die einzelnen Prozesse im Unternehmen gegliedert werden [Lei04]. In Anlehnung an [Lei04] werden nachstehend die einzelnen acht Abschnitte der Wertschöpfungskette 2.2 eines allgemeinen IT-

Unternehmens vorgestellt. Die Wertschöpfungskette beginnt bei dem Prozess **Forschung**

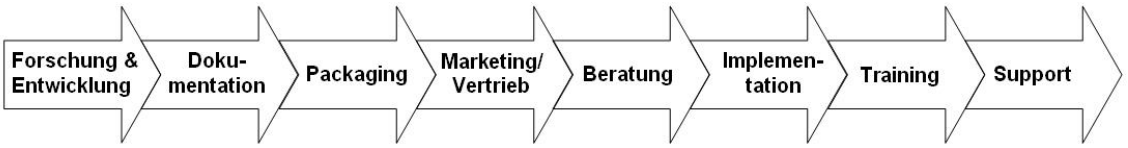


Abbildung 2.2.: Allgemeine Software Value Chain (Wertschöpfungskette) [Lei04]

und Entwicklung. Hier wird das Erstellen und Entwickeln von Software zusammen gefasst.

Im Anschluss an diesen Prozess folgt die **Dokumentation**. Die Dokumentation des Quellcodes passiert bereits im ersten Bereich. Unter Dokumentation ist hier das Erstellen von Informationen für den Einsatz der Software zu verstehen.

Das Zusammenfassen der einzelnen Bestandteile einer Software (Programme, Dokumentation, ...) in einem Paket, bezeichnet der Vorgang **Packaging**.

Die Phase **Marketing/Vertrieb** fasst die Vermarktungs- und Absatzaktivitäten eines Unternehmens zusammen. Ein Unternehmen definiert im Marketing, welche Produkte wie an wen verkauft werden.

Der Prozess **Beratung** bündelt eine Vielzahl an Aktivitäten eines Unternehmens zusammen. Dazu zählen zum Beispiel die Unterstützung der Kunden bei der Softwareinstallation, das Anfertigen von Studien, Analysen und Konzepten, sowie das Anpassen der implementierten Software an die Unternehmensprozesse.

Die Installation der Software vor Ort und alle dazugehörigen Prozesse, zum Beispiel das übernehmen von vorhandenen Datenbeständen, beinhaltet der Abschnitt **Implementation**.

Training beschreibt die Schulung des Kunden durch einen Dienstleister.

Unter **Support** versteht man die Betreuung der installierten IT-Systeme im laufenden Betrieb.

Unter Berücksichtigung der vorgestellten Wertschöpfungskette, werden im folgenden Kapitel Geschäftsmodelle im Bereich OSS vorgestellt.

2.2.4. Geschäftsmodelle im Bereich OSS

Nachstehend sind einige Geschäftsmodelle in Anlehnung an [Lei04] beschrieben, mit denen Unternehmen im OSS Bereich tätig sind. Inhalte die nicht von [Lei04] stammen, werden zitiert.

Distributoren

In einer Distribution werden verschiedene OSS Produkte zusammengefasst und aufeinander abgestimmt. Diese Kombination wird auf einem Datenträger inklusive Dokumentation ausgeliefert. Die Abbildung 2.3 zeigt die relevanten Prozesse der Distributoren in einer Wertschöpfungskette. Das Geschäftsmodell der OSS Distributoren besteht aus Entwicklung, Vermarktung, Vertrieb und Support der Distributionen. Beispiele für Distributoren sind SuSE (Deutschland) und RedHat(USA). Privatkunden und Unternehmen gehören zu den Zielgruppen der Distributoren. Für Unternehmen bieten einige Distributoren sogenannte „Professional-Version“ Distributionen an. Diese kostenpflichtigen Varianten unterscheiden sich häufig nur geringfügig von den kostenfreien Angeboten, beinhalten aber für die Unternehmen zusätzliche Angebote wie Service oder Schulungen.

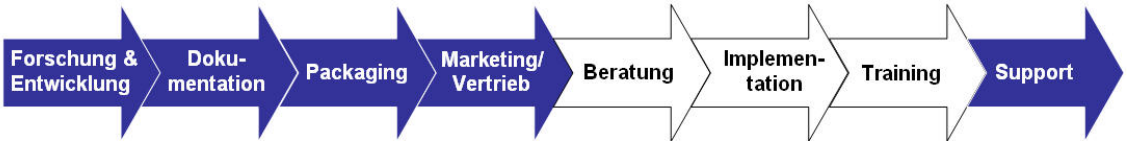


Abbildung 2.3.: Ressourcenfokus der Distributoren [Lei04]

OSS Applikationsanbieter

Die Wertschöpfung der Distributoren besteht darin, fremde Software in Pakte zusammenzustellen und diese dann zu Vertreiben. Bei einem OSS Applikationsabieter ist dies anders. Hierbei handelt es sich um Unternehmen, die selbst eigene Software entwickeln. Dabei kann man drei Fälle unterscheiden:

- Ein Unternehmen gibt den Quellcode einer zuvor proprietär entwickelten Software frei. Ein passendes Beispiel wurde in Kapitel 1.1 beschrieben. Netscape veröffentlicht den Quellcode des Netscape Navigators.

- Unternehmen entwickeln ab einem bestimmten Zeitpunkt eine Software unter eine OSS Lizenz. Die Lizenz ist zwar eine OSS Lizenz, dennoch dominiert das Unternehmen den Entwicklungsprozess, da keine bzw. wenige unabhängige Entwickler am Projekt beteiligt sind. Ein Beispiel hierfür ist RedHat.

- Ein OSS Projekt wird ab einem bestimmten Zeitpunkt von einem Unternehmen kommerziell betreut. Bei dieser „Projekt-Übernahme“ wechselt der Charakter des OSS Modell von frei auf kommerziell.

Die Wertschöpfungskette eines OSS Applikationsabieter zeigt die Abbildung 2.4. In der

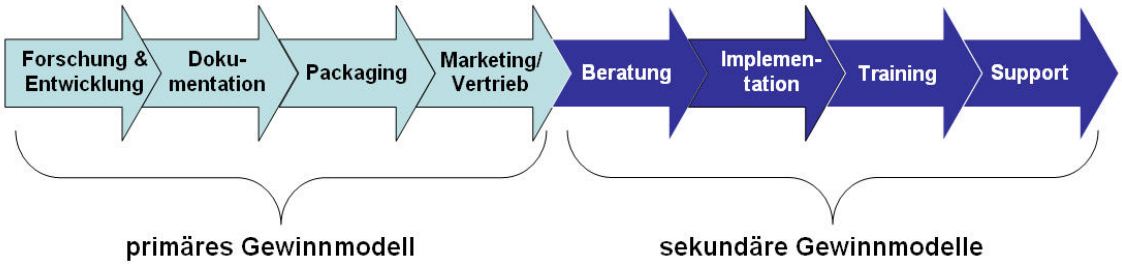


Abbildung 2.4.: Wertschöpfungskette der OSS Applikationsanbieter [Lei04]

Regel haben OSS Applikaionsanbieter keine Partner, die sich um den Vertrieb der Software kümmern. Die Software wird direkt über das Internet vertrieben. Im sekundären Gewinnmodell würden eventuelle Dienstleistungspartner konkurrieren.

Appliances

Unter einer Appliance, oder auch Computer Appliance genannt, versteht man einen Computer, der darauf ausgelegt ist, eine eng begrenzte Funktionalität zu bieten. Für den Anwender sind nur wenige Eingriffe für die Bedienung erforderlich [Intq]. Ein Beispiel für ein Appliance wäre ein DVD-Player, bei dem der Kunde das Betriebssystem nicht mehr wahrnimmt. Bei einigen Geräten kommt das Betriebssystem Linux zum Einsatz. Eigens programmierte Benutzerschnittstellen sind meistens proprietär. Abbildung 2.5 zeigt die Wertschöpfungskette eines Appliance-Herstellers. In der Regel wird der Support der

Abbildung 2.5.: Wertschöpfungskette der Appliance-Anbieter [Lei04]

Appliance über Partner abgewickelt. Dennoch gehört Support zur Wertschöpfungskette der Appliance-Hersteller, da die Partner Wartungsverträge abschließen können. Weitere Beispiele für Appliance sind Thin clients von IBM Net Vista, Linware, Whyse oder Set top boxen von Nokia, PersonalTV oder TiVo.

Dienstleister

Bei den bisher beschriebenen Geschäftsmodellen, handelt es sich um sogenannte Produkt-Geschäftsmodelle. Der Schwerpunkt bei diesen Geschäftsmodellen liegt auf der Entwicklung eines Softwareproduktes unter einer OSS Lizenz, sowie die Vermarktung und der Vertrieb der Software bzw. des Produktes. Bei Open Source Software Dienstleistern handelt es sich um Firmen, die kein eigenes OSS Produkt anbieten. Diese Unternehmen haben sich auf Support und Service rund um OSS spezialisiert. Abbildung 2.6 zeigt die Wertschöpfungskette eines Dienstleisters. Auf der Grundlage ist dieses Geschäftsmodell das am wenigsten umstrittene, da es fast keine Unterschiede zum herkömmlichen Geschäftsmodell gibt. Vergleicht man Dienstleister für proprietäre Software mit OSS Dienstleistern, so sind diese fast identisch.

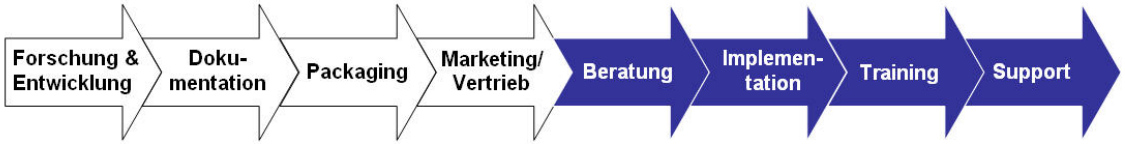


Abbildung 2.6.: Wertschöpfungskette der OSS Dienstleister [Lei04]

Mediator

Im Umfeld von OSS bringen Mediatoren verschiedene Interessengruppen mit Hilfe eines Marktplatzes zusammen. Bei den Interessengruppen handelt es sich zum Beispiel um Entwickler, Nutzer und Dienstleister. Ein Beispiel für einen Mediator ist SourceForge.

> SourceForge.net is the world's largest open source software development web site. We provide free services that help people build cool stuff and share it with a global audience [Intp].

Mediatoren betreiben einen Marktplatz und erzielen Erlöse durch den Verkauf von Bannern an Werbetreibende oder durch den Vertrieb von Büchern und Datenträgen (sogenannte Commerce-Produkte).

Sekundärmärkte

Bei dieser Form des Geschäftsmodells gibt der Anbieter die OSS komplett kostenlos an den Kunden ab. Der Anbieter muss sich einen alternativen Markt für seinen finanziellen Erfolg kreieren [Lei04].

Hardware-Enablement

In erster Linie konzentriert man sich bei diesem Geschäftsmodell auf den Vertrieb von Hardware. Direkt zum Verkauf der Hardware liegt aber ein Open Source Betriebssystem als so genannter „Preload“ bei. Dieses Geschäftsmodell ist auch als „Widget Frosting“

bekannt. Dieser Begriff geht auf Raymond zurück. Er verwendete diesen Begriff in seinem Paper „The Magic Cauldron“.

> This model is for hardware manufacturers (hardware, in this context, includes anything from Ethernet or other peripheral boards all the way up to entire computer systems). Market pressures have forced hardware companies to write and maintain software (from device drivers through configuration tools all the way up to the level of entire operating systems), but the software itself is not a profit center. It’s an overhead – often a substantial one [Ray].

Die Abbildung 2.7 zeigt die Wertschöpfungskette eines Hardware-Enablement.

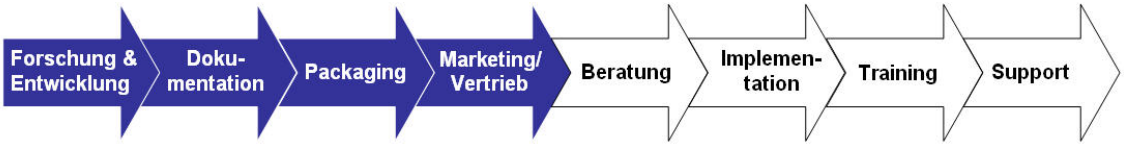


Abbildung 2.7.: Wertschöpfungskette Hardware-Enablement [Lei04]

Loss-Leader

Bei diesem Geschäftsmodell wird ein OSS Produkt, mit dem Ziel potentielle Kunden auf sich aufmerksam zu machen, als Lockmittel angeboten. Wenn der Kunde auf das Unternehmen aufmerksam geworden ist, wird im Zuge dessen meist ein teures Produkt verkauft.

> In this model, you use open-source software to create or maintain a market position for proprietary software that generates a direct revenue stream. In the most common variant, open-source client software enables sales of server software, or subscription/advertising revenue associated with a portal site [Ray].

Die Abbildung 2.8 zeigt die Wertschöpfungskette eines Loss-Leader 2.8.

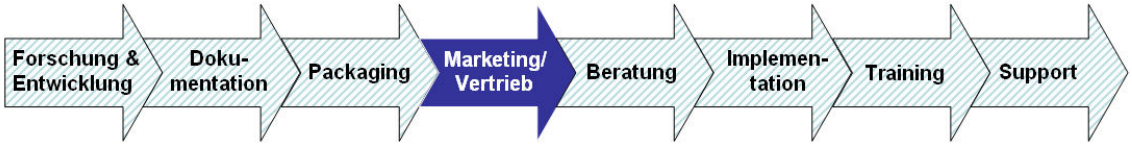


Abbildung 2.8.: Wertschöpfungskette Loss Leader [Lei04]

Hybridmodelle

Werden mehrere Geschäftsmodelle miteinander kombiniert, so spricht man von Hybridmodellen. Denkbar wäre zum Beispiel ein Distributor, der auch Service und Support für seine Distribution und andere OSS Produkte anbietet.

Dual Licensing

Ein und das selbe Open Source Produkt wird unter verschiedenen Lizenzen veröffentlicht [ABMS08]. Ein Beispiel für ein Unternehmen, welches Dual Licensing einsetzt, ist MySQL [ABMS08] [Webj]. MySQL wird in einer Version sowohl unter der GPL als auch gleichzeitig unter einer kommerziellen Lizenz veröffentlicht. Die Community profitiert von der Weiterentwicklung des Produktes durch die Mitarbeiter von MySQL. Für die Kunden bietet die kommerzielle Variante professionellen Support und eine Befreiung der Veröffentlichungspflicht des Quellcodes bei unternehmensindividuellen Anpassungen [ABMS08]. Die Abbildung 2.9 zeigt die Wertschöpfungskette des Dual Licensing.

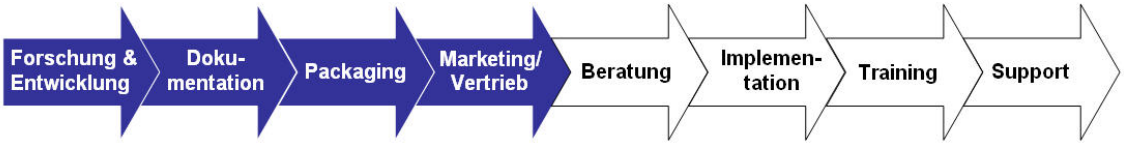


Abbildung 2.9.: Wertschöpfungskette Dual Licensing [Lei04]

Anmerkung zu den Grafiken 2.2 bis 2.9

Die blau bzw. dunkel markierten Pfeile sind Bestandteil der Wertschöpfungs-

kette. Die weißen bzw. hellen Pfeile sind nicht Bestandteil der Wertschöpfungskette.

Bevor in einem Beispiel eine OSS einem Geschäftsmodell zugeordnet wird, müssen erst die Vielzahl an Begriffen, die Autoren für die Begriffsdefinitionen im Bereich der OSS verwenden, geordnet werden. Das nächste Kapitel führt ein Geschäftsmodellzuordnungsverfahren ein.

2.2.5. Geschäftsmodellzuordnungsverfahren

Das Geschäftsmodellzuordnungsverfahren besteht im Wesentlichen aus einer Grafik, in der die einzelnen Schwerpunkte eines Geschäftsmodelles zugeordnet sind. Betrachten wir die oben angegebene Beschreibung von Distributoren von [Lei04]. Daraus geht hervor, dass ein Distributor verschiedene OSS Produkte vermarktet und eine Dokumentation mitliefert. Andere Autoren ordnen dem Distributor ebenfalls Vermarktung und Distribution, aber auch Wartungsverträge, Entwicklung und Support zu. Andere Autoren sehen die Distribution und die Entwicklung eher beim Geschäftsmodell Support Seller. Die Grafik 2.10 zeigt, wie die eben angesprochenen Begriffe einander zugeordnet werden. Es ist gut zu erkennen, dass es bei den Geschäftsmodellen Überschneidungen gibt. Die Abbildung 2.11 fasst die einzelnen Definitionen und deren Zuordnungen der Quellen [KA04] [Lei04] [Kri05] [Ale08] [Eve08] zusammen.

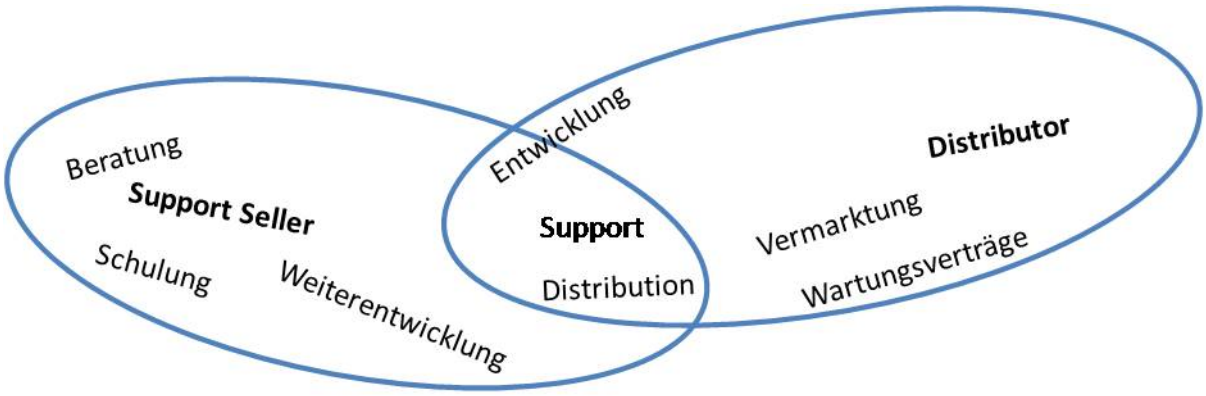


Abbildung 2.10.: Ausschnitt aus Abbildung 2.11

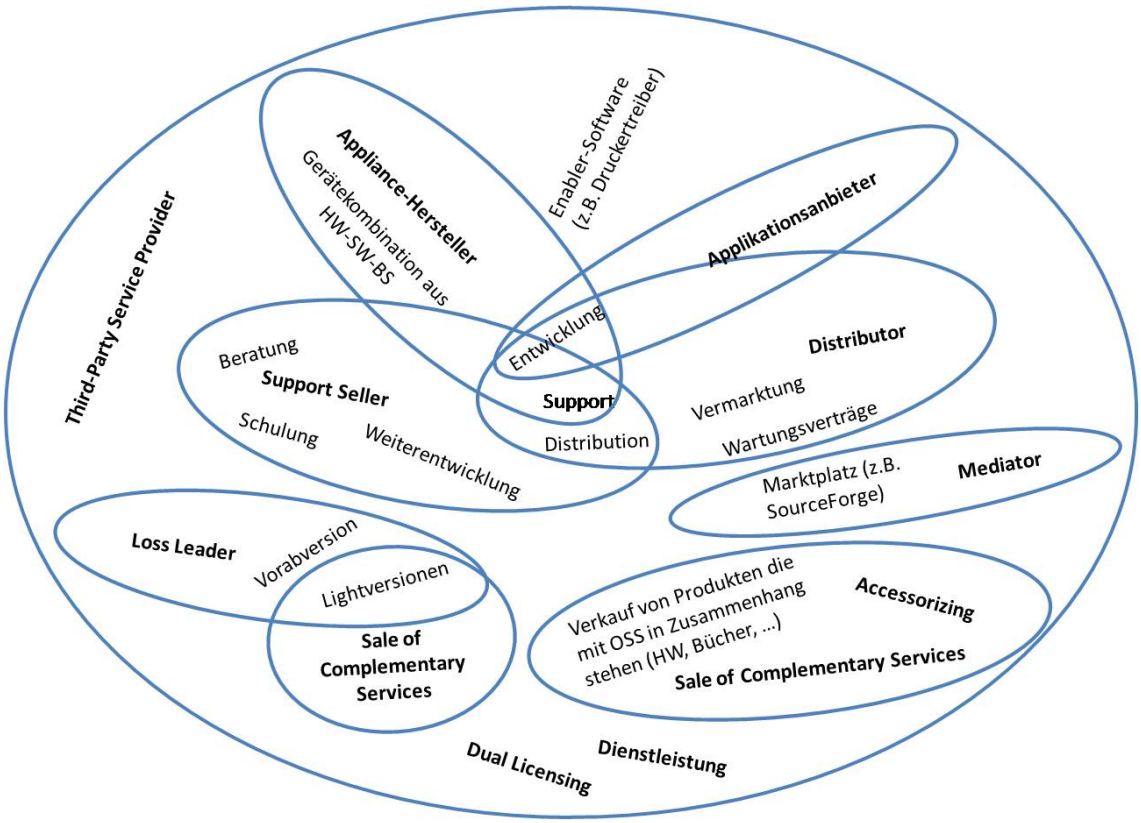


Abbildung 2.11.: Begriffsübersicht Geschäftsmodell (Darstellung des Autors)

Bei der Zuordnung von OSS zu Geschäftsmodellen (s. Kapitel 5 S. 69) wird diese Veranschaulichung in Abbildung 2.11 herangezogen. Um dies zu verdeutlichen betrachten wir das Produkt MySQL von Sun Microsystems. Ein Blick auf die Webseiten von MySQL [Webj] zeigt, das MySQL einige Produkte rund um die eigene Datenbank anbietet. So werden neben dem freien Comunity Datenbank Server auch eine kostenpflichtige Enterprise Version angeboten. Die Enterprise Version unterstützt mehr Funktionen und wird mit verschiedenen Services, zum Beispiel regelmäßige Updates, angeboten. Schulungen und Beratungen für die MySQL Datenbank runden das Angebot ab. Mit Hilfe der Zuordnungsgrafik lassen sich dem Unternehmen Sun Mycrosystems folgende Geschäftsmodelle zuordnen:

Geschäftsmodell	Begründung
Loss Leader	Nach [Lei04] sind Loss Leader Unternehmen, die Kunden mit einer freien Software auf ihr Unternehmen aufmerksam machen, um ihnen dann ein kostenpflichtiges Produkt zu verkaufen. Bereits in Abschnitt 1.2 wurde darauf hingewiesen, dass kostenfreie Vermarktung der MySQL Datenbank dem kommerziellen Geschäft hilft. Unternehmen, die den kostenfreien Comunity MySQL Server im Unternehmen einsetzen, werden auf Sun Microsystems aufmerksam und werden dann vielleicht den kostenpflichtigen Enterprise Server einsetzen.
Support Seller	Zum Produkt MySQL werden Beratung und Schulung angeboten.
Dual Licensing	MySQL gibt es mit zwei Lizenzen. Einmal steht MySQL unter der GPL und gleichzeitig wird es proprietär mit erweiterten Funktionen in der Enterprise Version verkauft.

Tabelle 2.2.: Geschätsmodelle MySQL

Im nächsten Abschnitt wird die Programmanalyse vorgestellt. Mit Hilfe dieser Methode, lässt sich eventuell ein Zusammenhang zwischen Geschäftsmodellen und OSS anhand des Quellcodes herstellen.

2.3. Programmanalyse

Reverse Engineering ist der Prozess, bei dem gegebene Systeme analysiert werden. Dabei müssen die einzelnen Komponenten und deren gegenseitige Beziehung zueinander identifiziert werden. In einem ersten Schritt wird der vorhandene Quellcode analysiert und aufbereitet. Die Ergebnisse werden in einem Repository gespeichert. Aufbauend auf diesen Daten kann dann eine explizite Repräsentation in anderer Form, zum Beispiel auf einer höheren Abstraktionsstufe, generiert werden [ERW08].

Ein Programm, mit dem eine Programmanalyse durchgeführt werden kann, wird an der Universität Koblenz Landau im Institut für Softwaretechnik entwickelt. Dieses Toll heißt Generische Umgebung zum Programmverstehen (GUPRO). Eine zentrale Funktionalität ist das Stellen von Anfragen an die im Repository gespeicherte Software. Durch die Repräsentation des Quellcodes mittels TGraphen sind den Anfragen kaum Grenzen gesetzt [Ebe98]. Bei den TGraphen handelt es sich um typisierte, attributierte, angeordnete und gerichtete Graphen [EWD+08]. Die Analyse mittels GUPRO findet auf einer sehr fein granularen Ebene statt. Für die Forschungsfrage „Gibt es einen Zusammenhang von Geschäftsmodellen und OSS“, wäre diese Analyse zu detailliert.

Die Programmanalyse kann man auf verschiedenen Ebenen durchführen. MacCormack und Rusnak [MRB06] unterscheiden drei Ebenen:

- **„subsystem level“**
  Quellcode Dateien (Source Files), die untereinander zusammenhängen und zu einem bestimmten Teil des Designs gehören.

- **„source file level“**
  Eine Ansammlung inhaltlich zusammenhängender Gruppen von Funktionen oder Programminstruktionen

- **„function level“**
  Einen Teil von Programminstruktionen, die eine sehr hoch spezifizierte Aufgabe erfüllen.

Anmerkung zum „subsystem level“
Der Programmierer hat eine bestimmte Sicht auf sein System und ordnet die Quellcodedateien in einer von ihm erdachte Verzeichnisstruktur ab. Diese Subsysteme werden durch die Kästchen in einer entsprechenden Grafik abgebildet. Die beiden Abbildungen 2.17 und 2.18 zeigen die Sicht des Programmierers (Verzeichnisstruktur) und gleichzeitig eine grafische Darstellung der Software. Diese Form der Darstellung wird in Kapitel 2.3.2 vorgestellt und eingeführt.

Aufbauend auf den Arbeiten von MacCormack und Rusnak [MRB06] wird die Struktur der OSS auf Ebene der „source file level“ analysiert. In Kapitel 2.3.2 wird darauf erneut eingegangen.

In Kapitel 4 werden verschiedene Analyseprogramme vorgestellt. Bevor die Programme im Detail erklärt werden, müssen noch einige Grundlagen behandelt werden. Die beiden nachstehenden Abschnitte 2.3.1 und 2.3.2 zeigen wie die Programme arbeiten und welche Ergebnis diese liefern.

2.3.1. Call Graph

Für die Analyse von Programmen kann man die Technik der Call Graphen verwenden. Ein Call Graph repräsentiert die Abhängigkeiten zwischen Quellcodedateien (engl. Sourcefiles). Die Abhängigkeit zwischen mindestens zwei Dateien kann man feststellen, in dem man Funktionsaufrufe analysiert. MacCormack und Rusnak (et. al. 2006) sprechen von einem „Function Call“ [MRB06]. In Anlehnung an MacCormack und Rusnak (et. al. 2006) zeigt das folgende Listing 2.1 ein kleines Beispiel Programm.

Es bleibt zu erwähnen, das es sich bei diesem Beispielprogramm um kein vollständiges Programm handelt, sondern um einen Programmausschnitt. Dennoch verdeutlicht dieser Programmausschnitt die Abhängigkeit zwischen den Funktionen „funcA“ und „funcB“. Speichert man die beiden Funktionen in unterschiedlichen Dateien ab, so ergibt sich eine Abhängigkeit zwischen diesen beiden Dateien. Funktion „funcA“ wird in „File 1“ und

```
1  function funcA;
2  begin
3    write 'I call function B';
4    funcB;
5  end
6
7  function funcB;
8  begin
9    write 'I am function B';
10 end
```

Listing 2.1: Pseudocode Funktionsaufruf

Funktion „funcB“ wird in „File 2“ abgespeichert. Abbildung 2.12 stellt die entstandene Abhängigkeit zwischen den beiden Dateien grafisch dar.

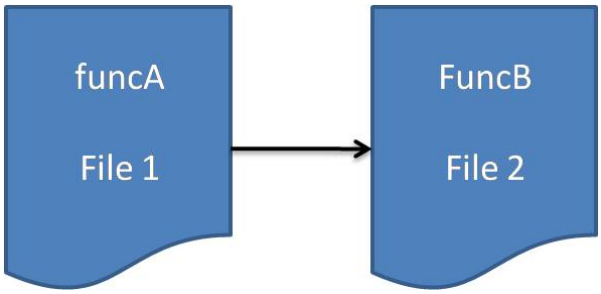


Abbildung 2.12.: Call Graph

Gründe für die Trennungen von Funktionen in verschiedene Dateien gibt es viele. Allgemein kann man sagen, dass ein schönes Softwaredesign verfolgt wird. Einen Einblick in diese Thematik bieten Brössler und Siedersleben in ihrem Buch [BS00]. Kapitel 7 (S. 95) erklärt was unter einer schönen Softwarearchitektur zu verstehen ist. Als weiterführende Literatur sei hier der Band von Helmut Balzert „Lehrbuch der Software-Technik“ genannt.

Eine weitere Form der Darstellung von Abhängigkeiten bietet die Design Structur Matrix (DSM). Für eine spätere Analyse und Strukturerkennung ist diese Darstellungsform der eben gezeigten vorzuziehen. Was man unter einer DSM versteht und wie man diese aus den Call Graphen generiert, zeigt das nächste Kapitel.

2.3.2. Design Structure Matrix

Das Management von komplexen Systemen ist eine wichtige Kernkompetenz für ein erfolgreiches Unternehmen. Die Design Structure Matrix (DSM) ist ein einfaches Werkzeug um dies zu erreichen. Die DSM ist auch bekannt als:

- The Dependency Structure Matrix
- The Problem Solving Matrix (PSM)
- Design Precedence Matrix

Die DSM kann für das Management und für die Analyse von komplexen Systemen verwendet werden [Intc]. Sie unterstützt eine kompakte Repräsentation eines komplexen Systems, in dem die Abhängigkeiten zwischen den einzelnen Elementen der Systeme grafisch dargestellt werden. Eine DSM ist eine quadratische Matrix. Die diagonalen Zellen (links oben nach rechts unten) bleiben leer. D_{ij} repräsentiert den Wert aus der Zelle in Reihe i und Spalte j [MMW09].

Die DSM betrachtet vier verschiedene Arten von Aktivitäten [OHFA04]:

1. Sequential (dependant)
2. Parallel or concurrent (independent)
3. Coupled (interdependent)
4. Contigent (conditional)

In Anlehnung an Oloufa und Hosni (et. al. 2004) sowie MacCormack und Rusnak (et. al. 2006) [OHFA04] [MRB06] zeigen die im Anschluss folgenden vier Abbildung 2.13 bis 2.16 einen Call Graph und eine entsprechende Darstellung in DSM für die oben genannten vier Arten von Aktivitäten.

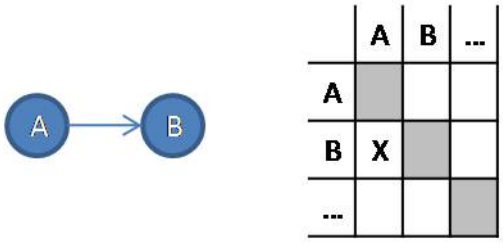


Abbildung 2.13.: Call Graph und DSM. Type: Sequential

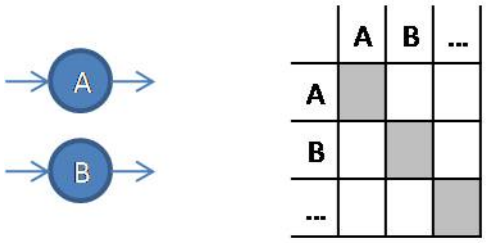


Abbildung 2.14.: Call Graph und DSM. Type: Parallel

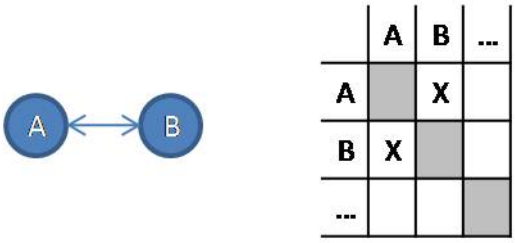


Abbildung 2.15.: Call Graph und DSM. Type: Coupled

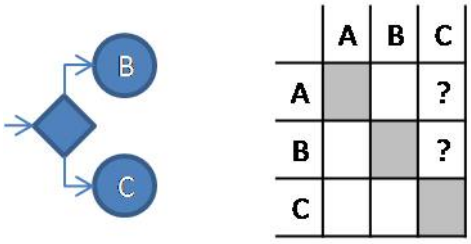


Abbildung 2.16.: Call Graph und DSM. Type: Condtional

Je nach Ebene, auf der Software analysiert wird, fällt die DSM mehr oder minder komplex aus. Bereits im Abschnitt Programmanalyse 2.3 wurden drei Ebenen der Programmanalyse vorgestellt. Die Granularität sollte zur Forschungsfrage passen. Die Analyse der OSS findet auf der Ebene „source file level“ statt. Die Abhängigkeiten zwischen den einzelnen Dateien (Source Files), ermittelt durch die oben vorgestellten „Call Graphen“, werden mittels DSM dargestellt.

> **Anmerkung zur Granularität**
>
> Wenn man die Funktionsweise eines Automobils als Ganzes verstehen möchte, muss man sich mit den Systemkomponenten („subsysteme“) (source file level) auseinandersetzen z.B. Räder, Motor, Getriebe, ... In diesem Kontext macht es wenig Sinn, sich über die Beschaffenheit der verwendeten Bolzen und Schrauben zu informieren [In Anlehnung an] [MRB06].

Die Grafiken 2.17 [2] und 2.18 [3] zeigen zum einen eine Ansicht einer sehr frühen Version des Linux Kernels in einer Verzeichnis Struktur und zum anderen die Darstellung in einer DSM. Die einzelnen schwarzen Punkte in der Grafik 2.18 zeigen die Abhängigkeiten der Quellcodedateien und die dünnen Kästchen außen herum stellen die Subsysteme dar. Bei einem Vergleich der Verzeichnisstruktur mit der DSM wird der Zusammenhang der Subsysteme und der Dateien ersichtlich.

Ein weiterer Vorteil der DSM ist das Berechnen von Metriken, mit denen Aussagen über das Softwareprodukt getroffen werden können. Zwei wichtige Metriken werden im nachfolgenden Abschnitt beschrieben.

2.3.3. Metriken

Damit man detaillierte Aussagen und Auswertungen aus den DSM gewinnen kann, verwendet man so genannte Metriken. In Zusammenhang mit DSM werden nachstehend zwei Metriken vorgestellt. Zum einen die so genannte „Propagation cost“, diese findet

[2] Die Abbildung stammt aus dem Programm Understand, welches in Kapitel 4.1 vorgestellt wird.
[3] Die Abbildung stammt aus dem Programm CsvToDms, welches in Kapitel 4.1 vorgestellt wird.

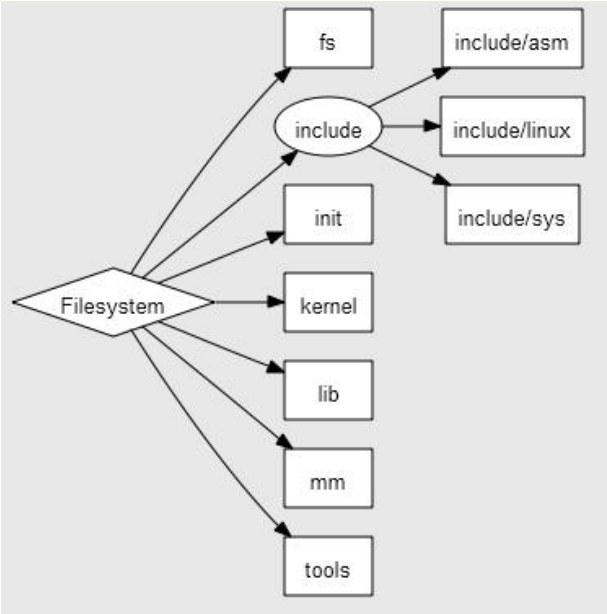


Abbildung 2.17.: Dateisystem Linuxkernel (Abbildung aus dem Programm Understand 2.0)

auch bei der späteren Analyse Verwendung und zum Vergleich und besseren Verständnis der „Propagation cost“ wird die so genannte „Clustered cost“ vorgestellt.

Propagation cost

Die Propagation cost misst das Ausmaß der Beeinflussung bei Veränderung eines Elementes auf ein anderes. So wird der Grad der Verbindungen ohne Betrachtung der Nähe zwischen den Elementen repräsentiert. Die Propagation cost berücksichtigt einfach die Existenz einer Abhängigkeit ohne Rechnung zu tragen, ob sich die einzelnen Elemente zum Beispiel im selben Subsystemen befinden [MMW09].

In Anlehnung an [MMW09] und [MRB06] wird die Propagation cost wie folgt definiert:

Wie in Kapitel 2.3.2 gezeigt wurde, enthalten die Zellen der DSM Werte. Für die nachstehende Betrachtung werden diese Werte reduziert. Das bedeutet das eine Zelle den Wert 1 hat, wenn Sie einen Wert enthält oder 0 wenn sie keinen Wert enthält. Man könnte hier auch von einer binären DSM sprechen. Alternativ kann man sich diese Matrix auch

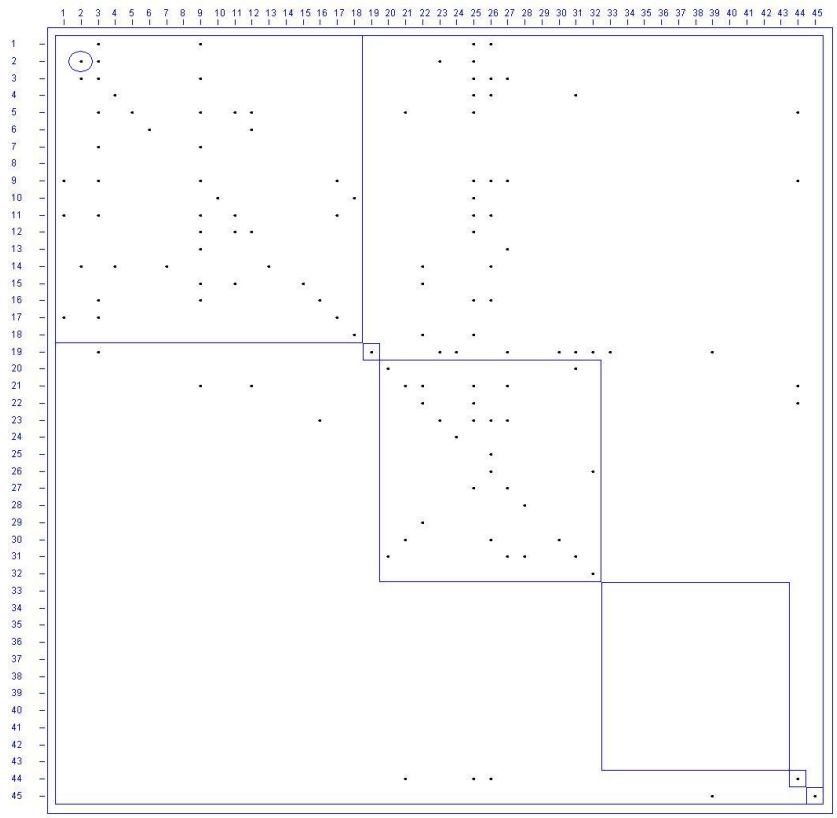


Abbildung 2.18.: DSM des Linuxkernel (Abbildung aus dem Programm CsvToDSM

als Matrix mit direkten Abhängigkeiten vorstellen. Dies entspräche der Pfadlänge 1.

$$D^1 = DSM = \begin{pmatrix} d_{11} & d_{12} & \cdots & a_{1N} \\ d_{21} & d_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N1} & d_{N2} & \cdots & a_{NN} \end{pmatrix} \tag{2.1}$$

Indirekte Abhängigkeiten können stufenweise angezeigt werden, in dem man die Pfadlänge schrittweise erhöht. Die Abbildung 2.19, auf die im Anschluss an die theoretische Erläuterung in einem Beispiel eingegangen wird, zeigt das im unteren Teil.

$$D^i = D \times D^{i-1} \qquad \text{if i} > 1 \tag{2.2}$$

Die Sichtbarkeitsmatrix entspricht der Summe all dieser Matrizen, die durch das schritt-

weise Erhöhen der Pfadllänge entstanden sind.

$$V = \sum_{i=0}^{n} D^i = \begin{pmatrix} d_{11} & d_{12} & \cdots & a_{1N} \\ d_{21} & d_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N1} & d_{N2} & \cdots & a_{NN} \end{pmatrix} \tag{2.3}$$

Damit man binäre Werte erhält, reduziert die folgende Funktion die Werte der DSM auf 1 bzw. auf 0. Für eine spätere Betrachtung geht es hier allein um die Existenz oder nicht Existenz einer Abhängigkeit.

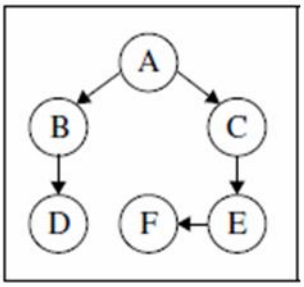
$$V' = f(V) = \begin{cases} v'_{ij} = 0 & \text{if } v_{ij} = 0 \\ v'_{ij} = 1 & \text{if } v_{ij} > 0 \end{cases} \tag{2.4}$$

Die Propagation Cost entspricht der Summe der Sichtbarkeitsmatrix V', geteilt durch das Quadrat von N. N entspricht dabei der Anzahl der einzelnen Elemente.

$$\text{Propagation Cost} = \frac{\sum_{i=0}^{N} \sum_{j=0}^{N} v'_{ij}}{N^2} \tag{2.5}$$

Damit gibt die Propagation Cost Auskunft über das Ausmaß der Elemente, die im Durchschnitt direkt oder indirekt betroffen sein können, wenn ein Element verändert wird. Um den Sachverhalt zu verdeutlichen, wird das ganze an einem Beispiel erklärt. Das Beispiel entstand in Anlehnung an [MRB06]. Die Grafik 2.19 zeigt zum einen in der oberen Hälfte ein Beispielsystem und zum anderen in der unteren Hälfte die einzelnen Abhängigkeiten der Quellcodedateien. M^0 bedeutet die Pfadlänge = 0. Man setzt voraus, dass jede Datei in sich selbst abhängig ist. M^1 zeigt die Abhängigkeiten für die Pfadlänge = 1. Die letzte Matrix mit der Überschrift $V = \sum M^n; n = [0, 4]$ zeigt die so genannte Sichtbarkeitsmatrix (engl. Visibilitymatrix). Sie stellt die Summe aller vorher gezeigten Matrizen da. [MRB06] Aus dieser Sichtbarkeitsmatrix lässt sich dann die Propagation Cost berechnen.

$$\text{Propagation Cost} = \frac{\sum_{i=0}^{N} \sum_{j=0}^{N} v'_{ij}}{N^2} = \frac{15}{6^2} = \frac{15}{36} = 0,42 \tag{2.6}$$



M^0

	A	B	C	D	E	F
A	1	0	0	0	0	0
B	0	1	0	0	0	0
C	0	0	1	0	0	0
D	0	0	0	1	0	0
E	0	0	0	0	1	0
F	0	0	0	0	0	1

M^1

	A	B	C	D	E	F
A	0	1	1	0	0	0
B	0	0	0	1	0	0
C	0	0	0	0	1	0
D	0	0	0	0	0	0
E	0	0	0	0	0	1
F	0	0	0	0	0	0

M^2

	A	B	C	D	E	F
A	0	0	0	1	1	0
B	0	0	0	0	0	0
C	0	0	0	0	0	1
D	0	0	0	0	0	0
E	0	0	0	0	0	0
F	0	0	0	0	0	0

M^3

	A	B	C	D	E	F
A	0	0	0	0	0	1
B	0	0	0	0	0	0
C	0	0	0	0	0	0
D	0	0	0	0	0	0
E	0	0	0	0	0	0
F	0	0	0	0	0	0

M^4

	A	B	C	D	E	F
A	0	0	0	0	0	0
B	0	0	0	0	0	0
C	0	0	0	0	0	0
D	0	0	0	0	0	0
E	0	0	0	0	0	0
F	0	0	0	0	0	0

$V = \Sigma M^n ; n = [0,4]$

	A	B	C	D	E	F
A	1	1	1	1	1	1
B	0	1	0	1	0	0
C	0	0	1	0	1	1
D	0	0	0	1	0	0
E	0	0	0	0	1	1
F	0	0	0	0	0	1

Abbildung 2.19.: Beispielsystem mit Sichtbarkeitsmatrix

Clustered Cost

Bei der Clustered Cost handelt es sich um eine besser entwickelte Metrik, welche verschiedene Kosten in Abhängigkeit der Stelle der Elemente im Cluster, sprich in der DSM, berücksichtig. Dies steht im Kontrast zur Propagation Cost, die nur berücksichtigt, dass es eine Abhängigkeit gibt [MMW09].

In Anlehnung an [MMW09] und [MRB06] wird die Propagation Cost wie folgt definiert:

Die Clustered Cost misst unter Berücksichtigung unterschiedlicher Kosten, je nach Lage des Elementes, die Abhängigkeiten. Elemente, die in einem Subsystem (Cluster) sind, haben geringere Abhängigkeitskosten, als Elemente, die nicht im selben Subsystem sind. Eine weitere Unterscheidung, die man trifft, ist, ob ein Element ein „vertical bus“ ist. Einige Elemente in einem Softwaresystem beinhalten Funktionen, die von einer sehr hohen Anzahl anderer Elemente verwendet werden. Die Abbildung 2.20 zeigt einen Ausschnitt aus einer DSM des Linux Kernels Version 2.1.105. Die Umrandung kennzeichnet den „vertical bus“.

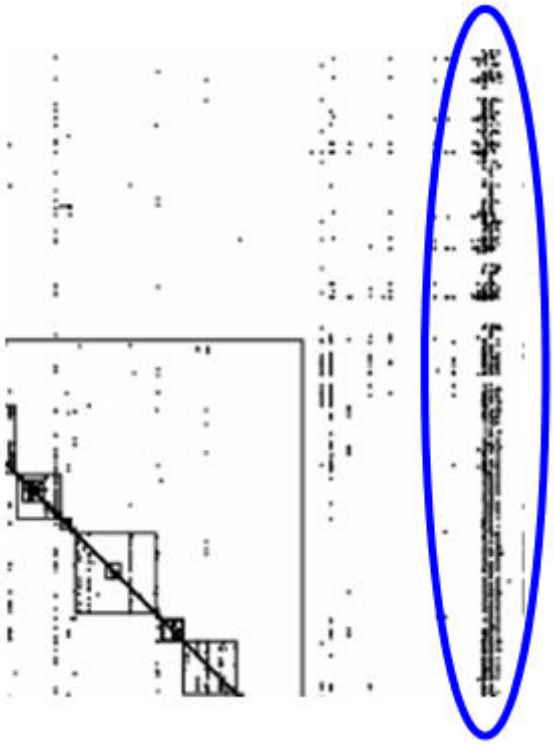


Abbildung 2.20.: Vertical Bus
[MRB06]

Anmerkung zum „vertical bus“
Ein Beispiel für ein Element, welches zu einem „vertical bus“ führt, wäre zum Beispiel eine Funktion für die Fehlerbehandlung. Viele Elemente greifen auf diese Routinen zurück, um entsprechende Fehlermeldungen zu generieren.

Wenn man diese Sachen berücksichtigt, ergibt sich folgende Formel für die Abhängig-

keitskosten (Dependency Cost):

$$\text{Dependency Cost}(i,j) = \begin{cases} n^{\lambda} d_{ij} \text{ if i and j are in the same cluster} \\ N^{\lambda} d_{ij} \text{ if i and j are not in the same cluster} \\ d_{ij} \text{ if j is a vertical bus} \end{cases} \tag{2.7}$$

N ist die Anzahl der Elemente der Matrix bzw. DSM.
n ist die Anzahl des kleinsten Subsystems bzw. Clusters welches i und j enthält.
λ ist ein vom Benutzer zu definierender Parameter. (Die Literatur [MMW09]+ schlägt einen Wert von $\lambda = 2$ ohne Angabe von Gründen vor).

Die Aggregation der Abhänigkeitskosten (Dependency Cost) zwischen allen Elementen in der DSM ergeben die Clustered Cost:

$$\text{Clustered Cost}(i,j) = \sum_{i=0}^{N} \sum_{j=0}^{N} \text{Dependency Cost}(i,j) \tag{2.8}$$

Eine wichtige Einschränkung ist, dass diese Metrik nur mit Metriken aus DSM mit gleichen Größen vergleichbar ist [MMW09].

Anmerkung zur Clustered Cost bezüglich der Verwendbarkeit
Mit dieser Metrik kann man zum Beispiel eine DSM in eine nahezu idealisierte modulare Form bringen. Dies geschieht, indem man die Elemente so anordnet, dass die Abhängigkeiten möglichst gering sind [MMW09] [MRB06]. Die Abbildung 2.21 zeigt eine DSM in einer solchen idealisieren Form.

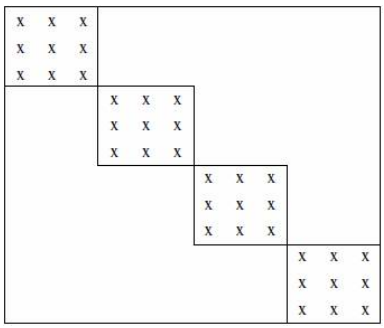


Abbildung 2.21.: Idealisierte Form einer DSM
[MRB06]

3. Forschungsfragen

Ziel der Arbeit ist es festzustellen, ob es einen Zusammenhang zwischen OSS und Geschäftsmodellen gibt. Um dieser Fragestellung nachgehen zu können, müssen neben den im vorangegangenen Kapitel 2 genannten Grundlagen noch einige Punkte geklärt werden. Zuerst wird auf die Untersuchungsvoraussetzung eingegangen. Dieser Abschnitt macht noch einmal den Zusammenhang zwischen Sourcecode, DSM und Geschäftsmodell deutlich. Abschließend werden Thesen und Fragestellungen genannt, mit denen sich diese Arbeit auseinandersetzen muss, um dem Ziel näher zu kommen.

3.1. Untersuchungsvoraussetzung

Eine Herangehensweise bei der Entwicklung komplexer Produkte ist das Zerlegen des Produktes in Systeme und das weitere Zerlegen dieser Systeme in einzelne Komponenten [SER03]. Diese Herangehensweise hilft komplexe Probleme und Sachverhalte zu lösen [BC99]. Baldwin und Clark schreiben in ihrem Buch, Design Rules - The Power of Modularity, dass Firmen ein unvorstellbares Level an Innovationen und Komplexität erreichen, indem sie das Konzept der Modularität anwenden. Man erstellt komplexe Produkte ausgehend von kleineren Subsystemen, die unabhängig voneinander entwickelt werden können, später aber als ganzes zusammenarbeiten [BC00]. Ein Beispiel, was zum Verständnis der Modularität beiträgt, nennt die Anmerkung zur Granularität in Kapitel 2.3.2. Bei Softwareprodukten wird ebenfalls mit dieser Methode gearbeitet. Dabei kann man für die Architektur folgende Aussage treffen:

> The architecture of a product is the scheme by which the functions it

performs are allocated to ist consitutent compoments [Ulr95].

Programmanweisungen und Funktionen sind Bestandteile von Programmen. Unter dem Begriff Architektur versteht man das Schema, also die Art und Weise wie diese Elemente zusammengefasst sind. Damit man die einzelnen Produktarchitekturen charakterisieren kann, benötigt man das Konzept der Modularität [MRB08]. Dieses hat sich in einer Vielzahl von verschiedenen Feldern, welche mit komplexen Systemen handeln, bewährt [BC99]. Modularität ist eine besondere Designstruktur, in welcher Parameter und Aufgaben innerhalb von Einheiten (Modulen) voneinander abhängig sind und darüber hinaus unabhängig. Für jedes Design kann die Eigenständigkeit oder die gegenseitige Abhängigkeit von Elementen mittels einer DSM bestimmt werden [BC99]. Ein Design ist eine komplette Beschreibung der bautechnischen Elemente einer Software [BC99]. Hierarchische Beziehungen und Abhängigkeiten können formal mit einer DSM abgebildet werden. Dieser Abbildungsprozess wurde von Donald Steward erfunden und wurde von Steven Eppinger weiterentwickelt [BC99]. Eine DSM hebt die innewohnende Struktur eines Design hervor. Die Abhängigkeiten zwischen den einzelnen Elementen werden in einer quadratischen Matrix dargestellt [Ste81] [EWSG94] [SER03].

Mit Hilfe der Modularität und der Technik DSM wird versucht, einen Zusammenhang zwischen OSS und Geschäftsmodellen zu finden, in dem man die einzelnen OSS, denen ein Geschäftsmodell zugeordnet wird, mit einer anderen OSS und deren Geschäftsmodell vergleicht. Möglich ist dies, da der Code eine Struktur widerspiegelt, die man mit Hilfe des Konzeptes der Modularität und DSM vergleichen kann. Das Ziel dieser Diplomarbeit ist es festzustellen, ob ein Zusammenhang zwischen dem Quellcode einer OSS und dem zur OSS gehörenden Geschäftsmodell existiert. Diesen Zusammenhang wird man nicht direkt messen können. Gestützt wird diese Annahme durch den Artikel von Mac Cormack [MRB06]. Aus dem Artikel geht hervor, dass man Modularität nicht direkt messen kann. Es lässt sich nur sagen, ob Produkt A weniger oder mehr modular aufgebaut ist als Produkt B [MRB06].

Die OSS Analyse nutzt den Vorteil, dass eine Software ein informationsbasiertes Produkt ist. Darunter ist zu verstehen, dass eine Reihe von Anweisungen (Source Code) dem Computer vorgeben, welche Aufgaben er verrichten soll. Wenn man dies Berücksichtigt, können die Abhängigkeiten der einzelnen Softwareelemente automatisch ermittelt wer-

den [MRB08]. Die von MacCormack angesprochenen Anweisungen (Source Code), sind die „functions it performs“ bei Ulrich oder die Aufgaben und Parameter bei Baldwin.

Gemessen wird die Modularität mit der Metrik Propagation Cost. Mit Hilfe einer DSM lässt sich diese ermitteln. Durch diesen Vergleich verschiedener OSS und deren zugeordneten Geschäftsmodellen lässt sich so eventuell zeigen, ob es einen Zusammenhang zum Geschäftsmodell gibt oder nicht.

3.2. Thesen und Fragestellungen

Das Thema der Diplomarbeit ist die Erforschung der Dualität zwischen Open Source Software und Geschäftsmodell. Darunter ist zu verstehen, ob ein Zusammenhang zwischen OSS und Geschäftsmodellen existiert. Um dieser Fragestellung nachgehen zu können, wurden bereits Grundlagen und Voraussetzungen beschrieben. Eine Erforschung dieser Fragestellung erfordert aber auch zusätzlich Thesen und Fragestellungen, mit der man sich der Antwort nähern kann. Nachstehend werden Fragen und Thesen vorgestellt, die im Rahmen der Diplomarbeit bearbeitet werden:

- Für die Analyse der OSS muss ein geeignetes Verfahren gefunden werden. Die Qualität der erstellten DSM muss anhand gegebener DSM aus der Literatur überprüft werden. Zum Finden des Verfahrens ist es notwendig verschiedene Analyseprogramme zu testen.

- Welche OSS soll analysiert werden?
  Damit man einen Zusammenhang zwischen der OSS und dem Geschäftsmodell finden kann, müssen auf jeden Fall OSS analysiert werden, bei denen unterschiedliche Geschäftsmodelle zum Einsatz kommen. Erst mit einem Vergleich der Analyseergebnisse verschiedener OSS lässt sich eventuell ein Zusammenhang erkennen und zeigen.

- Mit den Analyseergebnissen können dann folgende Thesen und Fragestellungen beantwortet werden:

– Spiegelt sich die Erkenntnis von MacCormack wieder, dass Programme, die OSS sind oder zu OSS wurden, strukturierter sind als nicht offene Entwicklungen. Dies müsste sich dadurch zeigen, dass spätere Versionen strukturierter sind als die Vorgängerversionen. MacCormack [MRB06] zeigt dies, indem er die erste Version einer OSS mit einer späteren Version vergleicht.

– Bleiben Programme konstant über die Versionen hinweg?
Weisen Programme zu starke Veränderungen zwischen den einzelnen Versionen auf, so wäre dies ein Indiz dafür, dass das Finden eines Zusammenhangs zwischen OSS und Geschäftsmodell nicht möglich ist. Da sich das Geschäftsmodell einer Software nicht mit jeder Version ändert, muss eine gewisse Stabilität der Software vorhanden sein.

– Weisen Programme, die auf den gleichen Modulen aufbauen oder gleiche Programmfragmente verwenden, ähnliche oder gleiche Strukturen auf?
Bei Webbrowsern ist es zum Beispiel so, dass oftmals eine Layout-Engine in verschiedenen Browsern verwendet wird[1]. Die gleichen Module, oder besser gesagt der gleicher Source Code, sollte zu mindestens an dieser Stelle für die gleiche Struktur sorgen. Wenn das der Fall ist und zwei verschiedene OSS mit jeweils unterschiedlichem Geschäftsmodell ein und dasselbe Modul verwenden oder darauf aufbauen und sich die Struktur gleicht, so wäre dies ein Zeichen dafür, dass es keinen Zusammenhang zwischen OSS und Geschäftsmodellen gibt.

– Besitzen Programme vom gleichem Typ ähnliche Strukturen?
Auf den Webseiten von SourceForge werden die Programme kategorisiert. Zum Beispiel bedeutet Kategorie A „Communications“, B „Database“ und Q „Text Editor“ [Cap09]. Es könnte zum Beispiel sein, dass Software einer gleichen Kategorie ähnlich strukturiert ist, weil dies einfach die Aufgabenstellungen des Programmes bzw. der Aufgabenstellung der Kategorie erfordert. Wenn OSS aus der gleichen Kategorie mit unterschiedlichen Geschäftsmodellen gleich strukturiert ist, so wäre dies ein Hinweis dafür, dass es keinen

[1] Mozilla Firefox, SeaMonkey und Nautilus, verwenden alle die Layout-Engine Gecko [Webc] [Webf]

Zusammenhang zwischen Geschäftsmodell und OSS gibt oder eben wenn es Unterschiede gibt, dass ein Zusammenhang existiert.

– Beinhalten OSS, die in unterschiedlichen Varianten angeboten werden, zum Beispiel das in Kapitel 2 beschriebene MySQL, strukturelle Unterschiede und oder Ähnlichkeiten im Vergleich zu den einzelnen Versionen untereinander? Wenn es ein Unterschied im Quellcode zwischen der freien und der kommerziellen Version existiert, ermöglicht wird dies durch Dual Licensing, dann wäre dies ein Zeichen für den Zusammenhang zwischen OSS und Geschäftsmodell.

Das Kapitel 4 beschäftigt sich mit dem Finden eines geeigneten OSS Analyseverfahrens. Das in den Thesen geforderter Prüfen auf die Qualität des Verfahrens wird dort ebenfalls angesprochen. Im Anschluss geht das Kapitel 5 auf die einzelnen OSS ein. Dazu zählt auch die Zuordnung des Geschäftsmodells zur OSS. In der Einleitung dieses Abschnittes wird auf die geforderte Kategorisierung eingegangen. Die übrigen Fragen und auch deren Ergebnisse, behandelt im Anschluss der beiden eben genannten Kapitel 4 und 5 das Kapitel 6.

4. Analyse

Dieses Kapitel beschreibt Programme, die zur Analyse von OSS verwendet werden können. Ein Bestandteil der Diplomarbeit ist das Finden von geeigneter Analysesoftware. Die einzelnen Programme werden kurz vorgestellt und anschließend bewertet, ob sie geeignet sind oder nicht. Da für die Analyse mehrere Programme zum Einsatz kommen, wird im Anschluss an die Programmvorstellung und Programmauswahl der Analysesoftware der Datenfluß und die eigentliche Datengewinnung beschrieben.

4.1. Analyseprogramme

Für die Programmanalyse gibt es eine Vielzahl von Möglichkeiten. Einen Einstieg in die Vielzahl der Programme rund um die Programmanalyse, insbesondere im Hinblick auf die DSM, bieten [Webb] [Intb] [Inta]. Die Varianten reichen von Bibliotheken für das Erstellen von eigenen Analysenprogrammen, bis hin zu fertigen sehr komplexen Analyseumgebungen.

4.1.1. jDSM

Ein passendes Beispiel für eine Bibliothek ist jDSM.

> „jDSM is a Java library for representing and analyzing Design Structure Matrices (DSM). It can be used to analyze any Java software with regards

to modularity.“ [Webe]

Das Interessante an jDSM ist, dass diese Bibliothek Algorithmen zur Berechnung der Metriken Propagation Cost und Clustered Cost beinhalten, sowie das Speichern einer DSM als Scalable Vector Graphic (SVG) ermöglicht. Das Finden von Abhängigkeiten übernimmt jDSM nicht. Dafür verwendet es das Programm Dependency Finder[1] [Webe]. Diese Variante der Programmanalyse wurde nicht weiter verfolgt. Die Bibliothek jDSM stellt Methoden und Datenstrukturen für DSM zur Verfügung und ist durch den Dependency Finder erst einmal auf Java Programme eingeschränkt. Alles in allem wäre hier ein viel zu hoher Entwicklungsaufwand notwendig, der sich aus zeitlichen Gründen und der Einschränkung auf die Analyse von Java-Programmen nicht rentiert. Ein weiterer Punkt, der gegen jDSM spricht, ist die Frage, wie man C/C++ Programme analysiert, entsprechend für jDMS aufbereitet und anschließend mit jDSM verarbeitet. Auch unklar ist das Ergebnis der DSM als SVG Grafik. Da kein Beispiel angegeben ist und aus den oben genannten Gründen kein einfaches Ausführen zu Testzwecken mit jDSM möglich ist, wird dieses Projekt nicht weiter verfolgt. Im späteren Verlauf dieses Kapitels wird gezeigt, dass die Größe einer DSM Grafik durchaus problematisch ist.

4.1.2. KCachegrind

Für UNIX konforme Betriebssysteme, insbesondere Linux, gibt es ein Open Source Programm mit dem Namen KCachegrind. KCachegrind kann für die Analyse von Call Graphen verwendet werden. Das Finden von Anhängigkeiten übernimmt KCachegrind nicht, sondern verwendet hier ein weiteres Tool mit dem Namen Valgrind [Webd]. Die Programme ließen sich zwar starten, unterbrachen aber auf Grund verschiedener Fehlermeldungen. Diese ließen sich nicht ausmerzen. Daher wurde dieser Ansatz nicht weiter verfolgt.

Ähnlich wie bei jDSM gibt es einige Fragestellungen die sich nicht einfach beantworten lassen. Zum Beispiel ist unklar, ob die Call Graphen aus KCachegrind sich in irgendeiner Art und Weise in eine DSM überführen lassen. Wie oben beschrieben, liegt der

[1] Dependency Finder is a suite of tools for analyzing compiled Java code [Weba].

Schwerpunkt von KCachgrind auf dem Erstellen und Analysieren von Call Graphen. Call Graphen sind ein notwendiger Teil für die Erstellung einer DSM und das Berechnen von Metriken.

Bis jetzt wurden nur Open Source Lösungen betrachtet. Auf der kommerziellen bzw. proprietären Seite gibt es ebenfalls eine Vielzahl von Programmen. Die bereits oben genannten Quellen [Webb] [Intb] [Inta] bieten auch hier einen guten Einstieg. Das nachstehende Kapitel stellt zwei proprietäre Varianten vor.

4.1.3. Understand und Lattix

Von Beginn der Recherche an waren die Programme Understand und Lattix sehr interessant. Beide Firmen bieten eine Lizenz für Forschungszwecke oder nicht kommerzielle Projekte an. Besonders interessant an den beiden Programmen ist, dass Understand eine Vielzahl von Programmiersprachen unterstützt und Lattix mit den Projektdateien von Understand umgehen kann. MacCormack und Rusnak prüften eine Reihe von Programmen auf die Tauglichkeit für Programmanalysen. In einem Nebensatz erwähnten sie:

> „A product called Understand C++ was selected given it best met all these criteria.“ [MRB06]

Es soll nicht unerwähnt bleiben, dass MacCormack und Rusnak ihr eigenes Analyseprogramm und einen eigenen DSM-Viewer entwickelt haben [MRB06] [MRB08]. Die Quelle [MRB08] mit dem gleichen Zitat von MacCormack und Rusnak war erst im letzten Teil der Arbeit relevant. Umso schöner ist es, dass das Zitat erst gegen Ende aufgefallen ist und damit die Entscheidung, diesen Weg einzuschlagen, bestätigt hat. Die nächsten beiden Abschnitte stellen die proprietären Programme Understand und Lattix vor.

Understand

Bei Understand handelt es sich um eine interaktive und wartungsorientierte Entwicklungsumgebung. Ähnlich wie das in Kapitel 2.3 erwähnte GUPRO der Universität Koblenz, dient auch Understand der Unterstützung von Wartung und Programmverständnis. Understand unterstützt eine Reihe von Programmiersprachen. Dadurch ist es möglich mit dem gleichen Programm verschiedene OSS zu analysieren.

> „Understand 2.0 is a cross-platform, multi-language, maintenanceoriented IDE (interactive development environment). It is designed to help maintain and understand large amounts of legacy or newly created source code. The source code analyzed may include Ada, C++, C#, FORTRAN, Java, JOVIAL, Delphi/Pascal, and/or PL/M.“ [Sci09]

Analog zu GUPRO erstellt Understand ein internes Repository der analysierten Software 2.3. Genaue Aussagen sind darüber nicht zu finden.

Understand liefert verschiedene Informationen über die untersuchte Software. Die für die spätere Analyse relevanten Informationen sind in der Ansicht „Project Overview“ zusammengefasst [Intm] [Sci09]. Diese besteht aus folgenden Punkten:

- **code breakdown** gibt das Verhältnis von Quellcode, Kommentaren, Leerzeilen, inaktiven Zeilen und Präprozessorzeilen wieder. Interessant ist dies, um zu vergleichen, ob sich zum Beispiel der Anteil von Kommentaren im Laufe der Entwicklung über die verschiedenen Programmversionen hinweg vergrößert oder eher vermindert hat.

- **function breakdown** gibt die Verteilung bzw. den Anteil von öffentlichen (public), privaten (private) und geschützten (protected) Funktionen wieder.

- **most complex functions** gibt die fünf komplexesten Funktionen wieder. Leider geben die Entwickler von Understand nicht bekannt, wie sie dies ermitteln. Eine Möglichkeit zum Ermitteln der Komplexität von Funktionen wäre zum Beispiel

das Zählen von Programmanweisungen und das Vorkommen von rekursiven Funktionsaufrufen oder das Vorhandensein von Schleifen, insbesondere verschachtelten Schleifen. Je mehr Programmanweisungen oder andere eben genannte Bestandteile vorhanden sind, desto komplexer ist eine Funktion.

- **largest files** gibt die größten fünf Dateien an. Interessant ist dies für die Analyse, da es ein Indikator für eine Umstrukturierung der Software ist, wenn hier zwischen den einzelnen Versionen unterschiedliche Dateien aufgeführt werden.

- **largest functions** gibt die größten Funktionen an.

- **most complex files** gibt die komplexesten fünf Dateien an. Dies könnte man analog zu den most complex functions ermitteln, allerdings wird hier die gesamte Datei berücksichtigt.

Understand bietet keine Darstellungsformen der analysierten Software in DSM an. Dies ist aber nicht weiter von Nachteil, da die abgespeicherten Projektdateien von einem anderen Programm importiert und aufbereitet werden können. Der nächste Abschnitt stellt genau dieses Programm mit dem Namen Lattix vor.

Lattix

Eine Funktion von Lattix ist das Darstellen einer Software in Form von DSM und die damit verbundenen Möglichkeiten Informationen zu entnehmen [Lat04] [Webi].

> The Lattix LDM(TM) solution employs a unique and powerful strategy for communicating, visualizing, analyzing, and managing a software system's architecture [Lat04].

Lattix ermöglicht das Analysieren von Java Programmen, aber nicht von C/C++ Programmen. Es gibt jedoch die Möglichkeit, die Projektdateien aus dem eben vorgestellten Understand zu importieren. Da Undertand mehrere Programmiersprachen unter-

stützt [Sci09], können Call Graphen von Quellcode verschiedener Programmiersprachen als DSM dargestellt werden. So auch von C/C++ Programmen.

Lattix ermöglicht das Abspeichern der DSM als Grafikdatei (*.jpg). Das Problem dabei ist allerdings, dass diese DSM enorm groß wird. Die Abbildung 4.1 zeigt einen Bildausschnitt einer DSM aus dem Programm Lattix. Man erkennt, dass die Informationen in

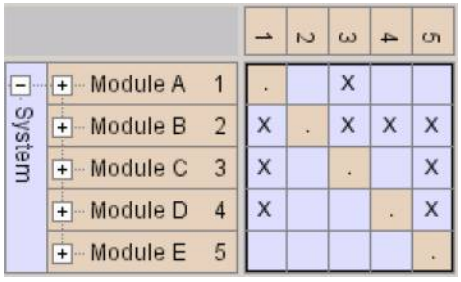


Abbildung 4.1.: Einfache DSM Abbildung aus dem Programm Lattix [Lat04]

der DSM auf Grund der Beschriftung viel Platz beanspruchen. Leider kann man dies mit keiner Einstellung des Programmes abändern. Speichert man ein sehr komplexes System als Grafik ab, lässt sich in der der Abbildung 4.2 nicht wirklich viel erkennen. Diese Abbildung zeigt eine frühe Version von MySQL.

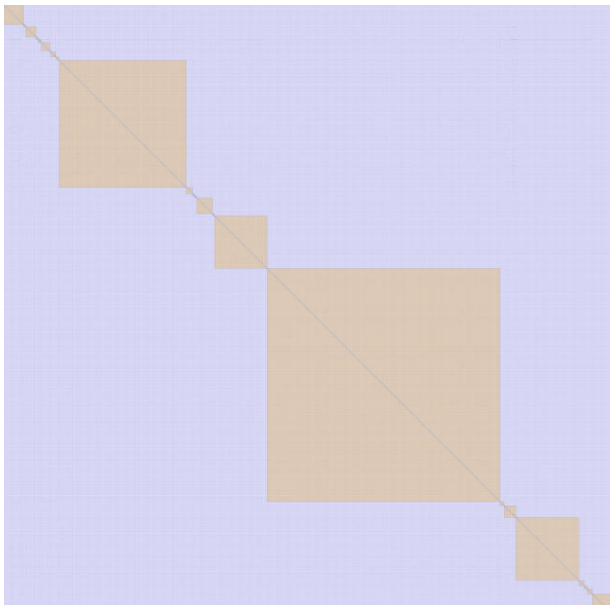


Abbildung 4.2.: Sehr große DSM aus Latix

Auf Grund der Größe der Datei kann es zu Problemen bei der Verarbeitung kommen („Java Heap Space“ Fehler s. B auf Seite 117). Diese Fehlermeldung gibt dem Anwender darüber Auskunft, dass der Speicher für die virtuelle Java Maschine (JVM) nicht ausreicht. Eine DSM von MySQL besteht aus ca. 1500 Elementen. Jedes Element nimmt ca. fünf Millimeter Platz in der Grafik ein. Das ergibt ein riesiges Bild. In einer solchen Grafik lassen sich kaum Strukturmerkmale erkennen. Die Abbildung 4.3 zeigt eine sehr kompakte DSM, in der dies möglich ist. Das eingerahmte C und der dazugehörige Kreis in der Grafik, markieren zum Beispiel einen „vertical bus“ wie er in Kapitel 2.3.3 beschrieben wurde. Das eingerahmte B und der dazugehörige Kreis in der gleichen Grafik 4.3 zeigen Subsysteme, die sehr hohe Abhängigkeiten aufweisen. Mit der Abbildung 4.2 sind solche Erkenntnisse nicht möglich, da sie auf Grund der Größe der DSM nicht zu erkennen sind. Das Abspeichern einer Grafik löst darüber hinaus auch nicht das Problem des

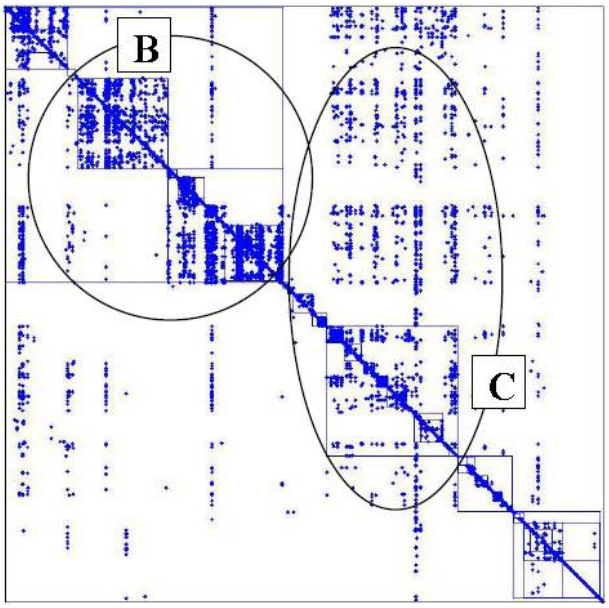


Abbildung 4.3.: DSM Programmanalyse
[MRB06]

Berechnens von Metriken. Abhilfe schafft hier die Exportmöglichkeit als CSV Datei. In Anlehnung an die Abbildung 4.1 würde die CSV Datei wie Listing 4.1 aussehen. Der Datenfluss der einzelnen Programme wird in Kapitel 4.2 beschrieben.

Die eben angesprochenen Probleme und Erklärungen machen deutlich, dass ein eigener DSM Viewer, der die CSV Dateien aus Lattix entsprechend aufbereitet, erforderlich ist.

```
1 ,  , 1, 2, 3, 4, 5
2 Modul A, 1, ., , X, ,
3 Modul B, 2, X, ., X, X, X
4 Modul C, 3, X, , ., , X
5 Modul D, 4, X, , , ., X
6 Modul E, 4, , , , , .
```

Listing 4.1: CSV-Datei der Abbildung 4.3

Alan MacCormack, ein Pionier bei der Betrachtung von Software mittels DSM, bestätigte in einer Email (s. Anhang A auf Seite 115) die Entwicklung eines eigenen Viewers, da die Darstellung von großen Systemen mittels DSM problematisch sei [Webi].

4.1.4. CsvToDSM

Bei dem Programm „CsvToDsm“ handelt es sich um eine Eigenentwicklung des Autors. Der Name des Programmes „CsvToDsm“ leitet sich aus der Funktionsweise des Programmes her. Csv steht für „Character Separated Values (CSV)“ zu Deutsch „Zeichen separierte Werte“. DSM steht für Design Structured Matrix. Das Programm wandelt eine Csv Datei in eine DSM um. Kurz CsvToDSM.

Wird das Programm gestartet, so zeigt sich folgende Bildschirmmaske (Abbildung 4.4). Die einzelnen mit Kreisen markierten und durchnummerierten Stellen in der Abbildung 4.4 haben folgende Bedeutung:

1. Öffnet ein Dateiauswahlfenster. In diesem Fenster werden die CSV Dateien geöffnet.

2. Mit den Button „Zoom out“ kann die DSM verkleinert werden.

3. Mit dem Button „Zoom in“ kann die DSM vergrößert werden.

4. Hier hat der Benutzer die Möglichkeit verschiedene Einstellungen vorzunehmen. So ist es zum Beispiel möglich, die Punkte der DSM auf die gleich Größe und auf die gleiche Farbe einzustellen.

5. Speichert die aufbereitete DSM als JPEG ab. Der Speicherort wird wie bei dem Öffnen der CSV Datei über ein Dateiauswahlfenster angegeben.

6. Hierbei handelt es sich um eine Informationsleiste. Sie liefert dem Benutzer Informationen über die Größe der DSM, die Anzahl der Dateien, die Propagation Cost und vieles mehr.

7. Das ist die als Grafik aufbereitete DSM.

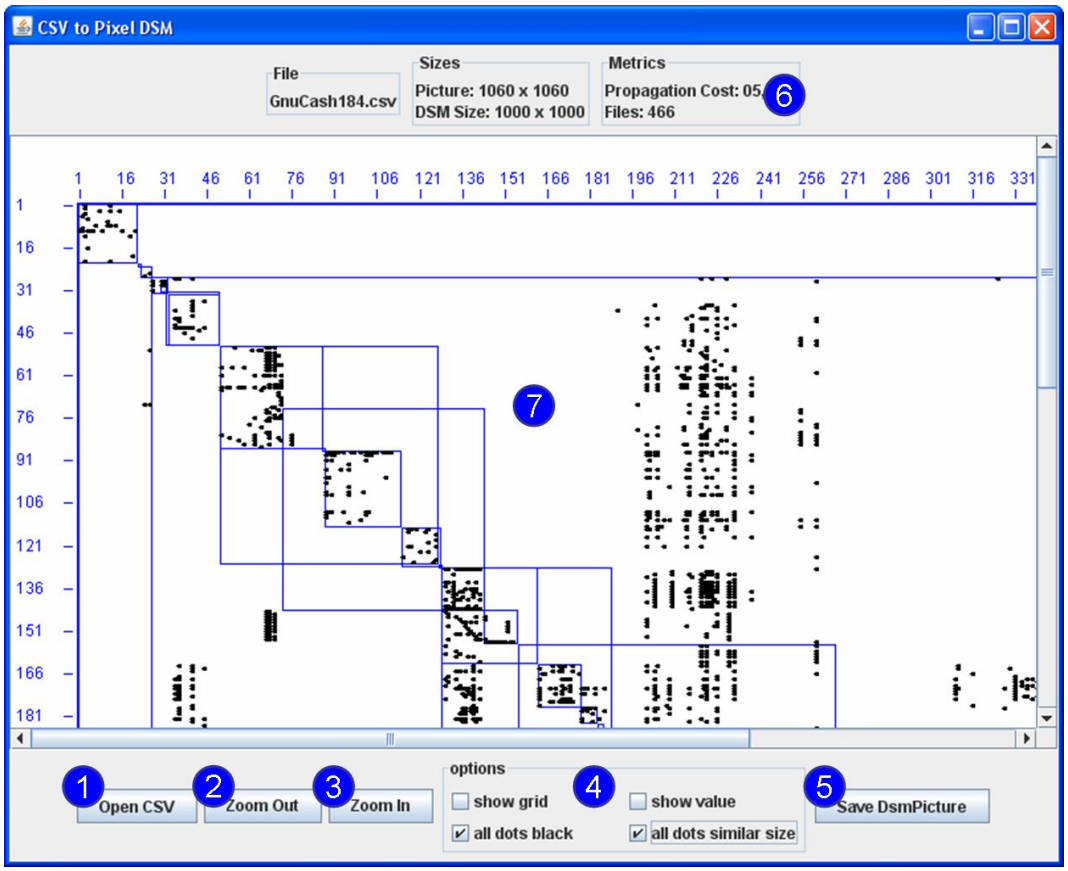


Abbildung 4.4.: Bildschirmmaske CsvToDms

Das Interessante an diesem kleinen Programm ist, dass die CSV Datei intern in einer Matrix abgespeichert ist. Bei dieser Matrix handelt es sich um ein zweidimensionales

```
1
2  private float calcPropagationCost(){
3   float PropagationCost = 0;
4   for (int xi = 1; xi < intMatrix.getCsvData().length; xi++)
5   {
6    for (int yi = 1; yi < intMatrix.getCsvData()[xi].length; yi++)
7    {
8     if(intMatrix.getData(xi,yi)>0) PropagationCost++;
9    }
10  }
11
12  PropagationCost = PropagationCost / ((matrixSize-1)^2 );
13
14  return PropagationCost;
15
16 }
```

Listing 4.2: Berechnung der Metrik PropagationCost (Quellcode Ausschnitt aus CsvToDSM

Integer Array. Integer bedeutet, dass Zahlenwerte abgespeichert werden. Ein zweidimensionales Array entspricht der Vorstellung einer Tabelle aus einem Tabellenkalkulationsprogramm. Diese interne Repräsentation der Daten ermöglicht das Implementieren von denen in Kapitel 2.3.3 vorgestellten Metriken.

Einzelne Programmzeilen aus dem Quellcode des DSM-Viewer CsvToDSM werden mit der Formel aus dem Kapitel 2.3.3 verglichen. Im Detail wird die für die spätere Analyse verwendete Berechnungen der Propagation Cost im Quellcode gezeigt. Die in Formel 2.1 gezeigte Darstellung der Matrix ist das Objekt intMatrix. Dieses Objetkt enthält die Daten der CSV-Datei aus Lattix. Mit dem Befehl intMatrix.getData(xi,yi) kann man auf die einzelnen Werte der DSM zugreifen. intMatrix.getData(xi,yi) entspricht dem D_{ij} aus Abschnitt 2.3.2. Die Formel 2.4 reduziert den Wert aus der DSM, auf 1 bzw. auf 0. Entweder existiert eine Abhängigkeit (1) oder es existiert keine Abhängigkeit (0) zwischen zwei Elementen. Dieser Berechnungsschritt passiert in Zeile 8 (Listing 4.2).

Die beiden For-Schleifen entsprechen den beiden Summenzeichen aus der Formel 2.5. Die Berechnung der Formel Propagation cost 2.5 passiert in Zeile 12 (Listing 4.2). Diese Zeile entspricht nicht eins zu eins der Formel 2.5. Die gesamte Funktion calcPropagationCost() repräsentiert aber die Metrik Propagation Cost.

4.2. Datengewinnung

Die Datengewinnnung besteht aus vier verschiedenen Ebenen. Die Abbildung 4.5 zeigt den Ablauf der Datengewinnung. Die Ebene Source Code umfasst alle Aktivitäten rund um den Quellcode. Bevor der Quellcode eines einzelnen Programmes genutzt werden kann, muss dieser erst einmal entpackt werden. Da die einzelnen Open Source Programme auf verschiedenen Plattformen, zum Beispiel Windows oder Linux, entwickelt werden, wird mehr als ein Komprimierungsverfahren verwendet. In dieser Ebene kommen verschiedene Packprogramme zum Einsatz. Ist der Quellcode entsprechend entpackt, kann mit der nächsten Phase der Datengewinnung begonnen werden.

Der aufbereitete Quellcode wird dann mit dem Programm Undertsand 2.0 aufbereitet. Nach dem Durchlauf des Programmes steht eine sogenannte *.udb Datei. Sie enthält projektrelevante Daten, mit denen später weitergearbeitet wird. Ferner bietet Understand selbst Analysen und Auswertungen an. Besonders interessant sind hier die Project Overview (s.o. 4.1.3). Diese werden in ein Tabellenkalkulationsprogramm übertragen und entsprechend aufbereitet. Die Aufbereitung dient dazu, dass alle einzelnen Overviews das gleich Aussehen haben. Dies ist wichtig, damit später die Auswertung dieser Project Overviews leichter erfolgen kann.

Im nächsten Schritt wird die *.udb Datei in ein Lattix Projekt importiert. Lattix bereitet die Daten entsprechend auf. Aus den oben in Kapitel 4.1.3 genannten Gründen ist ein direkter Export der DSM als Grafik nicht möglich. Lattix ermöglicht aber eine Export der DSM im sogenannten Character Separated Value (CSV) Format. Wie die DSM in diesem Format aussieht, zeigt das Kapitel 4.1.3.

In der letzten Stufe der Datengewinnung, werden die einzelnen CSV Dateien in eine

übersichtliche DSM überführt. Für diesen Zweck hat der Autor das Programm CsvToDSM entwickelt (s.o. 4.1.4). Dieses Programm ermöglicht das Erstellen einer DSM aus einer CSV, berechnet die Metrik Propagation Cost und gibt die Anzahl der Dateien (Elemente) an. Damit die DSM auch bei kleiner Größe gut zu erkennen sind, bietet das kleine Tool verschiedene Einstellung.

Anmerkung zur Dauer der Datengewinnung

Der Prozess Datengewinnung ist sehr zeitaufwändig. Je nach Umfang und Größe des Quellcodes dauert die Analyse eine gute Stunde. Einige OSS sind so groß, dass mit diesem Verfahren keine DSM erstellt werden kann.

Anmerkung zur Genauigkeit und Vergleichbarkeit der Datengewinnung

Dieses Verfahren zur Datengewinnung liefert im Vergleich mit Alan MacCormack die gleichen Daten. Ein Abgleich einer frühen Version des Linux Kernels und des Programmes GnuCash macht dies deutlich. Im Anhang C auf Seite 119 zeigen die Abbildungen C.1 und C.3 die DSM von Alan MacCormack und die Abbildungen C.2 und C.4 das Ergebnis des eben vorgestellten Prozesses der Datengewinnung. Die DSM sind fast gleich. Die DSM des Linux Kernels sind bis auf einen einzigen Punkt identisch. Die einzige Abweichung ist in der Abbildung C.2 mit einem Kreis markiert. Wenn man sich die Formel zur Berechnung der Metrik Propagation Cost 2.3.3 anschaut, so wird deutlich, dass diese kleine Abweichung keine große Auswirkungen auf das Ergebnis hat.

Selbst wenn größere Abweichungen vorhanden sein sollten, so wären die für alle analysierten Programme gleich. In Kapitel 3 wurde gezeigt, dass man die Strukturierung nicht direkt messen kann. Vielmehr geht es hier um einen Vergleich der Strukturierung. Für das Vorhaben der Diplomarbeit ist diese Form der Datengewinnung akzeptabel.

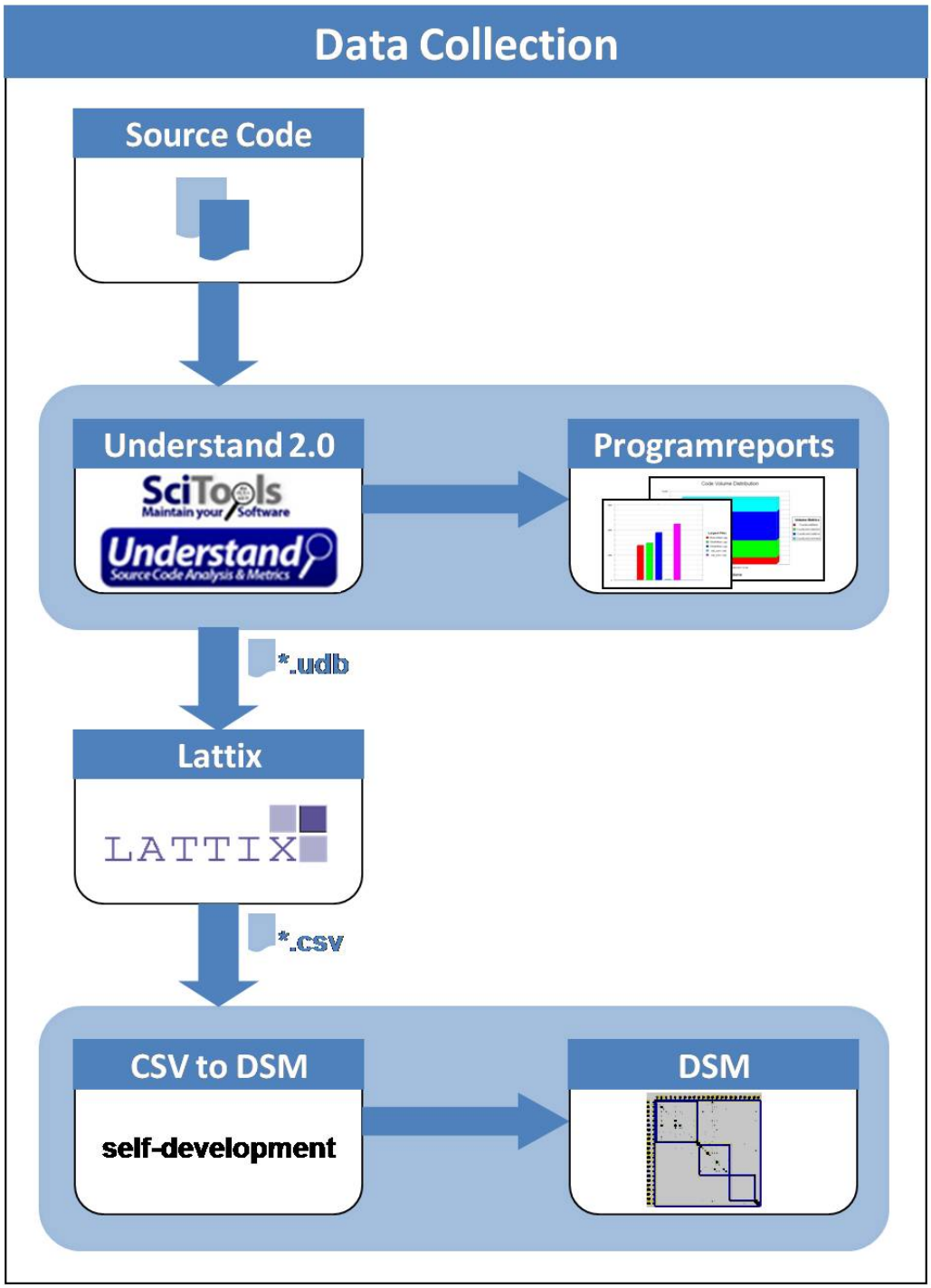


Abbildung 4.5.: Datengewinnung

5. Analysierte Open Soruce Software

Dieses Kapitel stellt die analysierten OSS vor. Nicht alle Source Codes stammen von SourceForge. In Kapitel 3 wurde auf die Kategorisierung der Programme hingewiesen. Diese wird vom Autor vorgenommen, in dem zum Beispiel Produkte rund um das Thema Datenbank der Kategorie Datenbank zugeordnet werden. Die analysierte OSS lässt sich in folgende Kategorien einordnen:

- Applikationsserver
- Browser
- CRM Systeme
- Datenbanken
- Datensicherung
- Financial
- Linux

Die Vorstellung und Beschreibung der OSS hat folgenden Aufbau:

- Webseiteninformation
  Eine Kurze Beschreibung des Programms oder sonstige interessante Informationen.

Diese Informationen stammen aus den angegebenen Webseiten im Punkt Überblick (s.u.).

- Geschäftsmodell
  Jeder Software wird ein Geschäftsmodell zugeordnet. Bei der Zuordnung wird das in Kapitel 2.2.5 vorgestellte Geschäftsmodellzuordnungsverfahren verwendet.

- Versionen
  Dieser Bereich zählt die analysierten Versionen auf.

- Überblick
  Bei dem Überblick handelt es sich um eine Tabelle, die alle wichtigen Informationen der OSS zusammenfasst.

5.1. Applikationsserver

5.1.1. JBoss Enterprise Application Platform

Webseiteninformation

JBoss Enterprise Application Platform is the market leading platform for innovative and scalable Java applications. Integrated, simplified, and delivered by the leader in enterprise open source software, it includes leading open source technologies for building, deploying, and hosting enterprise Java applications and services.

JBoss Enterprise Application Platform balances innovation with enterprise class stability by integrating the most popular clustered Java EE application server with next generation application frameworks. Built on open standards, JBoss Enterprise Application Platform integrates JBoss Application Server, with JBoss Hibernate, JBoss Seam, and other leading open source Java technologies from JBoss.org into a complete, simple

enterprise solution for Java applications.

Geschäftsmodell

Die Tabelle 5.1 zeigt die Geschäftsmodellezuordnung. [1]

Geschäftsmodell	Begründung
Subscription	Das Kernangebot von JBoss ist die sog. „JBoss Subscription“. Diese besteht aus einer Zusammenstellung von Leistungen und Programmen, mit denen die Kosten in allen Stadien des Lebenszyklus von Enterprise Applikationen reduziert werden können. Zu der JBoss Subscription zählen auch Leistungen wie Konzeption, Entwicklung und Bereitstellung, bis hin zur laufenden Administration und Überwachung.
Support Seller	Es werden auch außerhalb der JBoss Subscription Beratung, Schulung, und Support angeboten.

Tabelle 5.1.: Geschätsmodelle JBoss

Versionen

Folgende Versionen wurden analysiert:

1. 2.0-beta1
2. 2.6.2.GA
3. 2.7.0.CR1

[1] `http://www.jboss.de/services/`, `http://www.jboss.com/services/subscriptions/`

Überblick

Die Tabelle 5.2 fast alle wichtigen Informationen zusammen.

Kategorie	Application-Server
Sprache	Java
Lizenz	LGPL
Geschäftsmodell	Subscription, Support Seller
Webseite	`http://www.jboss.org/` `http://www.jboss.com/products/platforms/application/`
Quellcode	`http://www.jboss.org/jbossportal/download/index.html`

Tabelle 5.2.: Überblick jBoss EAP

5.2. Browser

5.2.1. Epiphany

Webseiteninformation

Epiphany is the web browser for the GNOME desktop. Its goal is to be simple and easy to use. Epiphany ties together many GNOME components in order to let you focus on the Web content, instead of the browser application. As part of the GNOME project, Epiphany is Free Software.

Epiphany displays webpages with the same speed and accuracy as other popular browsers, such as Safari or Firefox. In addition, it provides an elegant, responsive and uncomplicated user interface that fits in perfectly with GNOME, and it has been translated to over sixty languages!

Geschäftsmodell

Diesem Produkt kann kein Geschäftsmodell zugeordnet werden. Es gibt einige Möglichkeiten das Projekt Epiphany zu unterstützen. Man kann dem Projekt helfen, in dem man als Testperson, Entwickler oder Problemlöser (in Form einer Mailingliste) hilft. Eine ganz andere Form der Unterstützung ist es, wenn man der Verbreitung von Epiphany hilft. Dies geschieht, in dem man zum Beispiel einen Link zum Projekt und dem Browser auf seiner Internetseite veröffentlicht. Zu allen Bereichen in denen man helfen kann, gibt es ausführliche Anleitungen und Hinweise. Ein Geschäftsmodell ist aber in dem Sinne der Definition in dieser Diplomarbeit nicht zu erkennen.

Contribute

There are several ways to contribute to the Epiphany project.

Testing, go here if you find a problem with the browser, or if you have a feature request.

Developing, get the code, read the documentation, and try to fix errors and send patches to the maintainers. Planning for the next Epiphany releases takes place in the Epiphany Wiki.

Translating, localize the user interface and documentation for your language.

Helping out on the mailing list and IRC for people having problems, letting them know where to go and what to do.

Promote Epiphany! Link to us, and help to spread Epiphany around the Web! [2]

[2] `http://projects.gnome.org/epiphany/`

Versionen

Folgende Versionen wurden analysiert:

1. 2.22.3

2. 2.26.1

Überblick

Die Tabelle 5.3 fast alle wichtigen Informationen zusammen.

Kategorie	Browser
Sprache	C
Lizenz	GPL
Geschäftsmodell	unbekannt
Webseite	`http://projects.gnome.org/epiphany/`
Quellcode	`http://live.gnome.org/Epiphany/Downloads`

Tabelle 5.3.: Überblick Epiphany

5.2.2. Mozilla Firefox

Webseiteninformation

Der preisgekrönte Webbrowser Firefox bietet Sicherheit, Geschwindigkeit und neue Funktionen, die die Art und Weise, wie sie das Web nutzen, verändern werden.

Geschäftsmodell

Dem Browser Firefox kann kein Geschäftsmodell zugeordnet werden. Firefox ist aber ein eingetragenes Warenzeichen der Mozilla Foundation[3]. Die Mozilla Foundation ist als eine Non-Profit-Organisation registriert[4]. Anders als bei dem zuvor vorgestellt Browser Epiphany, ist es hier möglich mit Geldspenden das Projekt zu unterstützen. Der Webserver für die deutschsprachige Webseite `http://www.firefox-browser.de/` wird von der Thomas Krenn AG [5] gesponsert. Ein Geschäftsmodell in Anlehnung an die Definition in dieser Diplomarbeit ist nicht zu finden. Ob die mehrfach Lizenzierung wie im Falle von MySQL dazu dient das Produkt sowohl als freie und als kostenpflichtige Version zu verkaufen, lässt sich leider nicht klären.

Versionen

Folgende Versionen wurden analysiert:

1. 1.0
2. 2.0
3. 3.5

Überblick

Die Tabelle 5.4 fast alle wichtigen Informationen zusammen.

[3] http://www.mozilla.org/foundation/
[4] http://www.mozilla.org/about/
[5] `http://www.thomas-krenn.com/de/index.html`

Kategorie	Browser
Sprache	C++
Lizenz	Mehrfachlizenzierung: MPL/GPL/LGPL
Geschäftsmodell	unbekannt
Webseite	`http://www.mozilla-europe.org/de/firefox/`
Quellcode	`ftp://ftp.mozilla.org/pub/mozilla.org/firefox/releases`

Tabelle 5.4.: Überblick Mozilla Firefox

5.3. Customer Relationship Management Systeme

- Hipergate
- Opentaps

analysiert.

5.3.1. Hipergate

Webseiteninformation

Hipergate is an open source web based application suite.

It's mission is to cover a full range of technical requirements in any organization. All applications are addresses from Internet Explorer, without needing any other additional software in the client computer.

This suite is multi-company capable, and can be used in a single company, a corporate group or working as an ASP solution capable of serving an unlimited quantity of single customers.

Geschäftsmodell

Für das CRM System Hipergate werden verschiedene Services angeboten. Diese reichen von einfachen Installationsempfehlungen bis hin zu verschiedenen Support Paketen [6]. Aus diesem Grund passt das Geschäftsmodell **Support Seller** zu Hipergate. Ein weiteres Geschäftsmodell, welches man zuordnen könnte, wäre **Loss Leader**. Hipergate ist erst einmal kostenfrei und macht damit Unternehmen und Personen auf sich aufmerksam, um im zweiten Schritt kostenpflichtige Produkte, zum Beispiel Wartungsverträge, zu verkaufen.

Versionen

Folgende Versionen wurden analysiert:

1. 1.0.4-beta
2. 3.0
3. 4.0

Überblick

Die Tabelle 5.5 fast alle wichtigen Informationen zusammen.

5.3.2. Opentaps

[6] `http://www.hipergate.org/services/`

Kategorie	CRM Systeme
Sprache	Java
Lizenz	The Mozilla, Apache and the GNU General Public License, very close to GPLv3)
Geschäftsmodell	Loss Leader, Support Seller
Webseite	`http://www.hipergate.org/`
Quellcode	`http://www.hipergate.org/dist/hipergate/`

Tabelle 5.5.: Überblick Hipergate

Webseiteninformation

opentaps is a full-featured ERP + CRM suite which incorporates several open source projects, including Apache Geronimo, Tomcat, and OFBiz for the data model and transaction framework; Pentaho and JasperReports for business intelligence; Funambol for mobile device and Outlook integration; and the opentaps applications which provide user-driven applications for CRM, accounting and finance, warehouse and manufacturing, and purchasing and supply chain mmanagement.

The goal of opentaps is to create high quality and innovative solutions for business problems using open source software, from classic ERP and CRM to whole new ways to increase sales and profitability.

Geschäftsmodell

Opentaps bietet Dokumentationen und Hilfestellungen nur in Kombination mit Werbung an. Für eine werbefreie Dokumentation muss man ein Abonement abschließen. Für einen Betrag von 100$ pro Jahr und Person, erhält man dann werbefreien Zugriff auf die Dokumentation. Diese Form von Geschäftsmodell gehört zum **Subscription** Geschäftsmodell. Das Bereitstellen von Dokumentationen und Hilfestellungen ist auch eine Form von Support. Daher passt auch das Geschäftsmodell **Support Seller** zum Projekt Hipergate.

Versionen

Folgende Versionen wurden analysiert:

1. 0.8.5
2. 1.0.4

Überblick

Die Tabelle 5.6 fast alle wichtigen Informationen zusammen.

Kategorie	CRM Systeme
Sprache	Java
Lizenz	hipergate Public License (based on the Mozilla Apache and the GNU General Public License
Geschäftsmodell	Subscription, Support Seller
Webseite	`http://www.opentaps.org/`
Quellcode	`http://sourceforge.net/projects/opentaps/files/`

Tabelle 5.6.: Überblick Opentaps

5.4. Datenbanken

5.4.1. MySQL

Webseiteninformation

MySQL 5.0 stellt einen großen Schritt für das populärste Open Source Datenbankverwaltungssystem der Welt dar. Während MySQL jahrelang die bevorzugte Datenbank für die Verwaltung von Webseiten mit hohem Datenverkehr sowie integrierten Datenbankanwendungen war, bietet die neue Version 5.0 außergewöhnliche neue Funktionalitäten, die den Weg für den breiter angelegten Einsatz in Unternehmen ebnen.

Geschäftsmodell

Bereits in Kapitel 2.2.5 Geschäftsmodellgrundlagen auf Seite 36 wurde im Detail anhand des Beispiels MySQL das Geschäftsmodellzuordnungsverfahren erklärt. Ergebnis der Geschäftsmodellanalyse im Falle von MySQL waren:

- Loss Leader
- Support Seller
- Dual Licensing

Versionen

Folgende Versionen wurden analysiert:

1. 4.1.22
2. 5.0.83
3. 5.4.1-beta

Überblick

Die Tabelle 5.7 fast alle wichtigen Informationen zusammen.

Kategorie	Datenbanken
Sprache	C/C++
Lizenz	Duales Lizenzsystem (Proprietär und GPL)
Geschäftsmodell	Loss Leader, Support Seller, Dual Licensing
Webseite	`www.mysql.com/` `ftp://ftp.mysql.com/pub/mysql/src/`
Quellcode	`http://sourceforge.net/projects/firebird/files/`

Tabelle 5.7.: Überblick MySQL

5.4.2. Firebird

Webseiteninformation

Bei Firebird handelt es sich um eine Datenbank (Datenbanksystem). Es ist ein freier Ableger des kommerziellen Datenbankmanagementsystem (DBMS) Interbase der Firma CodeGear.

Geschäftsmodell

Für die Datenbank werden keine Dienste oder andere Serviceleistungen, zum Beispiel Service Level Agreements, angeboten. Eine Unterstützung des Projektes ist über die „FirebirdSQL Foundation“ möglich. Über diesen Weg kann man auf verschiedener Art und Weise das Projekt unterstützen. Angefangen bei Geldspenden, über die Tätigkeit als freier Entwickler, bis hin zum Stellen benötigter Ressourcen, zum Beispiel einen

Webserver, kann man bei der FirebirdSQL Foundation tätig werden. Eine Geschäftsmodellidentifikation ist nicht möglich.

Versionen

Folgende Versionen wurden analysiert:

1. 1.0.3.972 (Interbase)
2. 1.5.5.4926
3. 2.5.0.23247-Beta1

Überblick

Die Tabelle 5.8 fast alle wichtigen Informationen zusammen.

Kategorie	Datenbanken
Sprache	C/C++
Lizenz	IDPL (Freie Software)
Geschäftsmodell	unbekannt
Webseite	`http://firebird-datenbank.de/` `http://www.firebirdsql.org/`
Quellcode	`http://sourceforge.net/projects/firebird/files/`

Tabelle 5.8.: Überblick Firebird

5.4.3. PostgreSQL

Webseiteninformation

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness.

Geschäftsmodell

Neben dem freien und kostenlosen Support über die Webseite und diversen Foren, bietet PostgreSQL auch kommerzielle Unterstützung an. Diese Unterstützung wird aber nicht von PostgreSQL selbst angeboten, sondern von Fremdfirmen. Es ist anzunehmen dass die Firmen die einen kommerziellen Support anbieten, in irgend einer Art und Weise PostgreSQL unterstützen. Wenn man dies berücksichtigt, so tritt PostgreSQL als Entwickler für eine OSS auf, von der andere Unternehmen profitieren, indem sie nicht nur das Produkt einsetzen, sondern auch Dienstleistungen rund um das Produkt verkaufen. Unter diesen Annahmen wird PostgreSQL das Geschäftsmodell **Applikationsanbieter** zugeordnet. Unternehmen, die Bertung, Schulung und Support anbieten, werden dem Geschäftsmodell Support Seller zugeordnet. Da PostgreSQL auf Fremdfirmen verweißt, die diese Leistugnen erbringen, wird hier das Geschäftmodell Support Seller mit dem Zusatz „indirekt“ vermerkt.

Versionen

Folgende Versionen wurden analysiert:

1. v60
2. 8.0.19
3. 8.5alpha1

Überblick

Die Tabelle 5.9 fast alle wichtigen Informationen zusammen.

Kategorie	Datenbanken
Sprache	C
Lizenz	BSD-Lizenz
Geschäftsmodell	Applikationsanbieter, indirekter Support Seller
Webseite	`http://www.postgresql.org`
Quellcode (alte Versionen)	`http://www.postgresql.org/ftp/source/` `ftp://ftp-archives.postgresql.org`

Tabelle 5.9.: Überblick PostgreSQL

5.5. Datensicherung

5.5.1. Amanda

Webseiteninformation

AMANDA, the Advanced Maryland Automatic Network Disk Archiver, is a backup system that allows the administrator to set up a single master backup server to back up multiple hosts over network to tape drives/changers or disks or optical media. Amanda uses native dump and/or GNU tar facilities and can back up a large number of workstations running multiple versions of Unix.

Geschäftsmodell

Ähnlich wie bei der OSS PostgreSQL aus der Kategorie Datenbanken, bietet Amanda freien und kostenlosen Support an.

> Today, AMANDA is completely maintained by a volunteer group, including a user community that provides most of the support.[7]

Darüber hinaus gibt es aber auch Firmen, die kommerziellen Support anbieten. Für Amanda werden analog zu den eben vorgestellten PostgreSQL die Geschäftsmodelle **Applikationsanbieter** und **indirekt Support Seller** zugeordnet.

Versionen

Folgende Versionen wurden analysiert:

1. 2.4.3b2
2. 2.5.1p3

Überblick

Die Tabelle 5.10 fast alle wichtigen Informationen zusammen.

5.6. Financial

5.6.1. Gnucash

[7] http://www.amanda.org/about.php

Kategorie	Datensicherung
Sprache	C, Perl
Lizenz	BSD License
Geschäftsmodell	Applikationsanbieter, indirekter Support Seller
Webseite	`http://www.amanda.org`
Quellcode	`http://www.amanda.org/download.php`

Tabelle 5.10.: Überblick Amanda

Webseiteninformation

GnuCash is personal and small-business financial-accounting software, freely licensed under the GNU GPL and available for GNU/Linux, BSD, Solaris, Mac OS X and Microsoft Windows.

Designed to be easy to use, yet powerful and flexible, GnuCash allows you to track bank accounts, stocks, income and expenses. As quick and intuitive to use as a checkbook register, it is based on professional accounting principles to ensure balanced books and accurate reports.

Geschäftsmodell

Die Zuordnung eines Geschäftsmodells für die OSS Gnucash ist nicht möglich. Man kann dieses Projekt unterstützen, in dem man das Programm testet (Testperson) und die Fehler entsprechend meldet oder als Übersetzer arbeitet. Eine andere Variante ist das Spenden von Geld. Ein Geschäftsmodell, wie es mit dem Geschäftsmodellezuordnungsverfahren gefunden werden kann, gibt es nicht.

Versionen

Folgende Versionen wurden analysiert:

1. 1.8.4

Überblick

Die Tabelle 5.11 fast alle wichtigen Informationen zusammen.

Kategorie	Financial
Sprache	C, Scheme, Perl
Lizenz	GNU General Public License (GPL)
Geschäftsmodell	unbekannt
Webseite	`http://www.gnucash.org/`
Quellcode	`http://sourceforge.net/projects/gnucash/files/`

Tabelle 5.11.: Überblick Gnucash

5.7. Linux

5.7.1. Linux Kernel

Webseiteninformation

Linux ist ein frei verfügbares Multitasking und Multiuser Betriebssystem, das von Linus Torvalds und vielen freien Entwicklern weltweit entwickelt wird. Linux bietet mittlerweile all die Funktionalitäten, die man von modernen Betriebssystemen erwartet. Echtes (präemtives) Multitasking, virtuelle Speicherverwaltung, dynamisch nachladbare Bibliotheken mit Versionsnummern und andere moderne Konzepte machen das POSIX orientierte Betriebssystem zur optimalen Lösung für viele Einsatzgebiete.

Geschäftsmodell

Die hier analysierte OSS Linux versteht im Rahmen dieser Diplomarbeit nur den Linux Kernel. Diesem Kernel (Linux) selbst kann man kein Geschäftsmodell zuordnen. Viele Anbieter nutzen Linux für die Entwicklung ihrer eigenen Systeme. Dies beginnt bei Produkten, bei denen man beim ersten Anblick nicht gleich erkennt, dass im Hintergrund Linux arbeitet. Ein Beispiel hierfür wären DVD-Player. Das in den Grundlagen vorgestsellte Geschäftsmodell mit dem Namen Appliances spiegelt diesen Sachverhalt wieder. Dann gibt es Firmen, die nicht nur OSS sondern auch kommerzielle Software für Linux entwickeln. Und nicht zuletzt gibt es Unternehmen, die rund um Linux tätig sind und zum Beispiel Support, Hilfestellungen und vieles mehr anbieten. Da diese Unternehmen nicht im Zentrum der Betrachtung stehen, wird dem Linux Kernel kein Geschäftsmodell zugeordnet.

Versionen

Folgende Versionen wurden analysiert:

1. 0.01
2. 2.1.105
3. 2.6.30.2

Überblick

Die Tabelle 5.12 fast alle wichtigen Informationen zusammen.

Kategorie	Linux
Sprache	C/C++
Lizenz	GPL
Geschäftsmodell	keins
Webseite	`http://www.linux.de/`
Quellcode	`http://brahe-if-a.mirrorservice.org/sites/ftp.kernel.org/pub/linux/kernel/v2.1/`

Tabelle 5.12.: Überblick Linux Kernel

6. Ergebnisse

Dieses Kapitel geht auf die Ergebnisse und damit auch auf die in Kapitel 3 genannten Forschungsfragen und Thesen ein. Das letzte Kapitel 7 wertet die einzelnen Ergebnisse aus und versucht eine Antwort auf die Fragestellung der Diplomarbeit zu finden.

6.1. Analyseergebnisse

Dieses Kapitel führt alle Analyseergebnisse auf.

- Propagation Cost
- Program Reports
  - Program Overviews
  - Code Breakdown
  - Function Breakdown
- Geschäftsmodellzuordnung (inklusive Lizenzen)

Eine Auswertung findet in Abschnitt 6.2 statt.

6.1.1. Propagation Cost

Die Tabelle 6.1 zeigt die Ergebnisse der Propagation Cost, sowie die Anzahl der Dateien. Bei einigen OSS konnte keine Propagation Cost berechnet werden, da keine DSM erstellt werden konnte. Abschnitt 6.1.4 geht auf diesen Sachverhalt ein.

6.1.2. Program Reports

Die folgenden Daten, mit einem Tabellenkalkulationsprogramm grafisch aufbereitet, stammen aus Understand und geben weitere Anhaltspunkte und Vergleichsmöglichkeiten, aus denen Rückschlüsse gezogen werden können. Zuerst werden die Ergebnisse der Program Overviews (Reports) dargestellt (6.2). Anschließend werden die Ausgewerteten Code Breakdown aufgelistet (6.3). Zuletzt wird die Auswertung der Function Breakdown aufgelistet (6.4). Die Auswertung erfolgte in der Art und Weise, dass die Versionen der einzelnen OSS auf Veränderungen geprüft wurden. Bei den Program Overviews hat die erste Version einen Spiegelstrich, da erst mit der nächsten Version eine Änderung gemessen werden kann. Im Anhang (D.2) ab Seite 150 sind die einzelnen grafisch aufbereiteten Ergebnisse der Program Overviews enthalten. Die Auswertung des Code Breakdown und des Function Breakdown erfolgt analog. Hier wird geprüft ob das Verhältnis zur Vorgängerversion gleich geblieben ist. Auf Veränderungen wird mit den Worten gleich, weniger oder mehr hingewiesen. Fehlende Angaben werden mit einem „x“ gekennzeichnet. Nicht alle Programme haben zum Beispiel Funktionen mit dem Zusatz „Public“. Bei diesen Auswertungen geht es in erster Linie darum, eine Tendenz wiederzuspiegeln. Eine Beschreibung der Bedeutung der einzelnen Spalten (einzelne Program Overviews) kann dem Kapitel (4.1.3) Understand und Lattix entnommen werden.

6.1.3. Geschäftsmodellzuordnung

Die Tabelle 6.5 zeigt die Geschäftsmodellzuordnung und die zugeordnete Lizenz bzw. zugeordneten Lizenzen.

6.1.4. Design Structur Matrix

Die mit dem in Kapitel 4.2 vorgestellten Verfahren gewonnenen DSM werden im Anhang D.1 ab Seite 125 gezeigt. Die einzelnen Grafiken sind in den gleichen Kategorien, in denen auch die OSS zugeordnet sind, dargestellt. Anhand der Bezeichnung der Grafik kann der Name der OSS sowie die Version entnommen werden.

Nicht zu allen Programmen ließ sich eine DSM erstellen. In der Tabelle 6.1 ist dies daran zu erkennen, dass nicht zu allen Programmen eine Propagation Cost berechnet werden konnte. Für die Berechnung dieser Metrik ist die DSM Voraussetzung. Da diese nicht erstellt werden konnte, blieb in diesem Fall die Berechnung aus. Gründe für das Fehlen einer DSM werden in Kapitel 4 erläutert.

Ein Vergleich der erstellten DSM mit denen in der Literatur vorhandenen DSM von Alan MacCormack [MRB06] [MRB08] hat gezeigt, dass die DSM vergleichbar sind. Abschnitt 4.2 gibt im Detail über diesen Vergleich Auskunft.

Im Abschließenden Kapitel 6.2 werden die DSM zur Auswertung herangezogen.

Kategorie	Programm	Version	Anzahl Dateien	Propagation Cost
Applikationsserver	JBoss-Portal	2.0-beta1	517	4,31
Applikationsserver	JBoss-Portal	2.6.2.GA	111	4,62
Applikationsserver	JBoss-Portal	2.7.0.CR1	87	3,46
Browser	Epiphany	2.22.3	164	0,90
Browser	Epiphany	2.26.1	157	0,92
Browser	Firefox	1.0	43	1,41
Browser	Firefox	2.0		
Browser	Firefox	3.5		
CRM Systeme	Hipergate	1.0.4-beta	330	4,05
CRM Systeme	Hipergate	3.0	644	3,3
CRM Systeme	Hipergate	4.0	71	4,9
CRM Systeme	Opentaps	0.8.5	1245	5,12
CRM Systeme	Opentaps	1.0.4		
Datenbanken	MySQL	4.1.22	1138	0,69
Datenbanken	MySQL	5.0.83	958	0,96
Datenbanken	MySQL	5.4.1-beta	726	1,13
Datenbanken	Postgres	v60	339	0,97
Datenbanken	Postgres	8.0.19	703	1,31
Datenbanken	Postgres	8.5alpha1	879	1,23
Datenbanken	Interbase	1.0.3.972	432	2,71
Datenbanken	Firebird	1.5.5.4926	408	2,94
Datenbanken	Firebird	2.5.0.23247 -Beta1	1038	1,63
Datensicherung	Amanda	2.4.3b2	120	1,35
Datensicherung	Amanda	2.5.1p3	152	1,87
financial	Gnucash	1.8.4	466	5,54
Linux	Linux	0.01	45	2,64
Linux	Linux	2.1.105	1678	1,5
Linux	Linux	2.6.30.2		

Tabelle 6.1.: Analyse Ergebnisse: Propagation Cost

Programm	Version	Largest-Files	Largest-Functions	Most Complex Files	Most Complex Functions
JBoss-Portal	2.0-beta1	-	-	-	-
JBoss-Portal	2.6.2.GA	5	4	4	4
JBoss-Portal	2.7.0.CR1	4	5	0	3
Epiphany	2.22.3	-	-	-	-
Epiphany	2.26.1	0	0	0	0
Firefox	1.0	-	-	-	-
Firefox	2.0	4	1	3	2
Firefox	3.5	3	1	1	1
Hipergate	1.0.4-beta	-	-	-	-
Hipergate	3.0	2	3	4	4
Hipergate	4.0	0	0	1	0
Opentaps	0.8.5	-	-	-	-
Opentaps	1.0.4	5	3	1	1
MySQL	4.1.22	-	-	-	-
MySQL	5.0.83	3	4	5	5
MySQL	5.4.1-beta	1	1	0	1
Postgres	v60	-	-	-	-
Postgres	8.0.19	5	4	4	4
Postgres	8.5alpha1	5	5	4	5
Interbase	1.0.3.972	-	-	-	-
Firebird	1.5.5.4926	4	1	1	0
Firebird	2.5.0.23247-Beta1	3	3	4	4
Amanda	2.4.3b2	-	-	-	-
Amanda	2.5.1p3	2	2	5	4
Gnucash	1.8.4	-	-	-	-
Linux	0.01	-	-	-	-
Linux	2.1.105	5	5	5	5
Linux	2.6.30.2	5	5	5	5

Tabelle 6.2.: Analyse Ergebnisse: Reports

Programm	Version	Source-code	Com-ments	Blank Lines	In-active	Pre-Processor
JBoss-Portal	2.0-beta1	-	-	-	x	x
JBoss-Portal	2.6.2.GA	mehr	weniger	gleich	x	x
JBoss-Portal	2.7.0.CR1	gleich	gleich	gleich	x	x
Epiphany	2.22.3	-	-	-	-	-
Epiphany	2.26.1	gleich	gleich	gleich	gleich	gleich
Firefox	1.0	-	-	-	-	-
Firefox	2.0	gleich	gleich	gleich	gleich	gleich
Firefox	3.5	gleich	gleich	gleich	gleich	gleich
Hipergate	1.0.4-beta	-	-	-	x	x
Hipergate	3.0	mehr	weniger	gleich	x	x
Hipergate	4.0	gleich	gleich	gleich	x	x
Opentaps	0.8.5	-	-	-	x	x
Opentaps	1.0.4	gleich	gleich	gleich	x	x
MySQL	4.1.22	-	-	-	-	-
MySQL	5.0.83	gleich	gleich	gleich	gleich	gleich
MySQL	5.4.1-beta	gleich	gleich	gleich	gleich	gleich
Postgres	v60	-	-	-	-	-
Postgres	8.0.19	mehr	weniger	weniger	gleich	gleich
Postgres	8.5alpha1	mehr	weniger	gleich	gleich	gleich
Interbase	1.0.3.972	-	-	-	-	-
Firebird	1.5.5.4926	gleich	gleich	gleich	gleich	gleich
Firebird	2.5.0.23247 -Beta1	mehr	gleich	gleich	viel weniger	weniger
Amanda	2.4.3b2	-	-	-	-	-
Amanda	2.5.1p3	gleich	gleich	gleich	gleich	gleich
Gnucash	1.8.4	-	-	-		
Linux	0.01	-	-	-	-	-
Linux	2.1.105	viel mehr	weniger	weniger	gleich	gleich
Linux	2.6.30.2	weniger	mehr	gleich	viel weniger	mehr

Tabelle 6.3.: Analyse Ergebnisse: Code Breakdown

Programm	Version	Public	Private	Protected	Default
JBoss-Portal	2.0-beta1	-	-	-	-
JBoss-Portal	2.6.2.GA	weniger	mehr	weniger	weniger
JBoss-Portal	2.7.0.CR1	gleich	gleich	gleich	gleich
Epiphany	2.22.3	-	-	-	-
Epiphany	2.26.1	gleich	gleich	gleich	x
Firefox	1.0	-	-	-	-
Firefox	2.0	gleich	gleich	gleich	x
Firefox	3.5	gleich	gleich	gleich	x
Hipergate	1.0.4-beta	-	-	-	-
Hipergate	3.0	gleich	gleich	weniger	mehr
Hipergate	4.0	gleich	gleich	gleich	gleich
Opentaps	0.8.5	-	-	-	-
Opentaps	1.0.4	gleich	gleich	gleich	gleich
MySQL	4.1.22	-	-	-	x
MySQL	5.0.83	gleich	viel mehr	viel weniger	x
MySQL	5.4.1-beta	gleich	gleich	gleich	x
postgres	v60	x	x	x	x
postgres	8.0.19	x	x	x	x
postgres	8.5alpha1	x	x	x	x
interbase	1.0.3.972	-	-	-	x
firebird	1.5.5.4926	viel mehr	mehr	viel weniger	x
firebird	2.5.0.23247-Beta1	mehr	viel mehr	weniger	x
amanda	2.4.3b2	x	x	x	x
amanda	2.5.1p3	x	x	x	x
gnucash	1.8.4	-	-	-	-
linux	0.01	x	x	x	x
linux	2.1.105	x	x	x	x
Linux	2.6.30.2	x	x	x	x

Tabelle 6.4.: Analyse Ergebnisse: Function Breakdown

Kategorie	Programm	Geschäftsmodell	Lizenz
Applikationsserver	JBoss-portal	Subscription Support Seller	LGPL
Browser	Epiphany	unbekannt	GPL
Browser	Firefox	unbekannt	MPL GPL LGPL
CRM Systeme	Hipergate	Loss Leader Support Seller	Mozilla, Apache and the GNU General Public License
CRM Systeme	Opentaps	Subscription Support Seller	Hipergate Public License Apache and the GNU General Public License (GPL)
Datenbanken	MySQL	Loss Leader Support Seller Dual Licensing	Duales Lizenzsystem (Proprietär und GPL)
Datenbanken	Postgres	Applikationsanbieter indirekter Support Seller	BSD-Lizenz
Datenbanken	Firebird	unbekannt	IDPL (Freie Software)
Datensicherung	Amanda	Applikationsanbieter indirekter Support Seller	BSD License
financial	GnuCash	unbekannt	GPL
linux	Linux-Kernel	keins	GPL

Tabelle 6.5.: Analyse Ergebnisse: Geschäftsmodellzuordnung

6.2. Auswertung

Nachstehend werden in den einzelnen Abschnitten verschiedene Aspekte diskutiert. Dabei handelt es sich um eine Auswertung und Interpretation der mit dem in Kapitel 4 vorgestellten Verfahren ermittelten Daten, unter Berücksichtigung der in Abschnitt 3.2 gestellten Fragen und Thesen. Eine abschließende Betrachtung der Fragestellung der Diplomarbeit, unter Berücksichtigung der hier gesammelten Fakten und Annahmen gibt das Fazit.

Alan MacCormack schreibt, dass proprietäre Programme im Vergleich zur OSS weniger strukturiert sind. Die kommerzielle Entwicklung nennt er „closed source“. Wenige Entwickler sitzen an einem Ort und entwickeln die Software. Im Gegensatz dazu versteht er unter „open source“ mehrere hundert Entwickler, die an verschiedenen Orten arbeiten. Da der Quellcode von kommerziellen Produkten im Gegensatz zur OSS in der Regel nicht einsehbar ist, setzt MacCormack die erste veröffentlichte Version einer OSS mit der kommerziellen Entwicklungen „closed source“ gleich. Vergleicht man die erste Version, unter der eben angegebenen Voraussetzungen („closed source“), mit einer späteren Version („open source“), lässt sich zeigen, dass die OSS strukturierter ist. Den Nachweis erbringt MacCormack mit der Metrik Propagation Cost [MRB06] [MRB08]. Diese Erkenntnis spiegelt sich auch in den in der Diplomarbeit erhobenen Daten wider. Vergleicht man eine frühe Version eines Programmes mit einer späteren Version, so ist zuerkennen, dass die Propagation Cost angestiegen ist. Bei den erhobenen Daten gibt es, was diese These anbelangt, zwei Abweichungen. Bei der OSS Linux nimmt die Propagation Cost ab. Gleichzeitig steigt die Anzahl der Dateien um mehr als 1600. Ähnlich verhält es sich in der Kategorie Datenbanken mit dem Produkt Firebird. Bei der letzten Version handelt es sich um eine Betaversion. Es ist anzunehmen, dass in einer Betaversion zusätzliche Funktionen und Testroutinen untergebracht sind. Bei der OSS JBoss-Portal nimmt die Propagation Cost auf den ersten Blick ab. Zieht man aber in Betracht, dass bei der letzten Version nur 87 Dateien verwendet werden, so kann man unter Berücksichtigung der Formel zur Berechnung der Propagation Cost 2.3.3 davon ausgehen, dass der oben beschriebene Effekt auch hier zu erkennen ist. Die Idee von MacCormack spiegelt sich also auch in den hier gewonnenen Daten wieder. Die erstellten DSM machen diesen Effekt deutlich. Man erkennt zum Beispiel im Vergleich von DSM D.7, D.8 und D.9 der OSS

Hipergate, dass der Grad der Strukturierung zugenommen hat. Zu erkennen ist dies daran, dass die einzelnen Subsysteme mehr Abhängigkeiten aufweisen. Deutlich wird dies durch die Zunahme der schwarzen Punkte im Subsystem. Eine grafische Verdichtung der schwarzen Punkte, stellvertretend für die Abhängigkeit der einzelnen Elemente, ist ein Zeichen für einen höheren Grad an Strukturiertheit (s. 2.3.3). Unterstellt man einmal, dass den zwei verschiedenen Softwareentwicklungsansätzen „closed source“ und „open source“ unterschiedliche Geschäftsmodelle zugrunde liegen, wird die Annahme der Diplomarbeit gestärkt, dass ein Zusammenhang zwischen OSS und Geschäftsmodell existiert.

Betrachtet man die der OSS zugeordneten Geschäftsmodelle, unter Berücksichtigung der Lizenzen, stellt man fest, dass immer dann, wenn die General Public License (GPL) verwendet wird, kein Geschäftsmodell zugeordnet werden kann. Mit freier Software im Sinne der GPL ist es anscheinend schwierig kommerziell tätig zu werden. Bei der OSS Hipergate und MySQL gibt es dennoch ein Geschäftsmodell. Dies liegt wahrscheinlich daran, dass die OSS mehrfach lizenziert ist. Im Rahmen der Diplomarbeit wird nicht überprüft, welche Lizenzen mit welchen Geschäftsmodellen oft in Verbindung stehen. Für eine solche Untersuchung wäre es notwendig viele OSS und deren Lizenzen sowie Geschäftsmodelle zu überprüfen. Ließe sich ein Unterschied zwischen OSS verschiedener Lizenzen zeigen, so wäre dies ein weiteres Indiz für den Zusammenhang zwischen Geschäftsmodell und OSS. Zum Beispiel könnte es sein, dass OSS mit der Lizenz GPL strukturierter sind, im Vergleich zu OSS mit anderen Lizenzen, die nicht ganz so strikt sind. Mit Hilfe der DSM oder der Propagation Cost lassen sich hier keine Zusammenhänge erkennen. Auch gegen diese Annahme sprechen die OSS Produkte, die mehrfach lizenziert sind. Dies bedeutet, dass dieselbe Struktur (gleiche OSS), zu verschiedenen Lizenzen passt. Anhand der gesammelten Daten kann gezeigt werden, dass es keinen Zusammenhang zwischen Lizenz und Struktur gibt. Die OSS JBoss-Portal weist im Schnitt eine Propagation Cost von 4,13 auf. Die Lizenz ist LGPL und die Geschäftsmodelle Subscription und Support Seller wurde zugeordnet. Mit fast den gleichen Geschäftsmodellen und einer durchschnittlichen Propagation Cost von 3,6 bei dem CRM System Hipergate und einer durchschnittlichen Propagation Cost von 5,01 bei dem CRM System Opentaps könnte man vermuten, dass ein Zusammenhang besteht. MySQL ebenfalls mit zugeordneter GPL und einer kommerziellen Liezenz (Dual Licensing macht dies möglich), weist eine viel geringere Propagation Cost auf. Auch der Linux-Kernel (Lizenz GPL) hat eine

viel geringere Propagation Cost. Die letzten beiden Gegenbeispiele zeigen, dass es keinen Zusammenhang gibt. Eine Tabelle, in der man anhand der Strukturierung der OSS, dargestellt durch die Propagation Cost, nachsehen könnte, welche Lizenz oder welches Geschäftsmodell verwendet wird, ist daher nicht möglich.

Mit Hilfe der Program Reports Project Overview, Code Breakdown und Function Breakdown lässt sich zeigen, dass Programme über die Versionen hinweg konstant bleiben. Je weiter die Versionen jedoch auseinander liegen, desto stärker sind die Abweichungen. Wenn zwei Versionen weit auseinander liegen, liegt es auf der Hand, dass auch viele Änderungen am Quellcode vorgenommen wurden. Ein Programm erhält immer dann eine höhere Version, wenn es verbessert wurde oder neue Funktionen erhalten hat. Diese Änderungen im Quellcode führen dazu, dass Abweichungen vorhanden sein müssen. Ein Beispiel dafür ist der Linux Kernel. Vergleicht man die sehr frühe Version 0.01 mit einer aktuellen Version 2.6.30.2, so wird dies deutlich. Diesen Effekt erkennt man auch bei einem Vergleich der erstellten DSM. Die DSM (D.17, D.18) der OSS Hipergate sind fast identisch. Diese beiden DSM sind von Versionen, die sehr dicht beieinander liegen (1.0.3.972 und 1.5.5.4926). Vergleicht man diese beiden DSM (D.17, D.18) mit der DSM (D.19) der Version 2.5.0.23247Beta1, so sind deutliche Unterschiede zu erkennen. Man kann dies auch daran verdeutlichen, dass aus einem Finanzprogramm kein Emailprogramm entwickelt wird. Die Erkenntnis der Stetigkeit der Programme ist wichtig, da es sonst schwierig wird, einen Zusammenhang zwischen Geschäftsmodell und OSS zu finden. Bliebe das Geschäftsmodell gleich und würde gleichzeitig die dazugehörige OSS starken strukturellen Veränderungen unterliegen, so wäre das ein Anzeichen dafür, dass es keinen Zusammenhang zwischen OSS und Geschäftsmodell gibt. Die Stetigkeit der Programme wird durch die Propagation Cost gestützt. Hier sind nur geringe Unterschiede bei aufeinander folgenden Versionen zu verzeichnen. In Kapitel 2.3.3 und 3 wurde darauf hingewiesen, dass es sich dabei um ein Maß an Modularität handelt und damit die Struktur der Software widerspiegelt.

Der Fragestellung, ob OSS, die in unterschiedlichen Varianten angeboten werden, zum Beispiel das in Kapitel 2 beschriebene MySQL, strukturelle Unterschiede beinhalten oder Ähnlichkeiten im Vergleich der einzelnen Varianten untereinander aufweisen, kann nicht nachgegangen werden. MySQL gibt es einmal mit der Lizenz GPL und einmal als proprietäre Variante. Für letztere Variante müssen Unternehmen bezahlen. Die kommer-

zielle Variante ist der MySQL Enterprise Server. Die freie Variante (Comunity Server) ist für jeden mit der Lizenz GPL zu haben. Für die kommerzielle Variante werden Dienstleistungen und umfangreichere Funktionen angeboten [Intb]. Leider lässt sich dieser Fragestellung nicht nachgehen, da der Quellcode für den MySQL Enterprise Server nicht zu bekommen war. Anfragen bei DorsalSource [Intd] und MySQL [Intb] blieben erfolglos. MySQL ist der direkte Anbieter. DorsalSource bietet den MySQL Enterpriseserver kostenlos an, um einer Zweiklassengesellschaft vorzubeugen [Inte].

Der Frage, ob Programme, die auf gleichen Modulen aufbauen, gleiche oder ähnliche Strukturen aufweisen, kann nicht nachgegangen werden. Die Struktur und Modularität, welche man anhand des Quellcodes berechnen kann, geht verloren, wenn das Programm bzw. Modul kompiliert wird. In Form eines Binary (kompiliertes Programm), ist die Struktur nicht mehr zu erkennen. Wird dieses Modul, genauer sein Binary, in anderen Programmen verwendet, dann gibt es eine Abhängigkeit zwischen der Quellcodedatei, in der eine Funktion aus dem Binary aufgerufen wird. Die beiden Dateien, Quellcodedatei und Binary, sind dann zwei Elemente in einer DSM. Die Ursprüngliche Struktur der Binray (Modul) ist verloren. Ein Versuch dieser Frage nachzugehen, wurde mit den Programmen Mozilla Firefox, SeaMonkey und Nautilus gestartet. Diese Programme verwenden alle die Layout-Engine Gecko [Webc] [Webf]. Leider ließen sich für SeaMonkey und Nautilus keine DSM erzeugen. Beim Vergleich der Program Reports wird deutlich, dass zum Beispiel die meisten komplexen Funktionen (most Complex Functions) und auch andere Merkmale, bei Firefox und SeaMonkey identisch sind. Dies liegt wahrscheinlich nicht an der gemeinsamen Layout-Engine, sondern eher daran, dass SeaMonkey aus Mozilla Firefox hervorgegangen ist [Intn]. Hätte man einen Zusammenhang zeigen können, bei gleichzeitig unterschiedlichem Geschäftsmodell, so wäre dies ein weiterer fehlender Baustein beim Untermauern der These, dass ein Zusammenhang zwischen OSS und Geschäftsmodell existiert.

Eine im Abschnitt 3.2 gestellte Frage nach der Ähnlichkeit von Programmen gleichen Typs kann nicht mit ja oder nein beantwortet werden. Die analysierten OSS sind in sieben verschiedene Kategorien zugeordnet worden. Wie kann man zeigen ob Programme einer gleichen Kategorie bzw. gleichen Typs ähnlich sind? Mit Hilfe der DSM und der Propagation Cost können zumindest Vergleiche angestellt werden. Die Tabelle 6.6 zeigt die Kategorie mit der durchschnittlichen Propagation Cost.

Kategorie	**Propagation Cost**
Applikationsserver	4,13
Browser	1,08
CRM Systeme	4,34
Datenbanken	1,51
Datensicherung	1,61
Financial	5,54
Linux	2,07

Tabelle 6.6.: Durchschnittliche Propagation Cost pro Kategorie

Vergleicht man zum Beispiel die Kategorien Financial mit Applikationserver oder CRM-Systeme, so sind die Unterschiede sehr gering. Ähnlich verhält es sich auch mit den übrigen Kategorien Browser, Datenbanken und Datensicherung. Sicherlich sind die Daten begrenzt und lassen keinen allgemeinen Schluss zu. Dennoch zeigt diese Stichprobe, dass die Kategorien sehr ähnlich sind. Dieser Frage muss eigentlich auch nicht weiter nachgegangen werden, denn Geschäftsmodelle können in mehreren Kategorien angewandt werden. Zum Beispiel das Geschäftsmodell Support Seller ist in den Kategorien Applikationsserver, CRM-System, Datenbanken und Datensicherung vertreten. Über die Ähnlichkeit der Programme gleichen Typs einen Zusammenhang zu Geschäftsmodellen herzustellen, wäre nur dann sinnvoll, wenn einzelne Geschäftsmodelle auch nur bestimmten Kategorien zugeordnet werden können. Dies ist aber nicht der Fall.

Das Abschließende Kapitel 7 formuliert nach Nennung von Restriktionen, unter Berücksichtigung der Analyseergebnise, eine Antwort auf die Fragestellung der Diplomarbeit.

7. Fazit

In den zurückliegenden Kapiteln wurden wichtige Begriffe sowie die theoretischen Grundlagen, die für diese Arbeit wichtig sind, eingeführt. Von den Fragestellungen ausgehend, wurde ein Verfahren entwickelt, mit dem es möglich ist DSM zu erstellen. Das Verfahren wird mit einer eigenen Entwicklung vervollständigt. Diese ermöglicht das kompakte Darstellen von DSM. Nach einer Vorstellung der analysierten OSS wurden die Ergebnisse erst tabellarisch präsentiert und anschließend ausgewertet und interpretiert. Dieses Kapitel geht auf die Restriktionen, sowie die Stärken und Schwächen des Datengewinnungsverfahrens ein und erörtert das Forschungsergebnis.

7.1. Restriktionen

Die Zuordnung von OSS zum Geschäftsmodell ist schwierig. Die Zuordnungsmethodik aus Kapitel 2 funktioniert, dennoch ist die Anwendung komplizierter als zuvor angenommen. Bei den analysierten OSS geben die Unternehmen ihr Geschäftsmodell nicht bekannt. Anhand von Informationen, die von den Webseiten der OSS Anbieter stammen, wurde mit Hilfe der Zuordnungsmethodik versucht ein Geschäftsmodell zuzuordnen. Bei einigen Anbietern ist diese Zuordnung auf Grund der zur Verfügung stehenden Informationen auf den Webseiten sehr schwierig. Zusätzlich muss berücksichtigt werden, dass es sich bei der Zuordnung der Geschäftsmodelle zur OSS um eine Vermutung des Autors handelt. Diese Vermutungen beruhen auf den vorhandenen Informationen und der Zuordnungsmethodik, dennoch bleibt fraglich, ob diese Zuordnung für das Unternehmen gültig ist.

Für Unternehmen, die als sogenannte Non-Profit Unternehmen oder Non-Profit Organisationen tätig sind, konnten keine Geschäftsmodelle identifiziert werden. Diese Aussage reduziert sich jedoch nur auf die hier betrachteten Beispiele. Für eine generelle Aussage diesbezüglich, müssen mehrere Untersuchungen in dieser Richtung erfolgen. Generell müssten für solche Organisationen, die als Non-Profit Unternehmen tätig sind, neue Geschäftsmodelldefinitionen geschaffen und definiert werden. Vereinfacht gesagt versteht man unter einem Geschäftsmodell die Art und Weise wie Unternehmen Geld verdienen. Unternehmen, die von vornherein darauf ausgelegt sind, keinen Profit zu erwirtschaften, haben aber dennoch das Ziel die anfallenden Kosten zu decken. Dementsprechend müsste es Geschäftsmodelle geben, die diese Zwecke berücksichtigen und abdecken.

In Kapitel 4 wurde ein Verfahren entwickelt, mit dem die OSS analysiert werden kann. Bereits dort wurde darauf hingewiesen, dass es zu Problemen auf Grund der Größe der Projekte kommen kann. Leider konnten nicht alle Programme analysiert werden. Die Datenmenge reduziert sich daher auf die in Kapitel 5 vorgestellte OSS.

Da die Datenmenge nicht sehr umfangreich ist, müssen die Annahmen vorsichtig betrachtet werden. Sicherlich lässt sich daran eine Tendenz erkennen. Für eine umfassende und allgemein gültige Klärung des Sachverhaltes bedarf es weiterer Analysen.

7.2. Stärken und Schwächen

Für das Datengewinnungsverfahren wurde in Kapitel 3 gefordert, dass es mit den in der Literatur vorhandenen Daten verglichen wird. Der Abschnitt Datengewinnung 4.2 zeigt deutlich, dass die DSM durchaus mit denen in der Literatur vorhandenen DSM vergleichbar sind.

Bei der untersuchten OSS handelt es sich um Produkte aus verschiedenen Kategorien. Nach Möglichkeit wurde versucht in jeder Kategorie mindestens ein Programm zu finden, dem auch ein Geschäftsmodell zugeordnet werden kann. Zusätzlich wurden unterschiedliche Versionen der OSS betrachtet. Auf Grund dieser weit gestreuten OSS sind verschiedene Betrachtungsweisen und Fragestellungen möglich.

Der Datenanalyseprozess ist zeitaufwändig, liefert aber verschiedene nützliche Informationen. Von einfachen Program Reports, Berechnung der Propagation Cost und DSM. Diese verschiedenen Daten bieten eine gute Grundlage zur Beantwortung der Fragestellung.

7.3. Forschungsergebnis

Ein Zusammenhang zwischen Geschäftsmodell und OSS konnte nach Auswertung und Interpretation der gesammelten Daten nicht nachgewiesen werden. Es gibt einige Ideen, die für einen Zusammenhang zwischen OSS und Geschäftsmodell sprechen. Diese konnten aber mit Gegenbeispielen oder anderen Argumentationen und Fragestellungen widerlegt oder in Frage gestellt werden. Dass der Zusammenhang nicht nachgewiesen werden konnte, heißt nicht, dass es keinen Zusammenhang gibt. Es gibt noch Fragestellungen, die im Rahmen dieser Diplomarbeit nicht angesprochenen wurden. Zum Beispiel wird die Beziehung zwischen Geschäftsmodellen und Entwicklungsmodellen nicht berücksichtigt. Nach Evers [2008] stehen Geschäftsmodelle und Entwicklungsmodelle eng miteinander in Beziehung und müssen miteinander vereinbar sein [Eve08, S.17]. Es lässt sich nicht ausschließen, dass man für spätere Auswertungen das Entwicklungsmodell mit berücksichtigen muss.

In Kapitel 6.2 wurde beschrieben, das OSS modularer ist, als kommerzielle Software. Diese Erkenntnis ist das Ergebnis einer Untersuchung zwischen „open source“ und „closed source“ von MacCormack [MRB06] [MRB08]. Der Artikel [MRB08] zeigt sogar, dass sich die Struktur einer Organisation in der Software widerspiegelt. Hinter dem Entwicklungsansatz „colsed source“ stecken Firmen, mit vorgegebenen Organisationsstrukturen. Diese Organisationsstrukturen unterscheiden sich von den Organisationsstrukturen beim „open source“. Mit Hilfe der Modularität lässt sich ein Unterschied zwischen den beiden Ansätzen zeigen. Wenn man voraussetzt, dass ein Geschäftsmodell die Organisationsstruktur eines Unternehmens beeinflusst, dann existiert indirekt ein Zusammenhang zwischen OSS und Geschäftsmodell. Der Zusammenhang besteht indirekt, da das Geschäftsmodell die Organisationsstruktur beeinflusst, welche wiederum nachweislich (s. [MRB06] [MRB08]) einen Einfluss auf die Modularität des Quellcodes hat.

Dass Geschäftsmodelle einen Einfluss auf die Organisationsstruktur einer Unternehmung haben, zeigt Kapitel 2.2. Nachstehend werden einige offene Fragen genannt, die es noch zu klären gilt, um abschließend den Zusammenhang zwischen Geschäftsmodell und OSS bestätigen oder verneinen zu können.

7.4. Offene Fragen

Bereits eben genannt, sollte geklärt werden, ob verschiedene Geschäftsmodelle unterschiedliche Organisationsstrukturen haben. Ebenso wurde bereits angesprochen, ob es einen Zusammenhang zwischen Geschäftsmodell und Lizenz gibt. Sollte es dort einen Zusammenhang geben, so müsste geprüft werden, ob es einen Zusammenhang zwischen Lizenz und Quellcode gibt. Dieser Fragestellung könnte man nachgehen, in dem man prüft, ob sich das Geschäftsmodell beim Wechsel einer Lizenz ändert. Ließe sich dann eine Änderung im Quellcode nachweisen, die im ähnlichen Verhältnis zu ähnlichen Geschäftsmodellen und Lizenzwechsel stehen, dann wäre dies ein Zeichen für einen Zusammenhang. Ändert sich jedoch die Lizenz oder das Geschäftsmodell ohne eine strukturelle Änderung der Software, so wäre dies ein Zeichen dafür, dass es keinen Zusammenhang gibt.

7.5. Ausblick

Diese Arbeit hat gezeigt, dass Wirtschaftsinformatik eine Schnittstellenfunktion zwischen Betriebswirtschaftslehre und Informatik ist. Diese Grundlagenarbeit hat verdeutlicht, dass die Modularität einer Software nicht nur aus entwicklungs- und wartungsorientiertersicht wichtig ist, sondern auch Möglichkeiten und Ansatzpunkte für weitere Analysen liefert. Inoffiziell wurde dem Autor via Email mitgeteilt, das MacCormack seine Analyseergebnisse und DSM in Form einer Datenbank im Internet veröffentlichen wird. Mit Hilfe dieser Datenquelle und denen in dieser Arbeit gewonnenen Erkenntnissen und Verfahren, könnte den noch offenen Fragestellungen nachgegangen werden.

Literaturverzeichnis

[ABMS08] Asche, Michael (Hrsg.) ; Bauhus, Wilhelm (Hrsg.) ; Mitschke, Ernest (Hrsg.) ; Seel, Bernd (Hrsg.): *Open Source: Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen*. Waxmann, 2008

[Ale08] Alexy, Oliver ; Franke, Nikolaus (Hrsg.) ; Harhoff, Dietmar (Hrsg.) ; Henkel, Joachim (Hrsg.): *How Firms Can Profit From Being Open*. Technische Universität München, 2008 (Innovation und Entrepreneurship)

[BAPF04] Brügge, Prof. Dr. B. ; Arnold Picot, Prof. Dr. D. ; Fiedler, Dr. M. ; Brügge, Prof. Dr. B. (Hrsg.): *Open -Source -Software: Eine ökonomische und technische Analyse*. Springer-Verlag Berlin Heildlberg, 2004

[BC99] Baldwin, Carliss Y. ; Clark, Kim B.: *The Power of Modularity Volume 1*. MIT Press Cambridge, MA, USA, 1999 `http://portal.acm.org/citation.cfm?id=555152`

[BC00] Baldwin, Carliss Y. ; Clark, Kim B.: *Design Rules, Volume 1: The Power of Modularity*. Bd. 1. The MIT Press, 2000

[BS00] Brössler, Peter ; Siedersleben, Johannes ; Brössler, Peter (Hrsg.) ; Siedersleben, Johannes (Hrsg.): *Softwaretechnik: Praxiswissen für Software-Ingenieure*. Carl Hanser Verlag München Wien, 2000

[Cap09] Capiluppi, Andrea: Domain Drivers in the Modularization of FLOSS Sys-

tems. In: BOLDYREFF, Cornelia (Hrsg.) ; CROWSTON, Kevin (Hrsg.) ; LUNDELL, Björn (Hrsg.) ; WASSERMAN, Anthony I. (Hrsg.): *Open Source Ecosystems: Diverse Communities Interacting*. Springer-Verlag Berlin Heildlberg New York, 2009

[Die08] DIEDRICH, Oliver: Open Source im Jahr 2007. In: LUTTERBECK, Bernd (Hrsg.) ; BÄRWOLFF, Matthias (Hrsg.) ; GEHRING, Robert A. (Hrsg.): *Open Source Jahrbuch 2008: Zwischen freier Software und Gesellschaftsmodell*. Berlin : Lehmanns Media, 2008, S. 1–16

[Ebe98] EBERT, Jürgen: Die Geschichte von Gupro. In: EBERT, Jürgen (Hrsg.) ; GIMNICH, Rainer (Hrsg.) ; STASCH, Hans H. (Hrsg.) ; (HRSG.), Andreas W. (Hrsg.): *GUPRO Gernerische Umgebung zum Programmverstehen* Bd. 10. Fölbach, 1998, S. 3–10

[ERW08] EBERT, Jürgen ; RIEDIGER, Volker ; WINTER, Andreas: Graph Technology in Reverse Engineering, The TGraph Approach. In: GIMNICH, Rainer (Hrsg.) ; KAISER, Uwe (Hrsg.) ; QUANTE, Jochen (Hrsg.) ; WINTER, Andreas (Hrsg.): *10th Workshop Software Reengineering (WSR 2008)* Bd. 126. Bonn : GI, 2008, S. 67–81

[Eve08] EVERS, Steffen: *Ein Modell der Open-Source-Entwicklung*. Bd. 154. Technische Universität Berlin, 2008

[EWD+08] EBERT, Jürgen ; WINTER, Andreas ; DAHM, Peter ; FRANZKE, Angelika ; SÜTTENBACH, Roger: Graph-Based Modeling and Implementation with EER/GRAL. In: GIMNICH, Rainer (Hrsg.) ; KAISER, Uwe (Hrsg.) ; QUANTE, Jochen (Hrsg.) ; WINTER, Andreas (Hrsg.): *10th Workshop Software Reengineering (WSR 2008)* Bd. 126. Bonn : GI, 2008, S. 67–81

[EWSG94] EPPINGER, Steven D. ; WHITNEY, Daniel E. ; SMITH, Robert P. ; GEBALA, David A.: A model-based method for organizing tasks in product development. In: *Research in Engineering Design* 6 (1994), mar,

Nr. 1, 1-13. http://dspace.mit.edu/bitstream/handle/1721.1/2468/SWP-3569-29894176.pdf?sequence=1

[Hen04] Henkel, Joachim: Ökonomische und historische Aspekte. In: Picot, Arnold (Hrsg.) ; Doeblin, Stefan (Hrsg.): *Open Source: Tagungsband*. Bonn: Hüthig Telekommunikation, 2004, S. S.19 – 28

[Inta] Internetseite: *Call graph*. http://en.wikipedia.org/wiki/Call_graph, Abruf: 25.06.2009

[Intb] Internetseite: *Codequalität*. http://www.mysql.de/why-mysql/quality/, Abruf: 20.05.2009

[Intc] Internetseite: *Design Structure Matrix (DSM) (© Copyright 2009 Prof. Lindemann - Technische Universität Münschen)*. http://www.dsmweb.org/, Abruf: 25.05.2009

[Intd] Internetseite: *DorsalSource*. http://dorsalsource.org/, Abruf: 04.11.2009

[Inte] Internetseite: *DorsalSource.org - MySQL Enterprise zum kostenlosen Download Solid und Proven Scaling startet MySQL-Download-Website für die Community*. http://www.golem.de/0704/51920.html, Abruf: 04.11.2009

[Intf] Internetseite: *Free Software and the GNU Operating System*. http://www.fsf.org/about/, Abruf: 15.06.2009

[Intg] Internetseite: *The Free Software Definition*. http://www.gnu.org/philosophy/free-sw.html, Abruf: 14.06.2009

[Inth] Internetseite: *The GNU Manifesto*. http://www.gnu.org/gnu/manifesto.html, Abruf: 14.06.2009

[Inti] INTERNETSEITE: *History of the OSI.* `http://www.opensource.org/history`, Abruf: 15.06.2009

[Intj] INTERNETSEITE: *The Open Source Definition.* `http://www.opensource.org/docs/osd`, Abruf: 20.05.2009

[Intk] INTERNETSEITE: *Overview of the GNU System.* `http://www.gnu.org/gnu/gnu-history.html`, Abruf: 15.06.2009

[Intl] INTERNETSEITE: *Richard Stallman: Free Unix! im Usenet, 29. September 1983.* `http://groups.google.com/group/net.unix-wizards/msg/4dadd63a976019d7?output=gplain`, Abruf: 14.06.2009

[Intm] INTERNETSEITE: *SciTools - Maintaining your Software.* `http://www.scitools.com/products/understand/`, Abruf: 06.10.2006

[Intn] INTERNETSEITE: *Sea Monkey.* `http://www.seamonkey-project.org/`, Abruf: 05.11.2009

[Into] INTERNETSEITE: *Sun Press: Oracle to Buy Sun.* Internetseite: Sun press release. `http://www.sun.com/aboutsun/pr/2009-04/sunflash.20090420.1.xml`, Abruf: 04.05.2009

[Intp] INTERNETSEITE: *What is SourceForge.net?* `http://apps.sourceforge.net/trac/sourceforge/wiki/What%20is%20SourceForge.net?`, Abruf: 19.05.2009

[Intq] INTERNETSEITE: *Wikipedia: Computer Appliance.* `http://de.wikipedia.org/wiki/Computer_Appliance`, Abruf: 22.06.2009

[KA04] KNYPHAUSEN-AUFSESS, Prof. Dr. D.: Wie kann man mit Open Source Software Geld verdienen? In: PICOT, Arnold (Hrsg.) ; DOEBLIN, Stefan (Hrsg.): *Open Source: Tagungsband.* Bonn: Hüthig Telekommunikation, 2004, S. S.97 – 109

[Kno06] Knoblich, Werner: Erfolgreich mit Open Source - Das Red-Hat-Open-Source-Geschäftsmodell. In: *Open Source Jahrbuch 2006 - Zwischen Softwareentwicklung und Gesellschaftsmodell*. Bern Lutterbeck and Matthias Bärwolff and Robert A. Cehring, 2006, S. 155 – 166

[Kri05] Krishnamurthy, Sandeep: An Analysis of Open Source Business Models. In: Feller, Joseph (Hrsg.) ; Fitzgerald, Brian (Hrsg.) ; Hissam, Scott A. (Hrsg.) ; Lakhani, Karim R. (Hrsg.): *Perspectives on Free and Open Source Software*. MIT Press, 2005, S. 279 – 296

[Lat04] Lattix ; Lattix, Inc. (Hrsg.): *The Lattix (TM) Approach - DSM for Managing Software Architecture*. Lattix, Inc., nov 2004

[Lei04] Leiteritz, Raphael: Open Source-Geschäftsmodelle. In: Gehring, Robert A. (Hrsg.) ; Lutterbeck, Bernd (Hrsg.): *Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodel*. http://www.opensourcejahrbuch.de/2004/ : Lehmanns Media, Berlin, 2004, S. 153–186

[MMW09] Milev, Roberto ; Muegge, Steven ; Weiss, Michael: Design Evolution of an Open Source Project Using an Improved Modularity Metric. In: Boldyreff, Cornelia (Hrsg.) ; Crowston, Kevin (Hrsg.) ; Lundell, Björn (Hrsg.) ; Wasserman, Anthony I. (Hrsg.) ; (eds.) (Hrsg.): *Open Source Ecosystems: Driverse Communities Interacting*. Springer, jun 2009, S. 20–33

[MRB06] MacCormack, Alan ; Rusnak, John ; Baldwin, Carliss Y.: Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. In: *Management Science* 52 (2006), jul, Nr. 7, S. 1015–1030

[MRB08] MacCormack, Alan ; Rusnak, John ; Baldwin, Carliss Y.: Exploring the Duality between Product and Orgnizational Architecturees: A Test of the Moirroring Hypothesis. In: *Working Paper* (2008), oct

[OHFA04] Oloufa, Amr A. ; Hosni, Yasser A. ; Fayez, Mohamed ; Axelsson, Pär: Using DSM For Modeling Information Flow In Construction Design Projects. In: *Civil Engineering and Environmental Systems* 21 (2004), jun, Nr. 2, S. 105–125

[Ray] Raymond, Eric S.: *The Magic Cauldron (June 1999) (Internetseite).* `http://catb.org/~esr/writings/magic-cauldron/magic-cauldron-9.html#ss9.2`, Abruf: 14.11.2009

[Ray01] Raymond, Eric S. ; Raymond, Eric S. (Hrsg.): *The Cathedral and the Bazaar: Musings on Linux nd Open Source by an Accidental Revolutionary.* O'Reilly Media, 2001

[Sci09] SciTools ; Scientific Toolworks, Inc. (Hrsg.): *Understand - User Guide and Reference Manual.* http://getunderstand.com/documents/manuals/pdf/understand.pdf: Scientific Toolworks, Inc., jan 2009

[SDL03] Scheer, Christian ; Deelmann, Thomas ; Loos, Peter: Geschäftsmodelle und internetbasierte Geschäftsmodelle - Begriffsbestimmung und Teilnehmermodell. In: *ISYM - Information Systems und Management* (2003)

[SER03] Sosa, Manuel E. ; Eppinger, Steven D. ; Rowles, Craig M.: Identifying modular and integrative systems and their impact on design team interactions. In: *ASME Journal of Mechanical Design* 125 (2003), jun, S. 240–252

[Stä01] Stähler, Patrick ; Szyperski, Nobert (Hrsg.) ; Schmid, Beat F. (Hrsg.) ; Scheer, August-Wilhelm (Hrsg.) ; Pernul, Günther (Hrsg.) ; Klein, Stefan (Hrsg.): *Staehler2001Geschäftsmodelle in der digitalen Ökonomie.* Josef Eul Verlag, 2001

[Ste81] Steward, Donald V.: Design Structure System: A Method for Managing the Design of Complex Systems. In: *IEEE TRANS. ENG. MGMT* EM28, no. 3 (1981), S. 7174

[Ulr95] Ulrich, Karl: The Role of Product Architecture in the Manufacturing Firms. In: *Research Policy* 24 (1995), 419-440. http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V77-3YCN1YX-16&_user=10&_coverDate=05%2F31%2F1995&_rdoc=1&_fmt=high&_orig=browse&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=5e6680271e426e337469465b5298c36a

[Weba] Webseite: *Dependency Finder*. http://depfind.sourceforge.net/, Abruf: 01.10.2009

[Webb] Webseite: *Design Structure Matrix*. http://en.wikipedia.org/wiki/Design_Structure_Matrix, Abruf: 01.10.2009

[Webc] Webseite: *Gecko (Software)*. http://de.wikipedia.org/wiki/Gecko_(Software), Abruf: 07.07.2009

[Webd] Webseite: *http://kcachegrind.sourceforge.net/cgi-bin/show.cgi*. KCachegrind-ProfilingVisualization

[Webe] Webseite: *jDSM - Java DSM library*. http://jdsm.sourceforge.net/, Abruf: 01.10.2009

[Webf] Webseite: *Layout-Engine*. http://de.wikipedia.org/wiki/Layout-Engine, Abruf: 07.07.2009

[Webg] Webseite: *Licenses*. http://www.gnu.org/licenses/licenses.html, Abruf: 15.06.2009

[Webh] Webseite: *Open-Source ausschlaggebend für das erfolgreiche Geschäftsmodell von MySQL*. http://www.mysql.de/news-and-events/generate-article.php?id=2002_9, Abruf: 27.06.2009

[Webi] WEBSEITE: *Software quality depends on the architecture.* `http://www.lattix.com/`, Abruf: 06.10.2009

[Webj] WEBSEITE: *Webauftritt des Prouktes MySQL von Sun Microsystems.* `http://www.mysql.de`, Abruf: 03.07.2009

[Yuh04] YUHANNA, Noel: Open Source Databeses Come Of Age. In: *Forrester* - (2004), dec, S. 1–17

A. Email Alan MacCormack

Matthias Gerz

Von: Alan MacCormack [alanmac@MIT.EDU]
Gesendet: Donnerstag, 6. August 2009 14:45
An: UNI
Betreff: Re: Software analyze and DSM Tool

Hi Matthias,

Glad you are working in this field - we welcome everyone who is building on the idea of using DSMs for software! We don't currently distribute our software, as we are upgrading it to include new languages and new metrics. But I will certainly let you know when we decide it is robust enough to release externally.

I would be very interested in what you are doing to visualize DSMs - we also know the problems in viewing large systems with thousands of source files. A viewer would be a great addition for people who wish to create a real-time tool for aiding software development. Some guys from BCG did this recently with some social network type software - you could zoom in and out of the system to examine particular dependencies. I think they wrote a book on architecture which you may be able to track down. We gave them data from Mozilla on source file calling patterns so they could experiment with their software.

I'd be happy to look at any prototype versions you have as you build your system, plus if I get across to Europe for a conf in the near future I will let you know.

Good luck, and all the best!

Cheers, Alan.

Alan MacCormack
Visiting Associate Professor
MIT Sloan School of Management

On Aug 4, 2009, at 5:16 PM, UNI wrote:

Dear Alan MacCormack!

I am a computer scientist student and I am writing my diploma thesis.
On the basis of your article “Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code“ is my topic to analyze open source software in addition to business models. I use the software understand 2.0. I assume that is the latest version of the software which you used “anderstand c++”.
My problem is that DSM tools (e.g. Lattix) create to large DSM, because they show to many details in the graphic and have less options. So I am programming my own DSM-Viewer which work up with the results from the software understand 2.0.
I hope to paint the same pictures like you from the Linux kernel and Mozilla Browser.
May it is possible to use your software too, to calculate and paint DSM?
Please send me an email if you need more information about my thesis or about me.

I am looking forward to your answer!

Kind regards

Matthias Gerz

B. Java Fehlermeldung

Java Fehlermeldung bei zu großen Dateien.

```
Exception in thread "AWT-EventQueue-0" java.lang.OutOfMemoryError: Java heap space
at StringMatrix.<init>(StringMatrix.java:11)
at CsvReader.read(CsvReader.java:77)
at Gui.actionPerformed(Gui.java:159)
at javax.swing.AbstractButton.fireActionPerformed(Unknown Source)
at javax.swing.AbstractButton$Handler.actionPerformed(Unknown Source)
at javax.swing.DefaultButtonModel.fireActionPerformed(Unknown Source)
at javax.swing.DefaultButtonModel.setPressed(Unknown Source)
at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(Unknown Source)
at java.awt.Component.processMouseEvent(Unknown Source)
at javax.swing.JComponent.processMouseEvent(Unknown Source)
at java.awt.Component.processEvent(Unknown Source)
at java.awt.Container.processEvent(Unknown Source)
at java.awt.Component.dispatchEventImpl(Unknown Source)
at java.awt.Container.dispatchEventImpl(Unknown Source)
at java.awt.Component.dispatchEvent(Unknown Source)
at java.awt.LightweightDispatcher.retargetMouseEvent(Unknown Source)
at java.awt.LightweightDispatcher.processMouseEvent(Unknown Source)
at java.awt.LightweightDispatcher.dispatchEvent(Unknown Source)
at java.awt.Container.dispatchEventImpl(Unknown Source)
at java.awt.Window.dispatchEventImpl(Unknown Source)
at java.awt.Component.dispatchEvent(Unknown Source)
```

```
at java.awt.EventQueue.dispatchEvent(Unknown Source)
at java.awt.EventDispatchThread.pumpOneEventForFilters(Unknown Source)
at java.awt.EventDispatchThread.pumpEventsForFilter(Unknown Source)
at java.awt.EventDispatchThread.pumpEventsForHierarchy(Unknown Source)
at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
at java.awt.EventDispatchThread.run(Unknown Source)
```

C. Vergleich Datengewinnung

Dieser Abschnitt zeigt die selbst erstellten DSM im Vergleich zu den DSM von Alan MacCormack. Zuerst wird eine frühe Version von Linux (Linux Kernel) verglichen. Im Anschluss wird die DSM der OSS GnuCash verglichen.

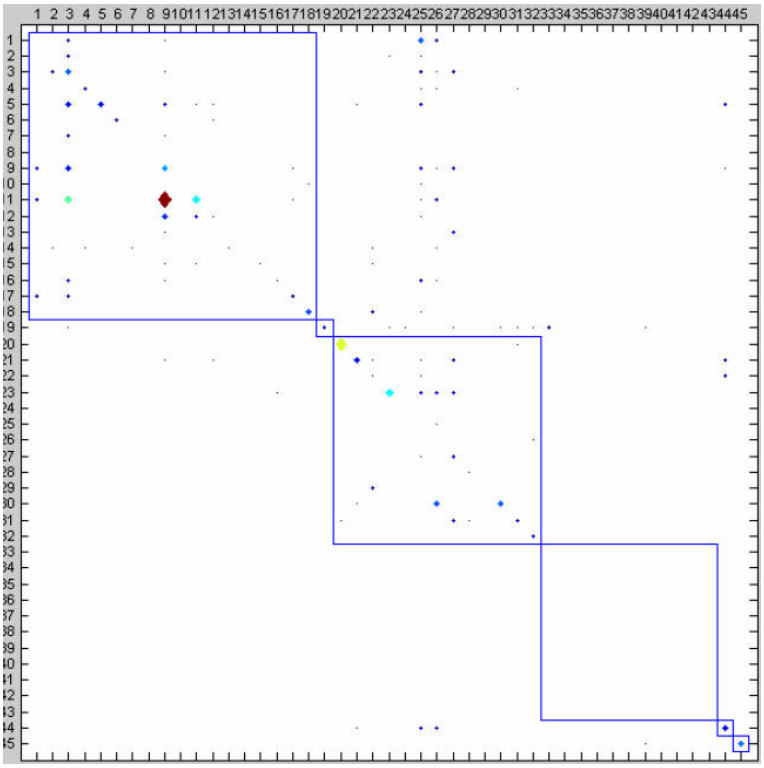


Abbildung C.1.: MacCormack: Frühe Version des Linux Kernels (Version 0.01)

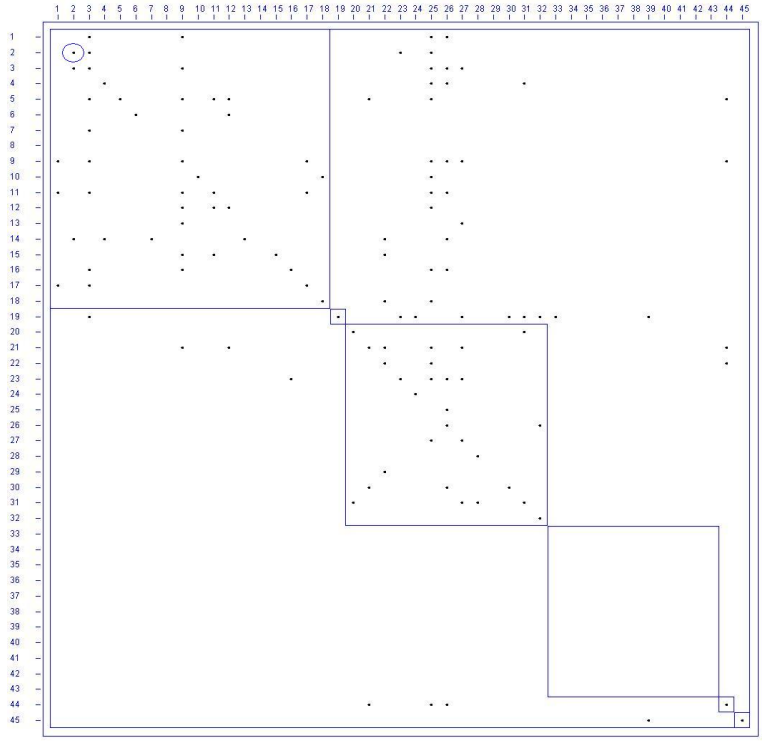


Abbildung C.2.: Ergebnis des Datengewinnungsprozesses: Frühe Version des Linux Kernels (Version 0.01)

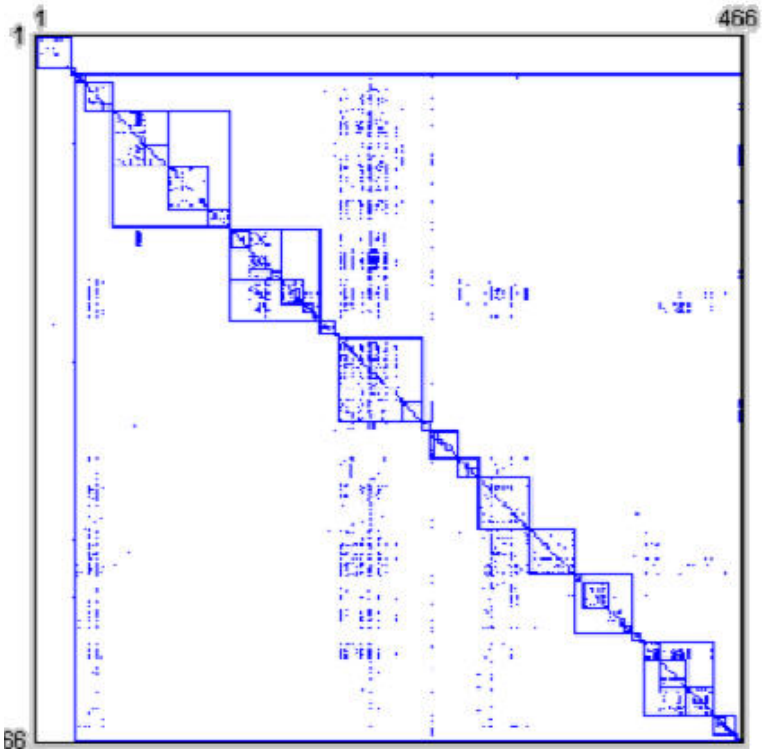


Abbildung C.3.: MacCormack: Gnucash 1.8.4

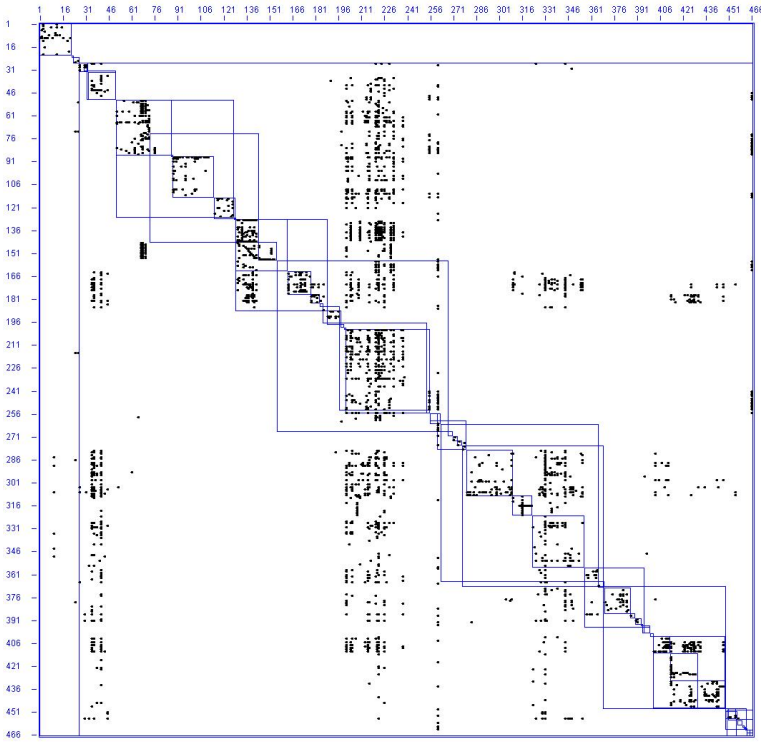


Abbildung C.4.: Ergebnis des Datengewinnungsprozesses: Gnucash 1.8.4

D. Analyse Ergebnisse Bilder

Die nachstehenden Seiten zeigen die Ergebnisse der Programmanalyse. Zuerst werden die einzelnen DSM aufgelistet. Die Anordnung der Programme erfolgt in den im Kapitel 5 genannten Kategorien. Die Reihenfolge der Kategorien ist der nachstehenden Liste zu entnehmen.

1. Applikationsserver
2. Browser
3. CRM Systeme
4. Datenbanken
5. Datensicherung
6. Financial
7. Linux

Der Name des Programms und die entsprechende Version ist der Abbildungsbezeichnung zu entnehmen. Gleich im Anschluss an die einzelnen DSM werden die aufbereiteten Project Overview gezeigt. Analog zu den DSM können die Programmbezeichnung und Versionsangaben der Abbildungsbezeichnung entnommen werden. Bei den Project Overview gibt es sechs verschiedene Angaben. In dieser angegebenen Reihenfolge sind die Abbildungen je Programm und Version abgebildet:

- code breakdown
- function breakdown
- most complex files
- most complex functions
- largest files
- largest functions

Den Namen der einzelnen Project Overview ist der Abbildungsbezeichnung zu entnehmen. Eine Beschreibung der einzelnen Project Overviews liefert das Kapitel 4.1.3.

D.1. Design Structure Matrix

Dieses Kapitel zeigt die DSM der einzelnen Programme und den jeweiligen Versionen. Den Programmnamen und die entsprechende Version ist der Abbildungsbezeichnung zu entnehmen.

D.1.1. Applikationsserver

D.1.2. Browser

D.1.3. CRM Systeme

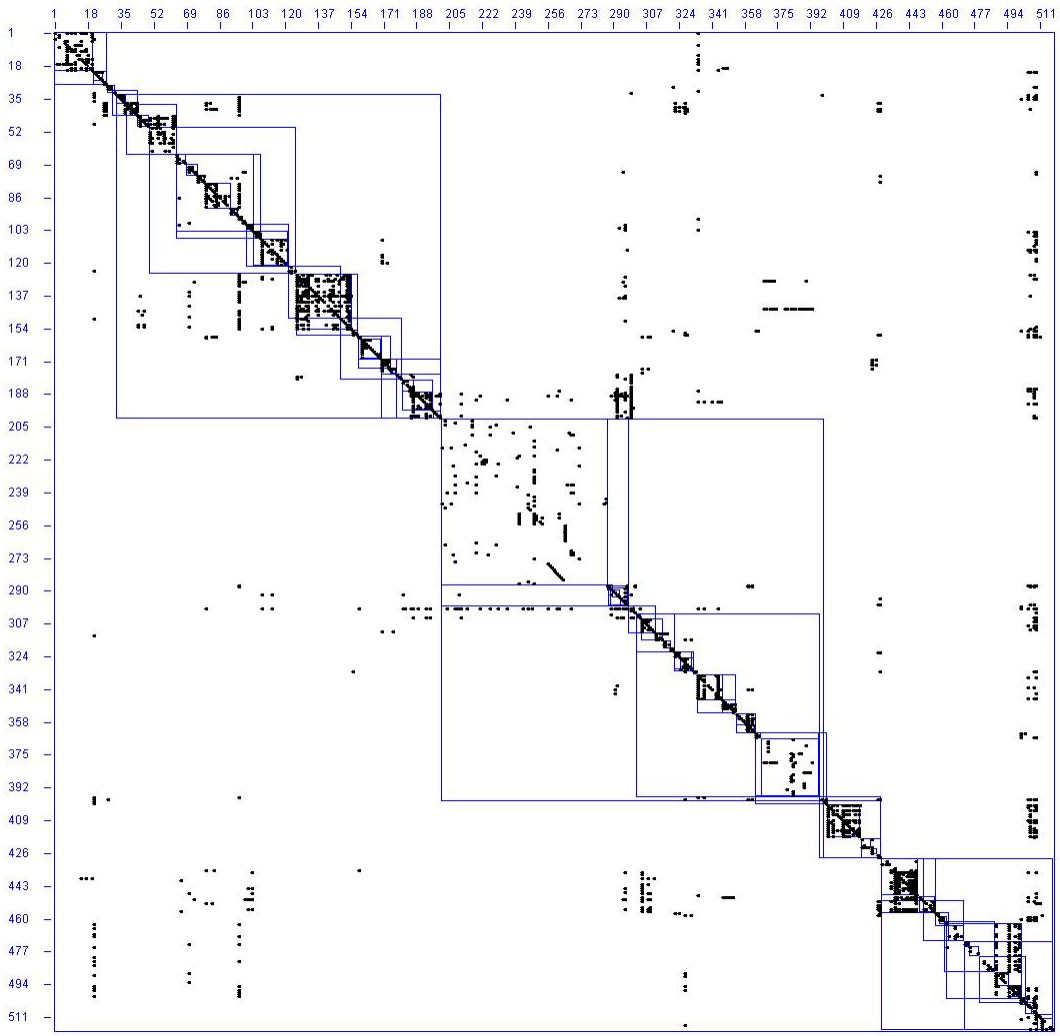


Abbildung D.1.: DSM jBoss EAP Version 2.0-beta1

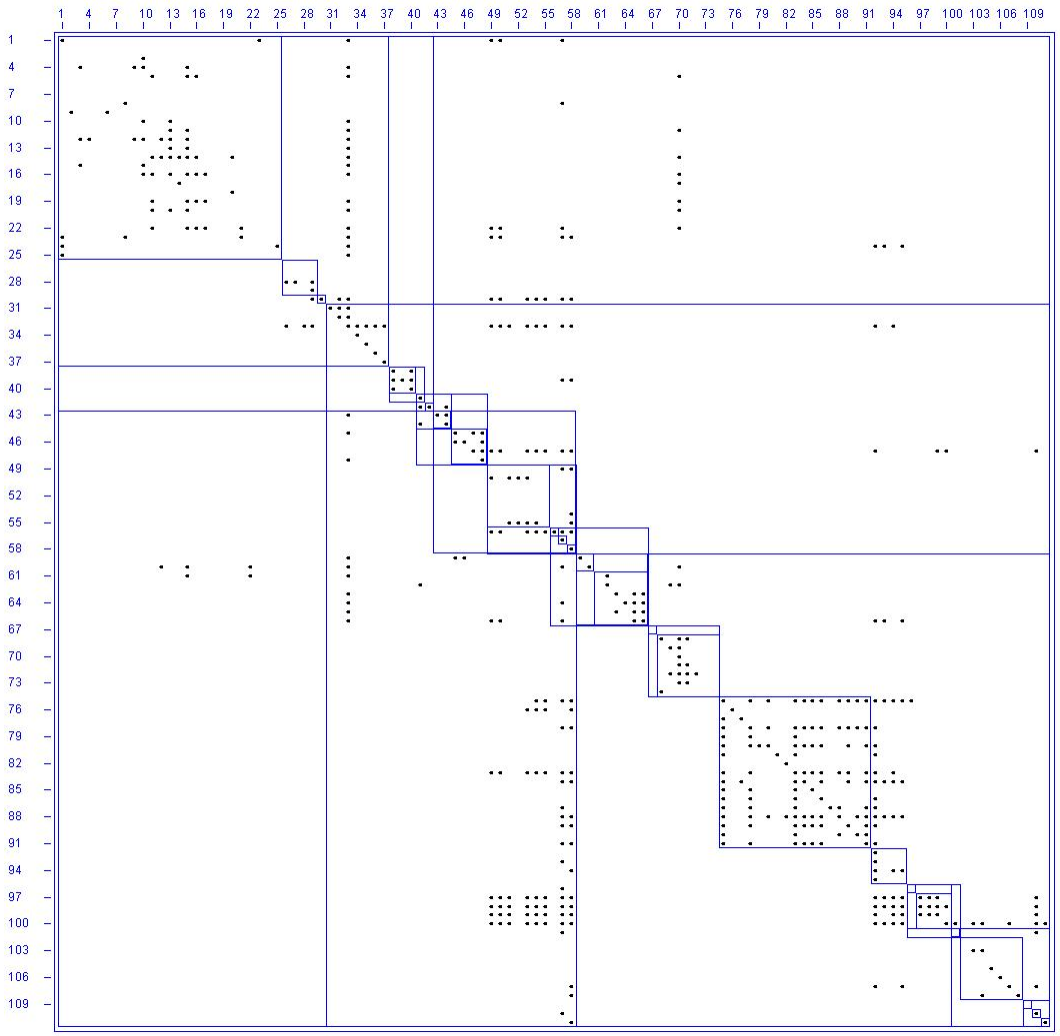


Abbildung D.2.: DSM jBoss EAP Version 2.6.2.GA

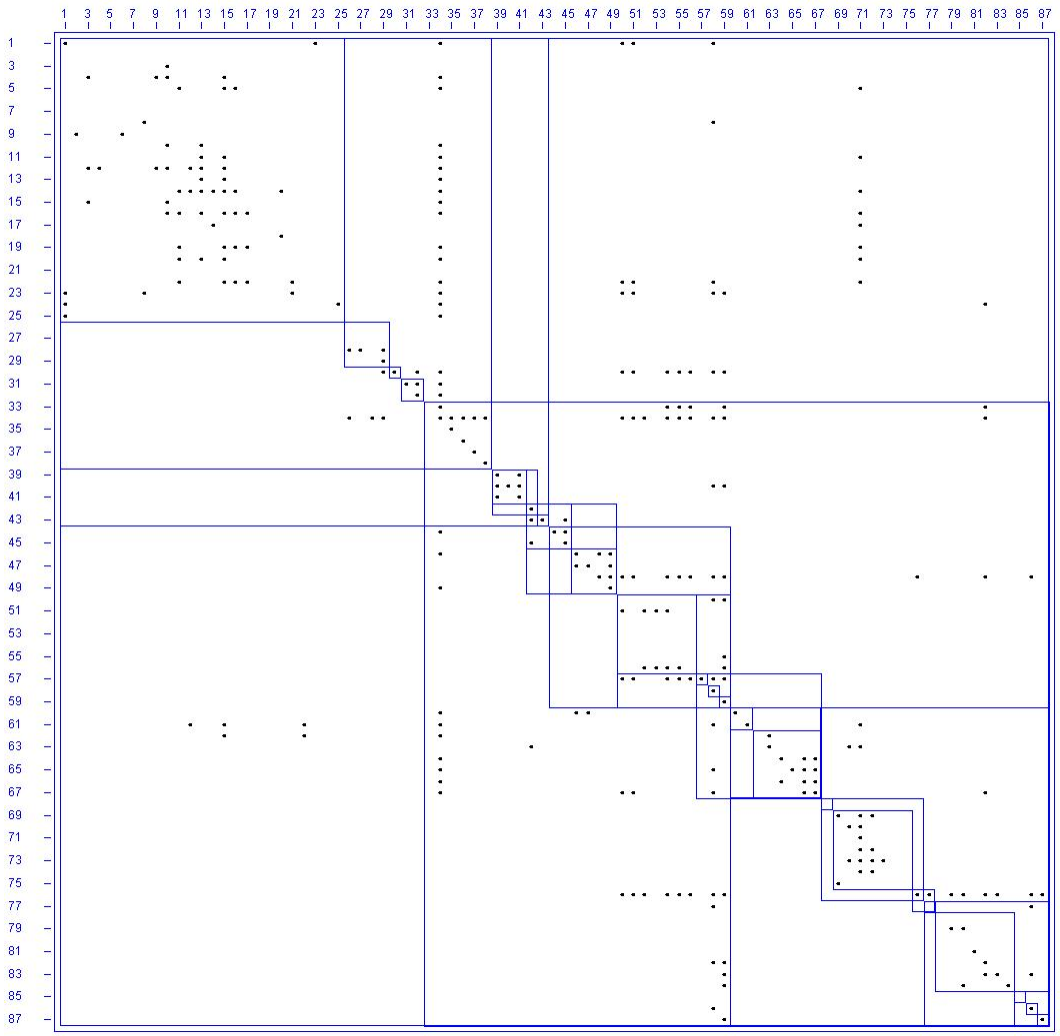


Abbildung D.3.: DSM jBoss EAP Version 2.7.0.CR1

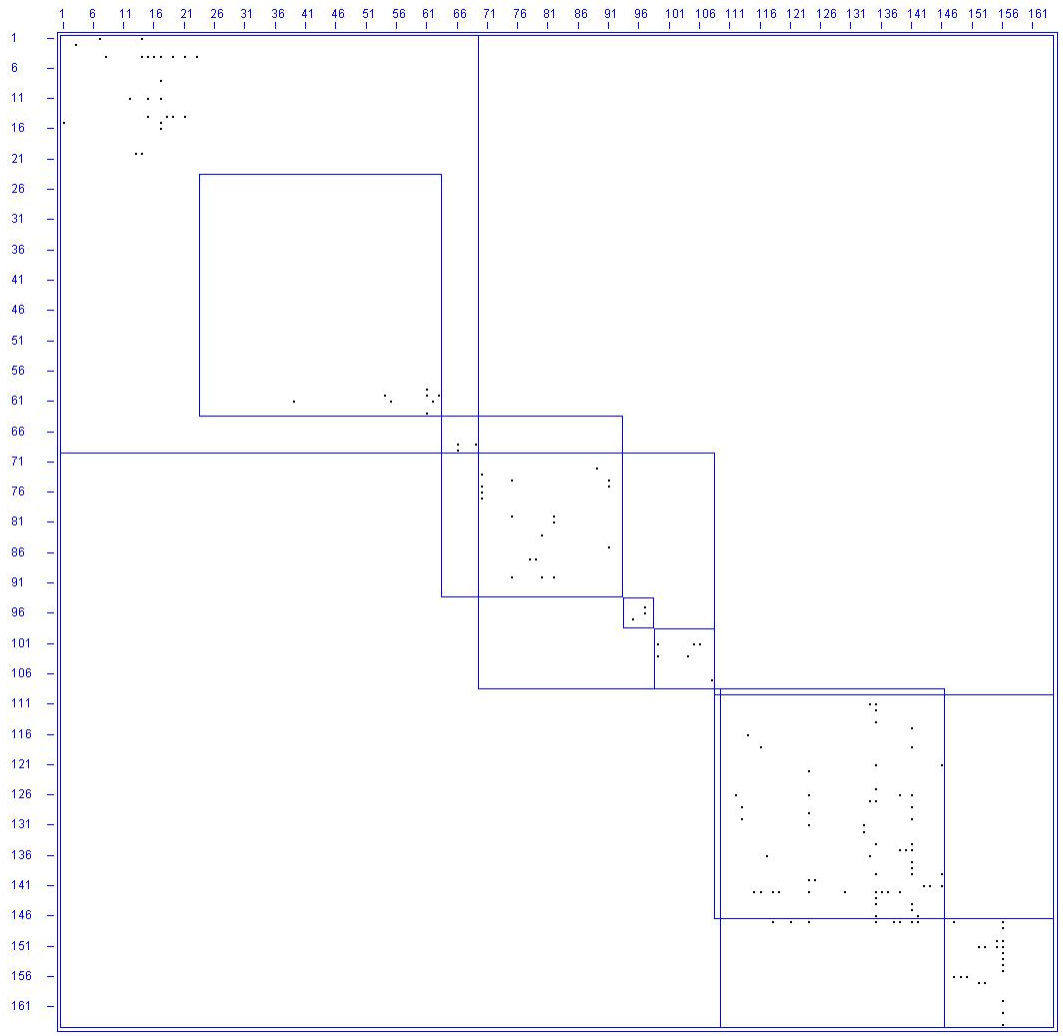


Abbildung D.4.: DSM Epiphany Version 2.22.3

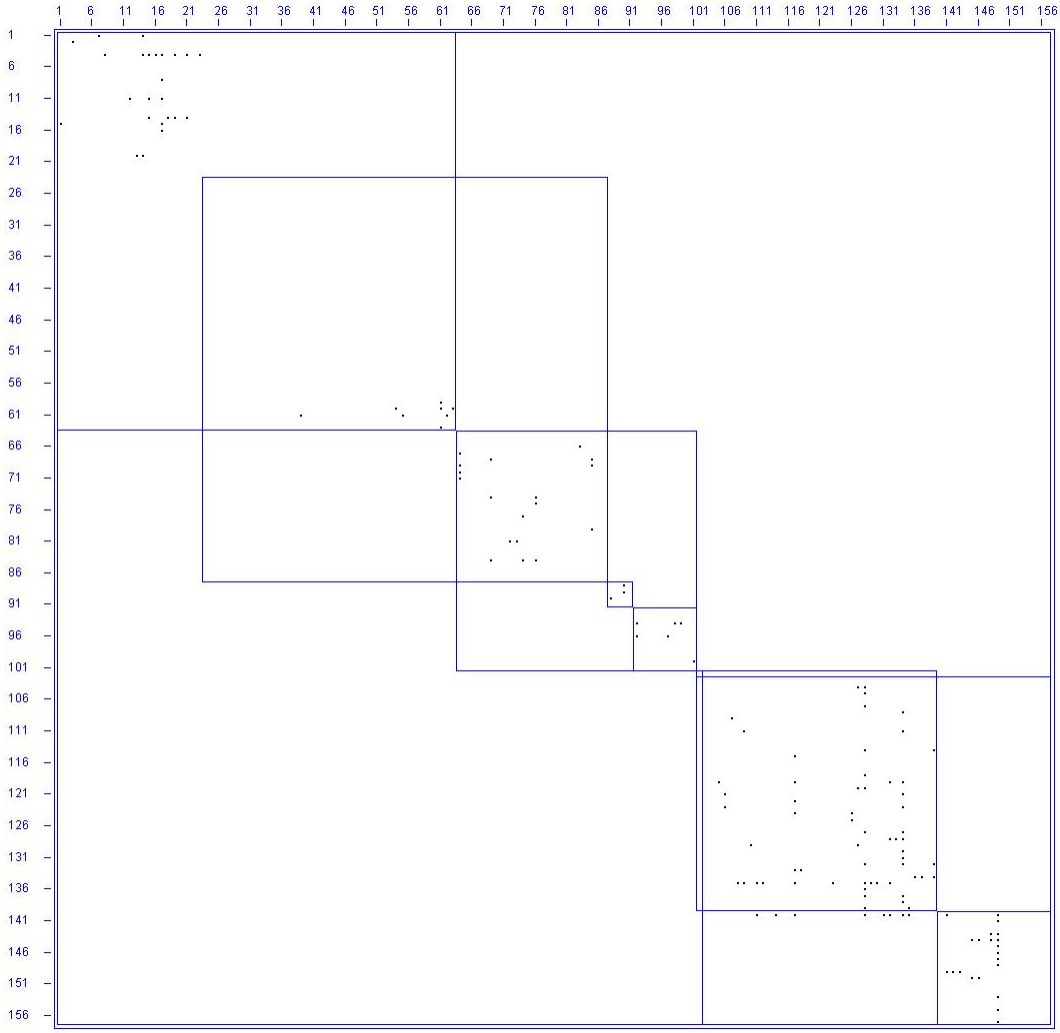


Abbildung D.5.: DSM Epiphany Version 2.26.1

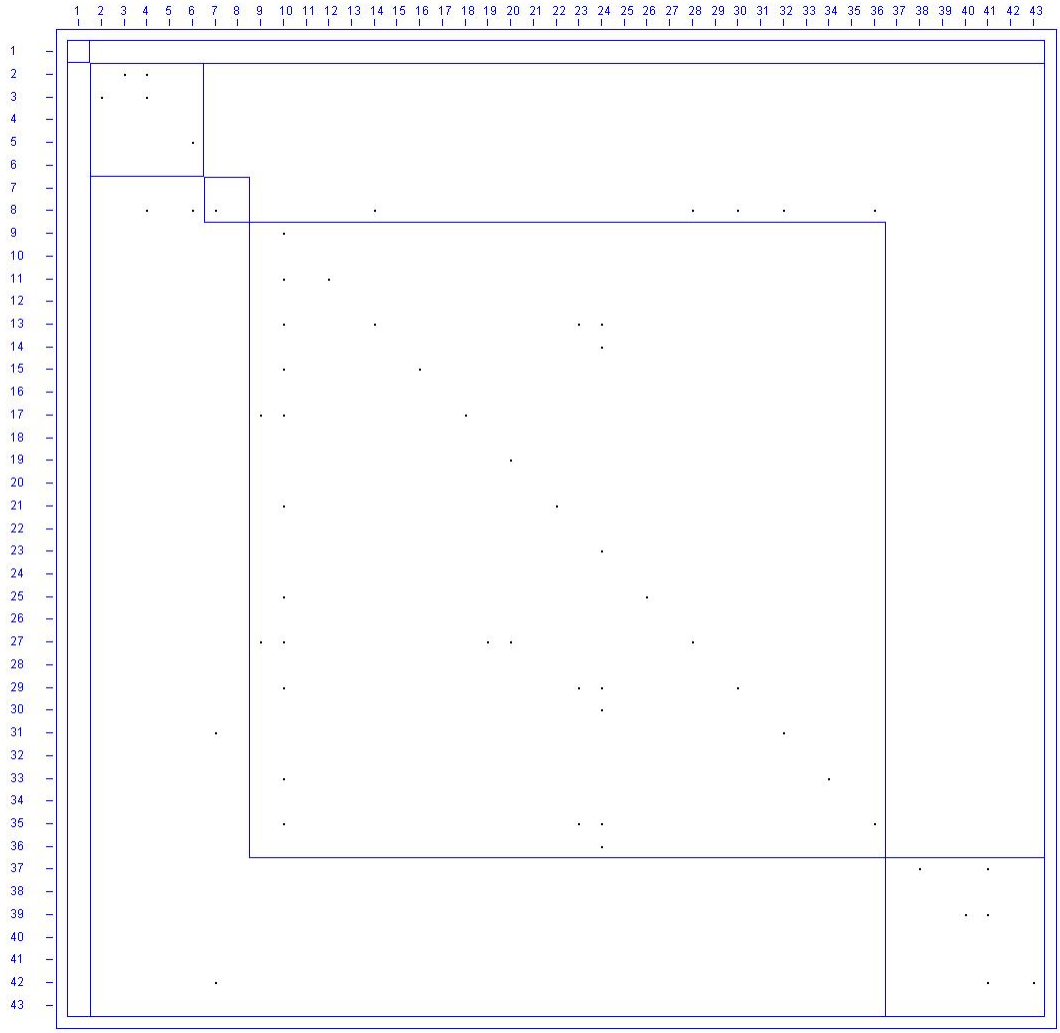


Abbildung D.6.: DSM Mozilla Firefox Version 1.0

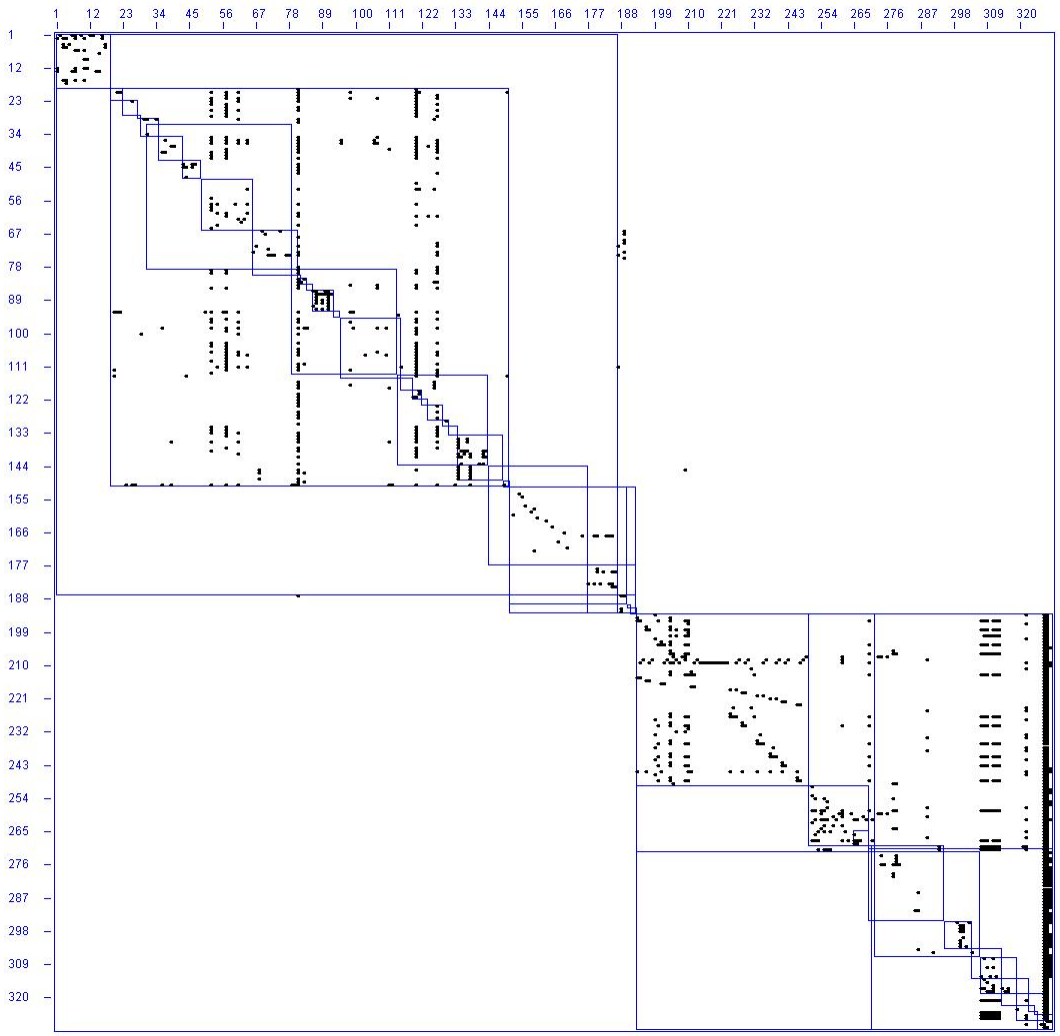


Abbildung D.7.: DSM Hipergate Version 1.0.4-beta

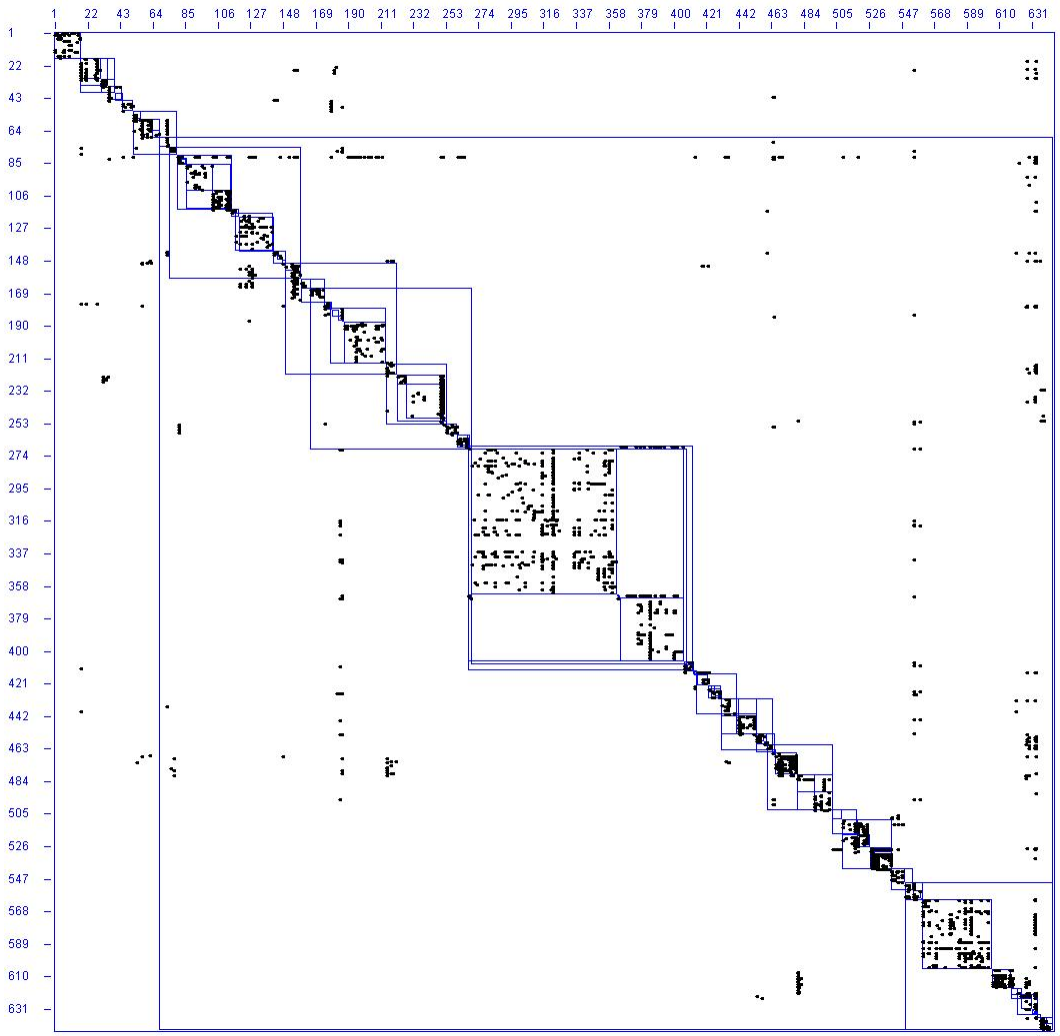


Abbildung D.8.: DSM Hipergate Version 3.0

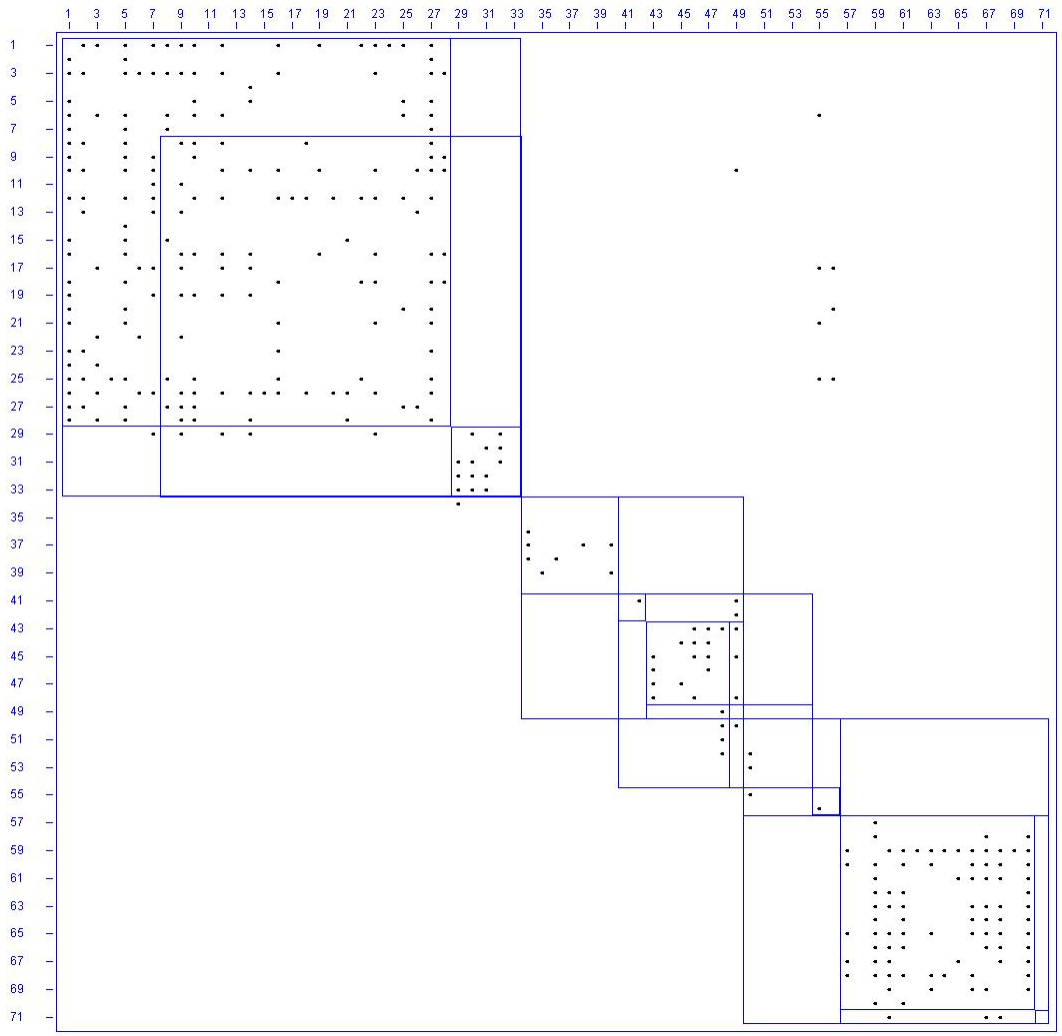


Abbildung D.9.: DSM Hipergate Version 4.0

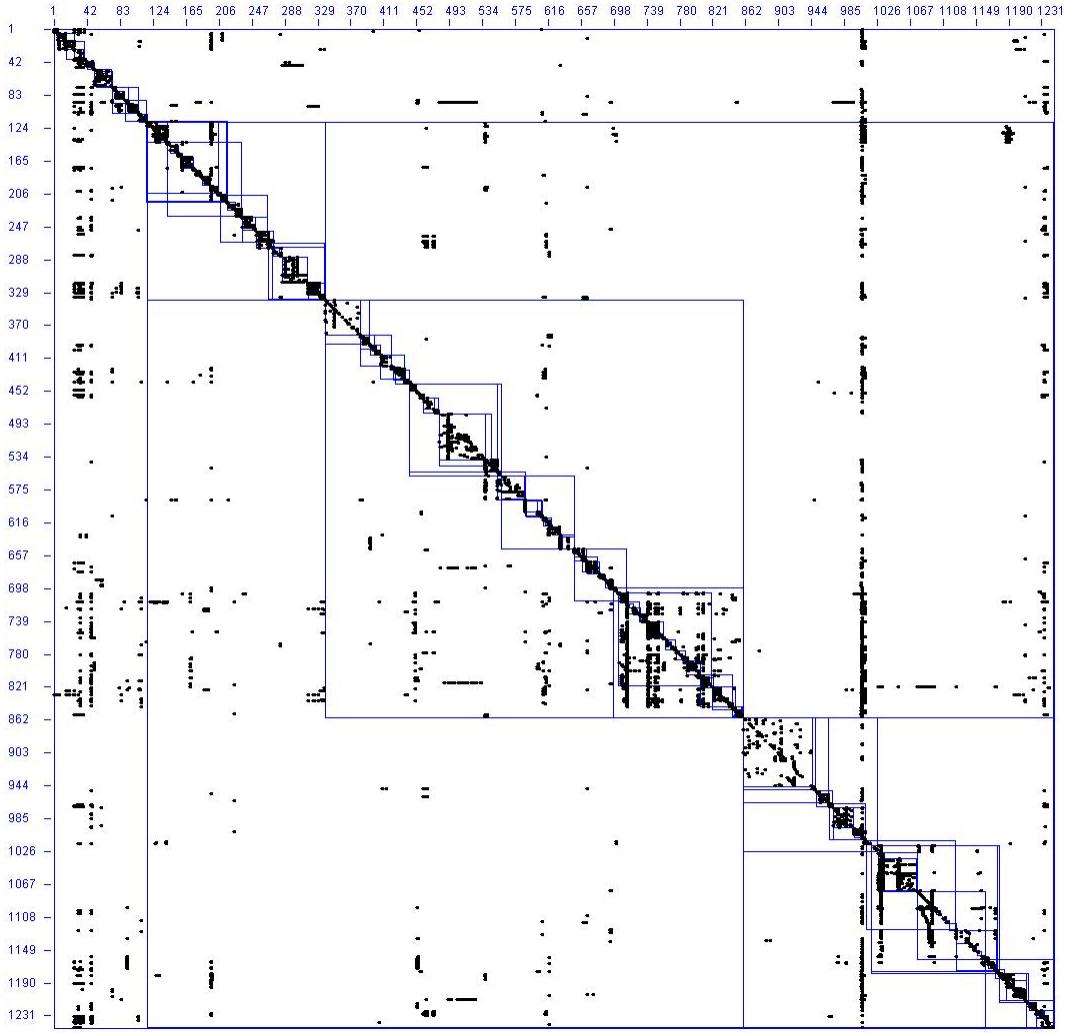


Abbildung D.10.: DSM Opentaps Version 0.8.5

D.1.4. Dantenbanken

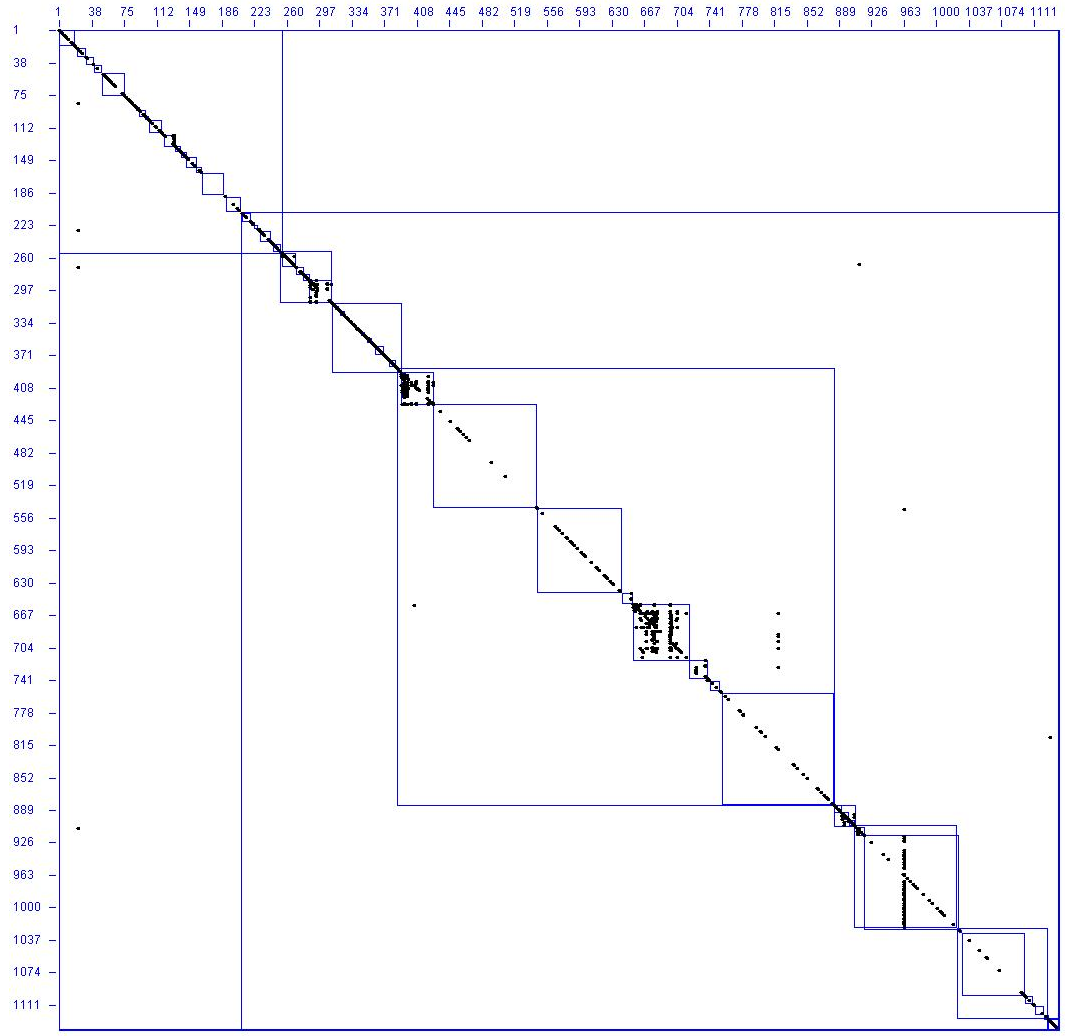


Abbildung D.11.: DSM MySql Version 4.1.22

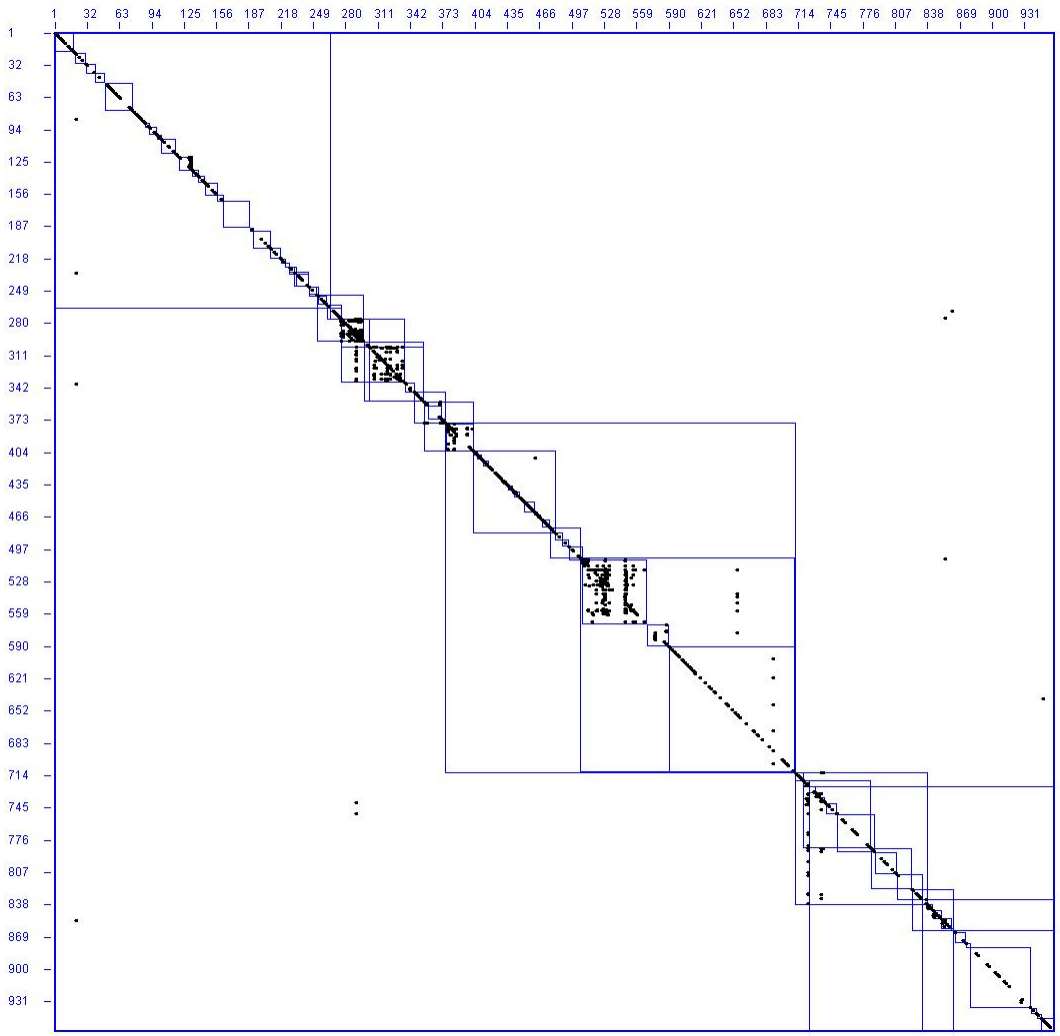


Abbildung D.12.: DSM MySql Version 5.0.83

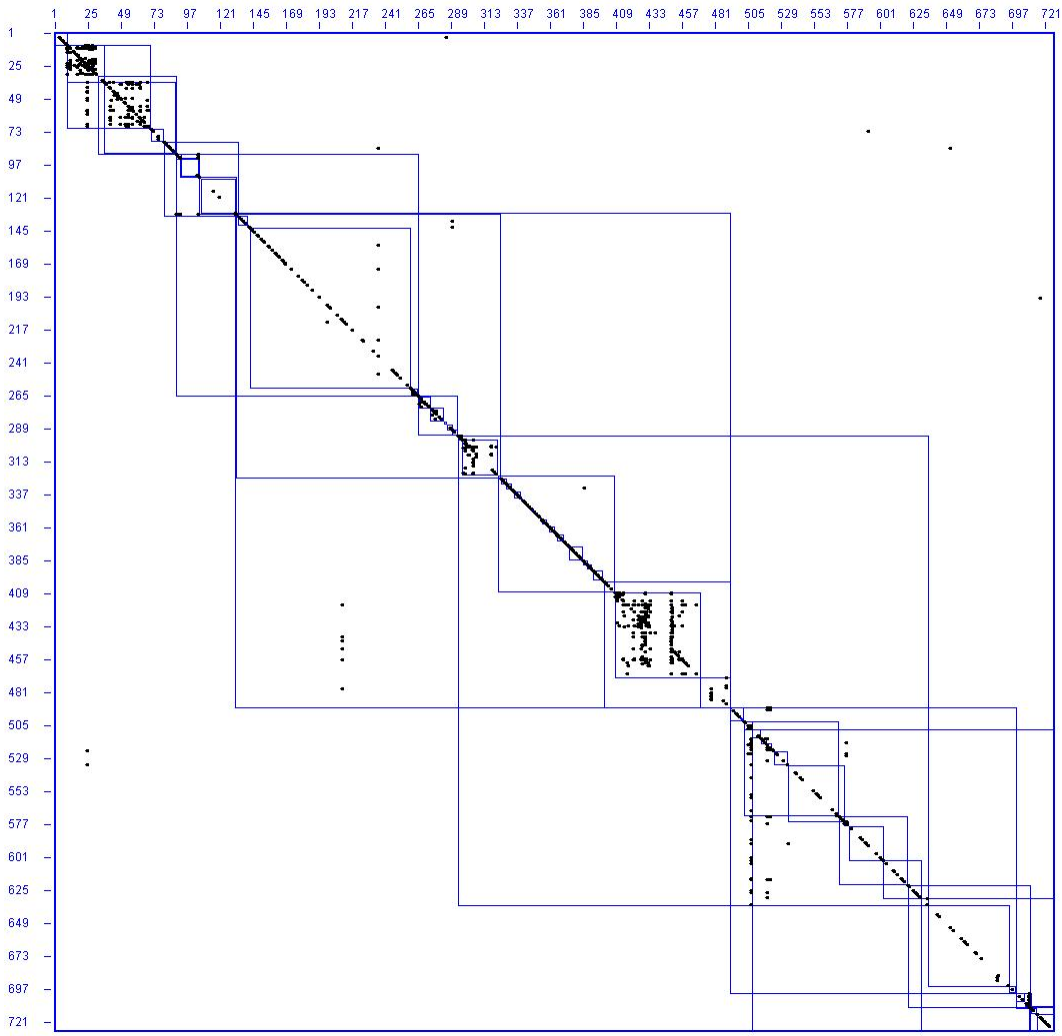


Abbildung D.13.: DSM MySql Version 5.4.1-beta

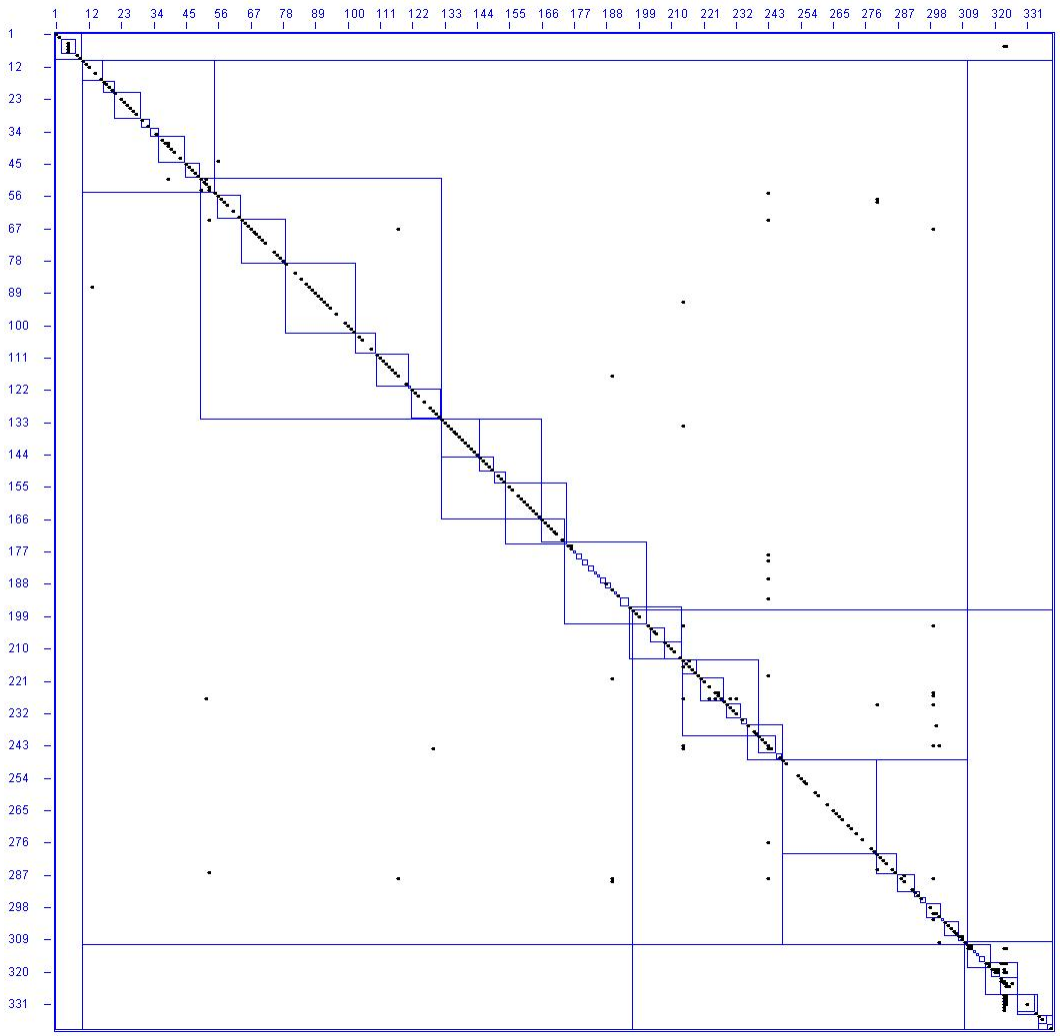


Abbildung D.14.: DSM Postgres Version v60

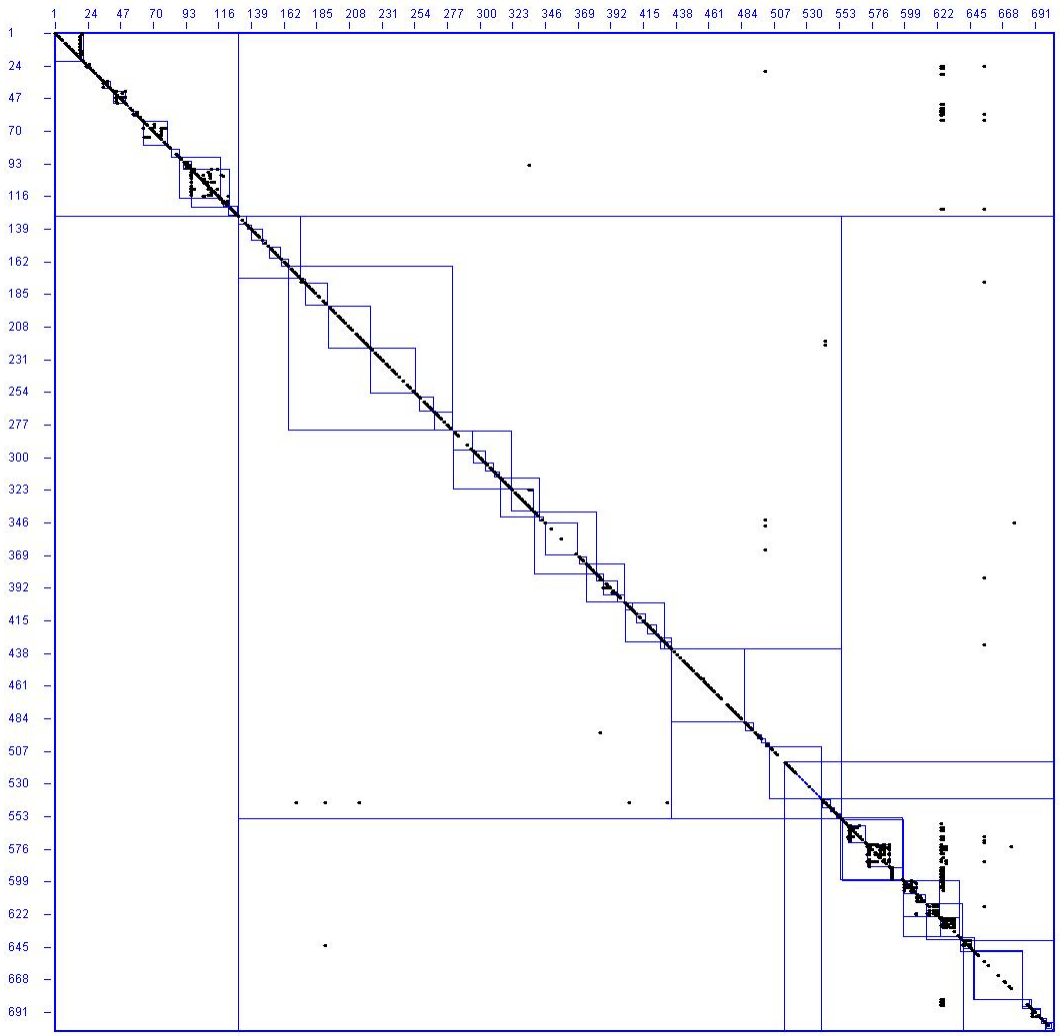


Abbildung D.15.: DSM Postgres Version 8.0.19

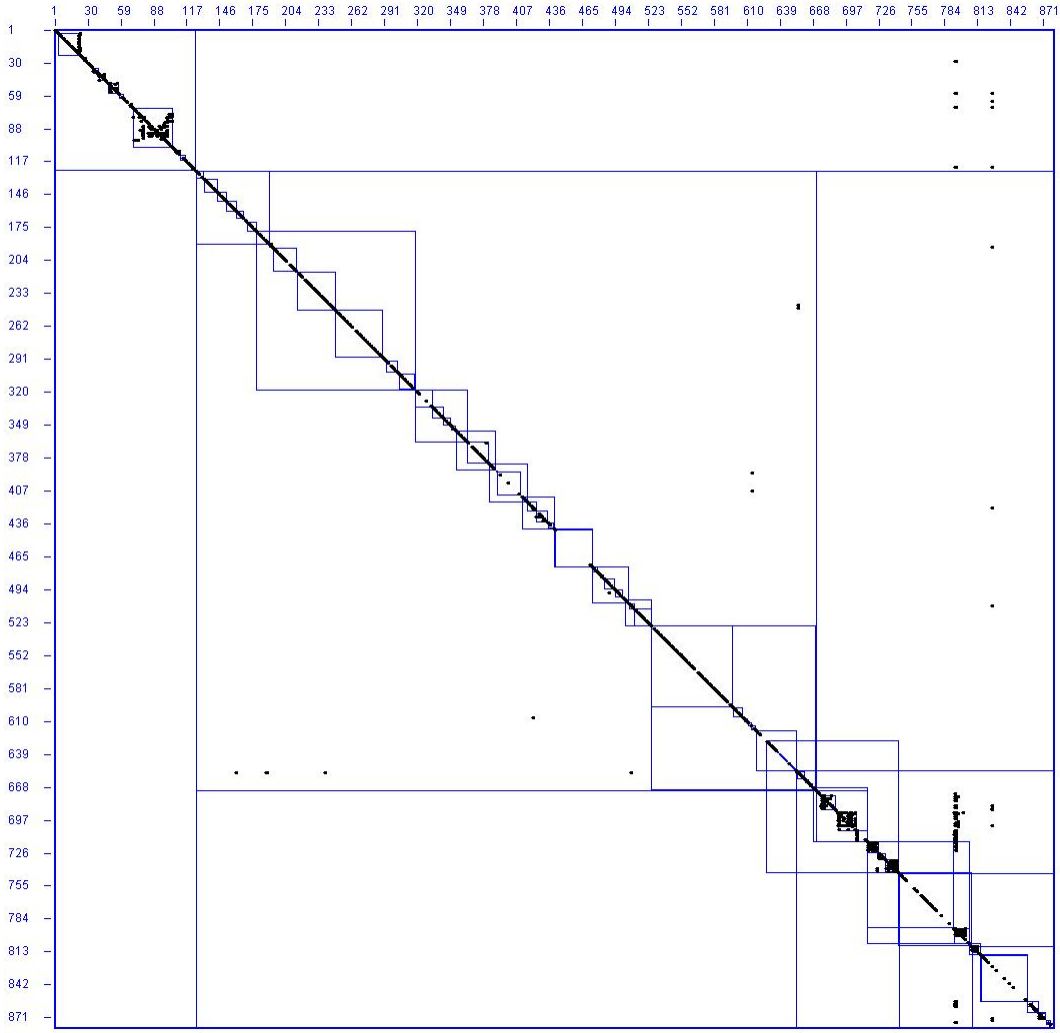


Abbildung D.16.: DSM Postgre Version 8.5alpha1

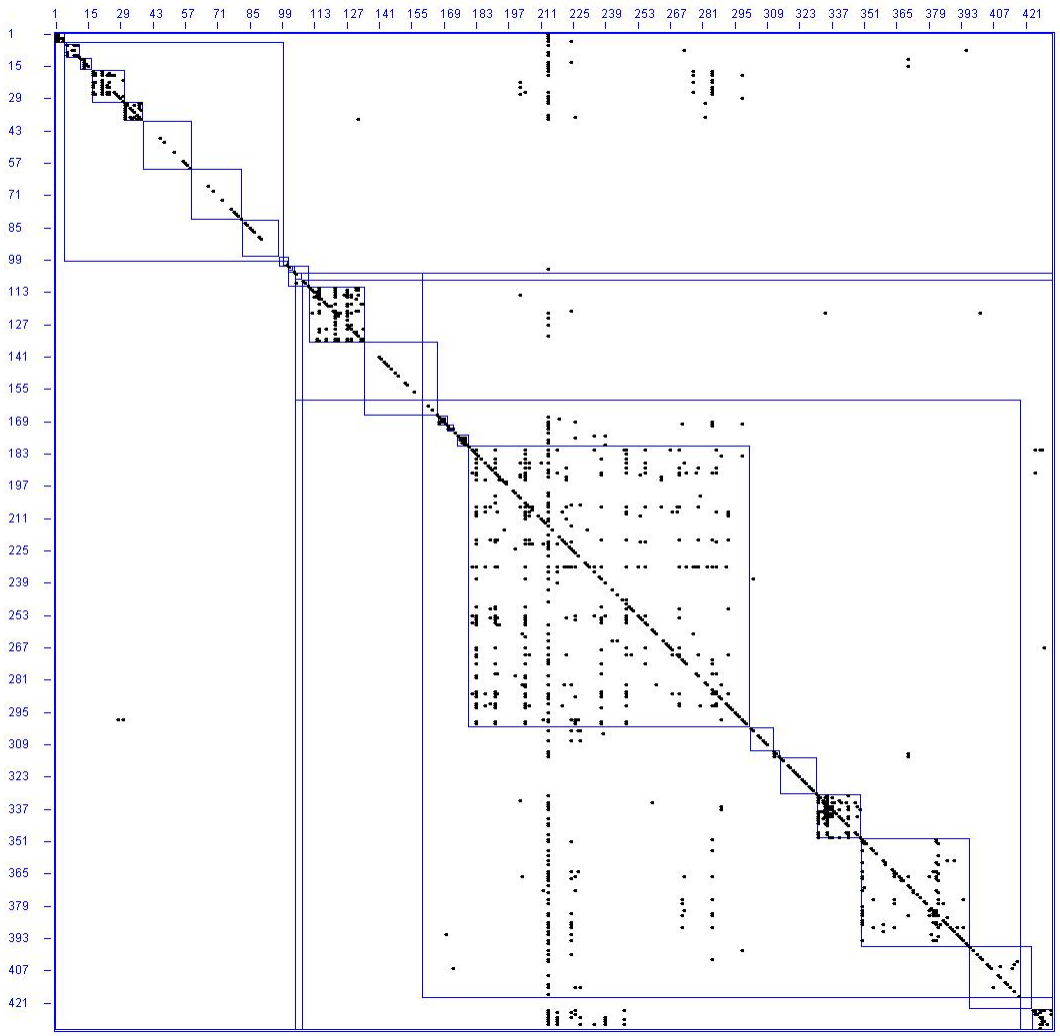


Abbildung D.17.: DSM Interbase (später Firebird)

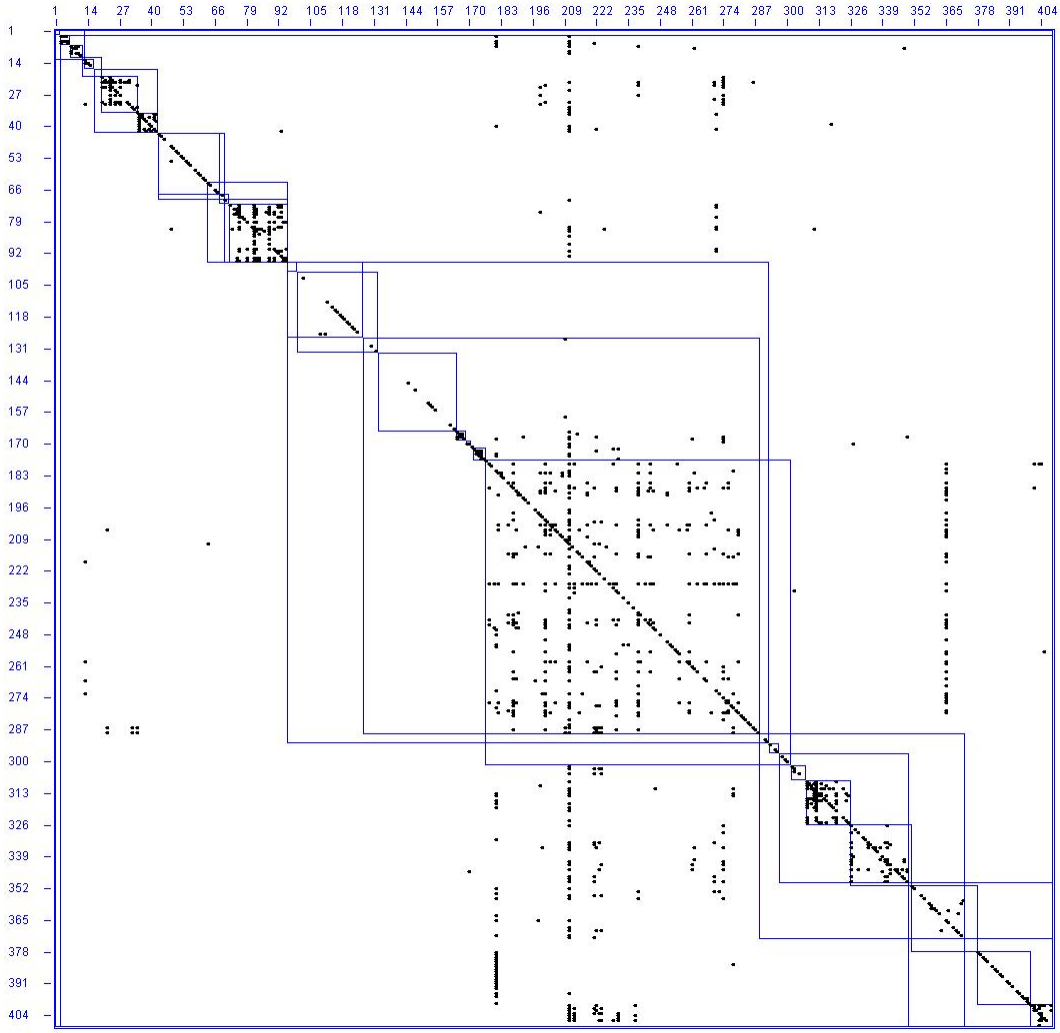


Abbildung D.18.: DSM Firebird Version 1.5.5.4926

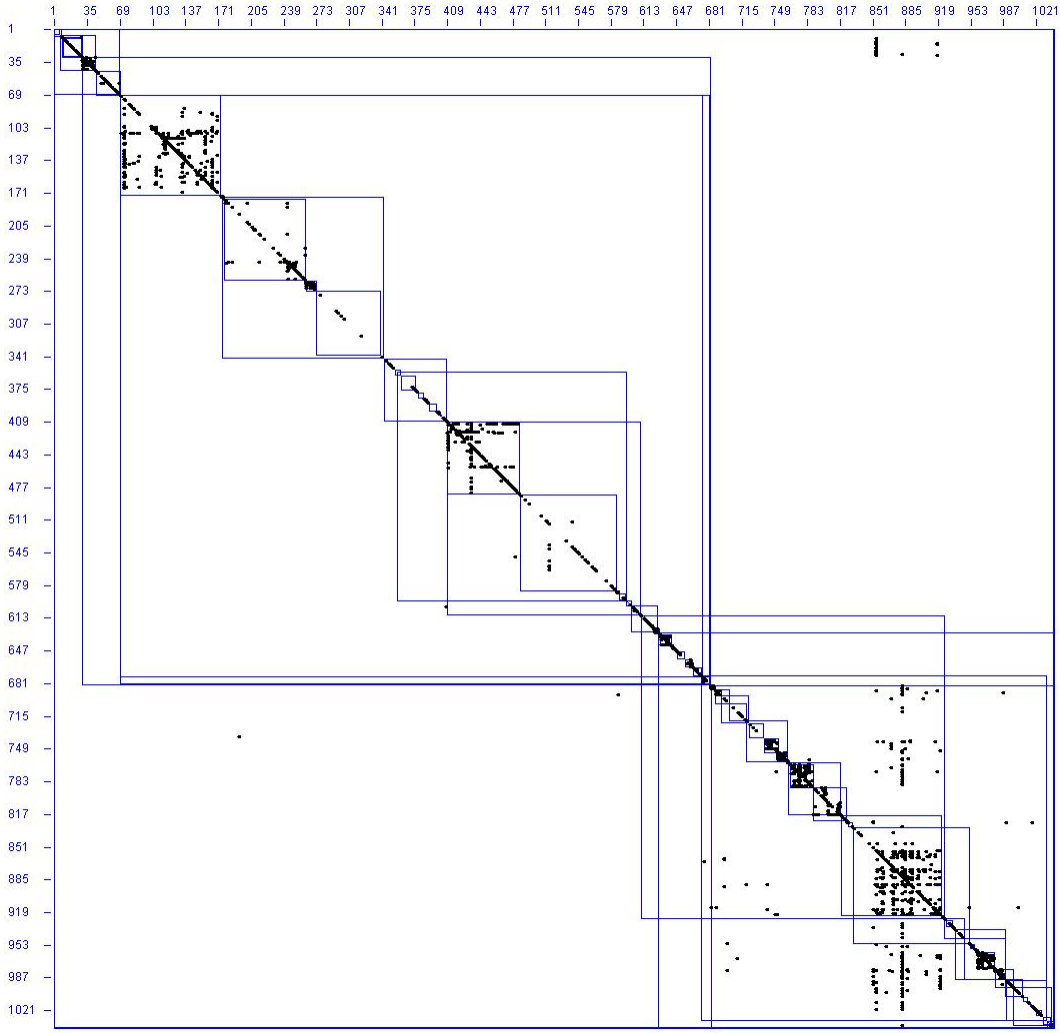


Abbildung D.19.: DSM Firebird Version 2.5.0.23247-Beta1

D.1.5. Dantensicherung

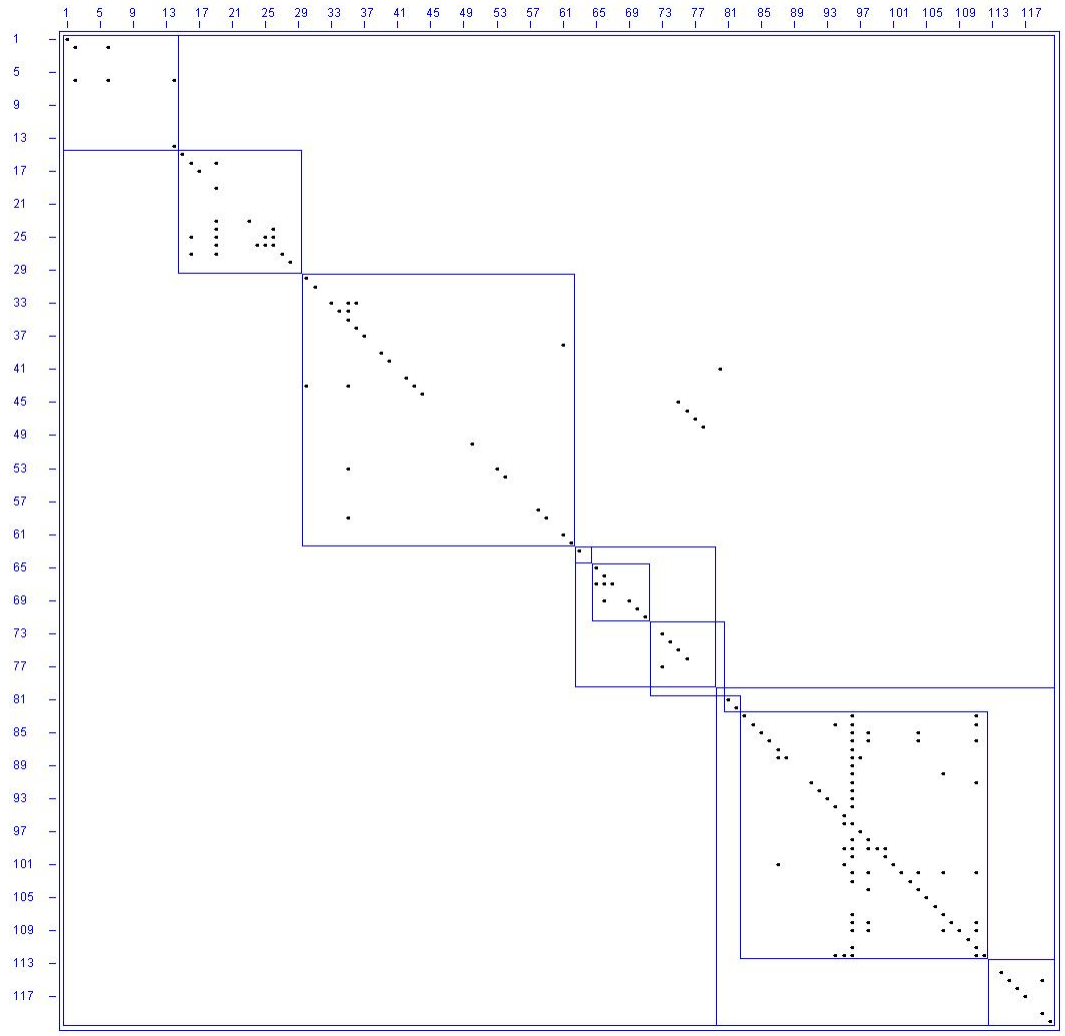


Abbildung D.20.: DSM Amanda Version 2.4.3b2

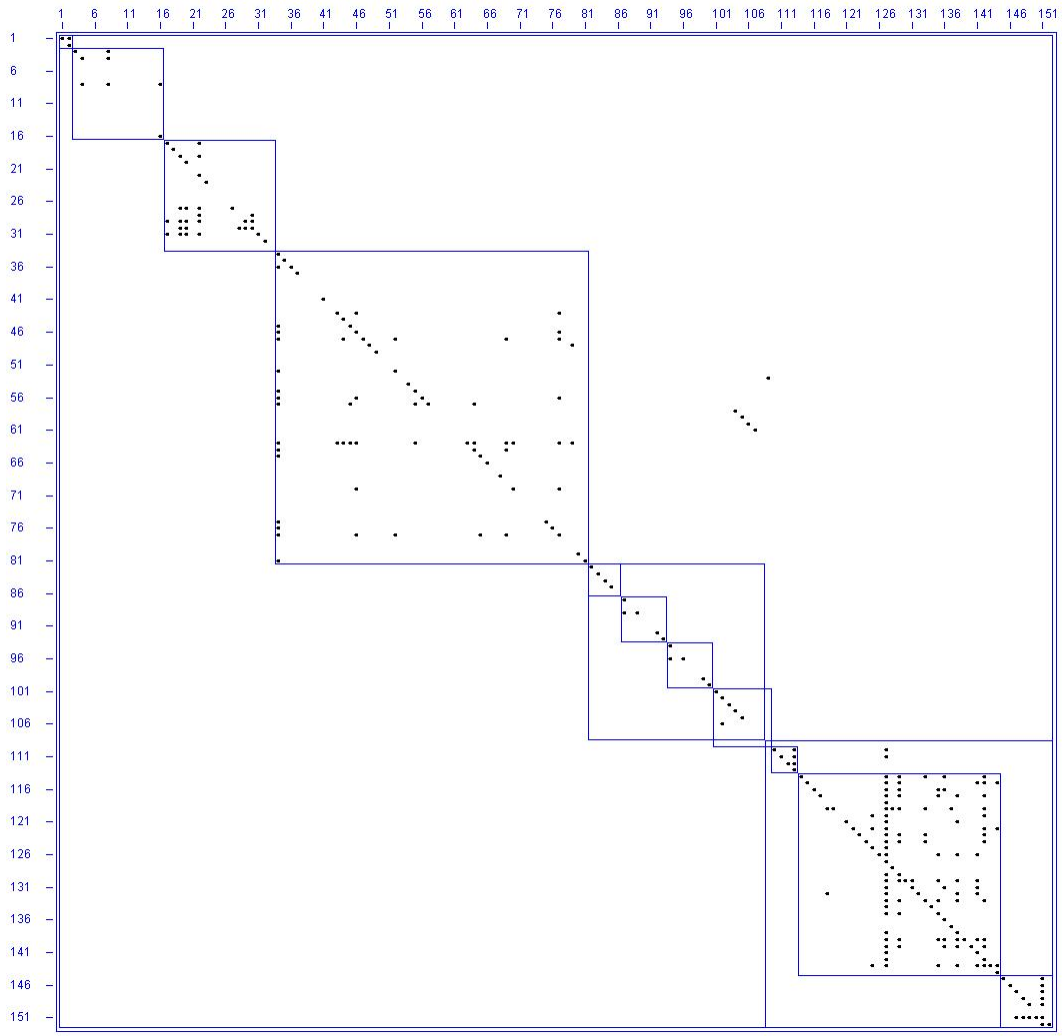


Abbildung D.21.: DSM Amanda Version 2.5.1p3

D.1.6. Financial

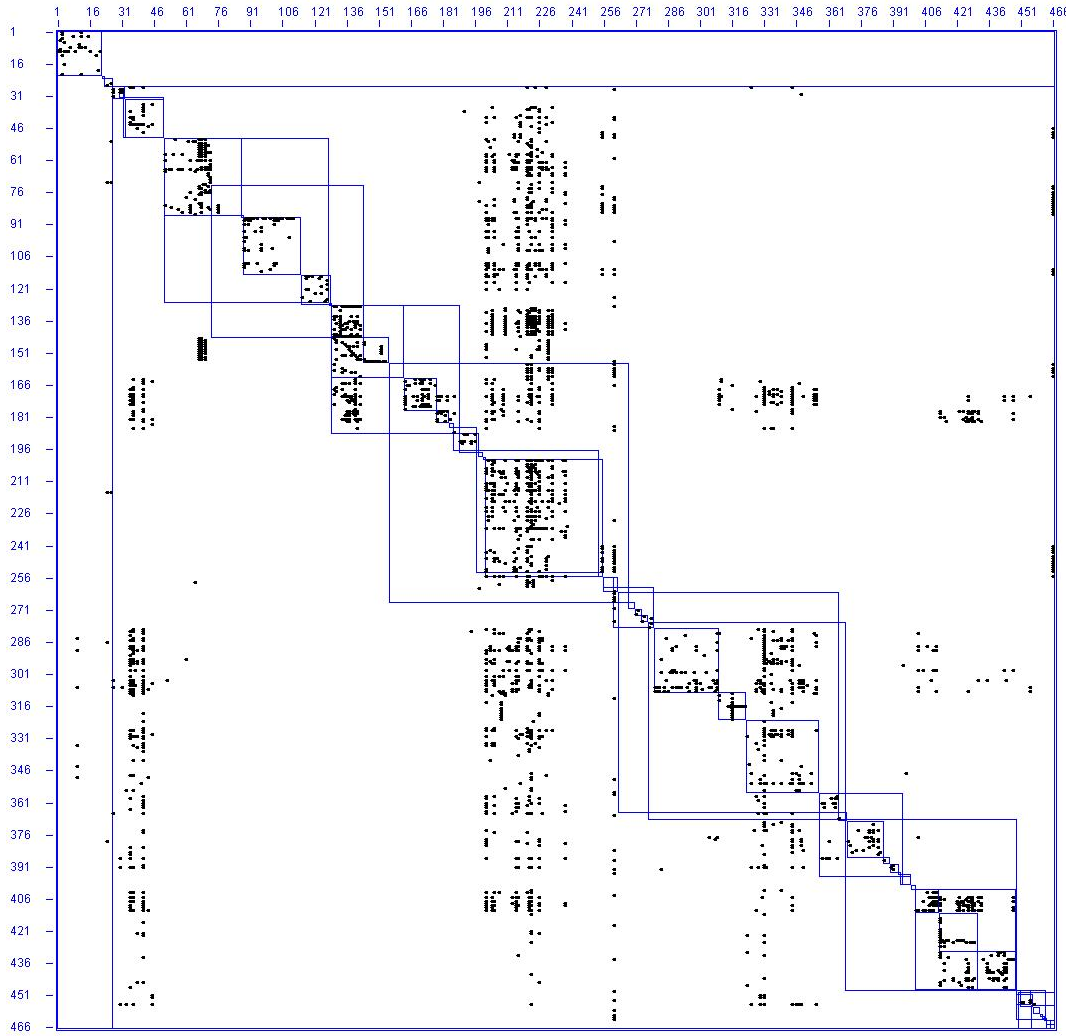


Abbildung D.22.: DSM Gnucash Version 1.8.4

D.1.7. Linux

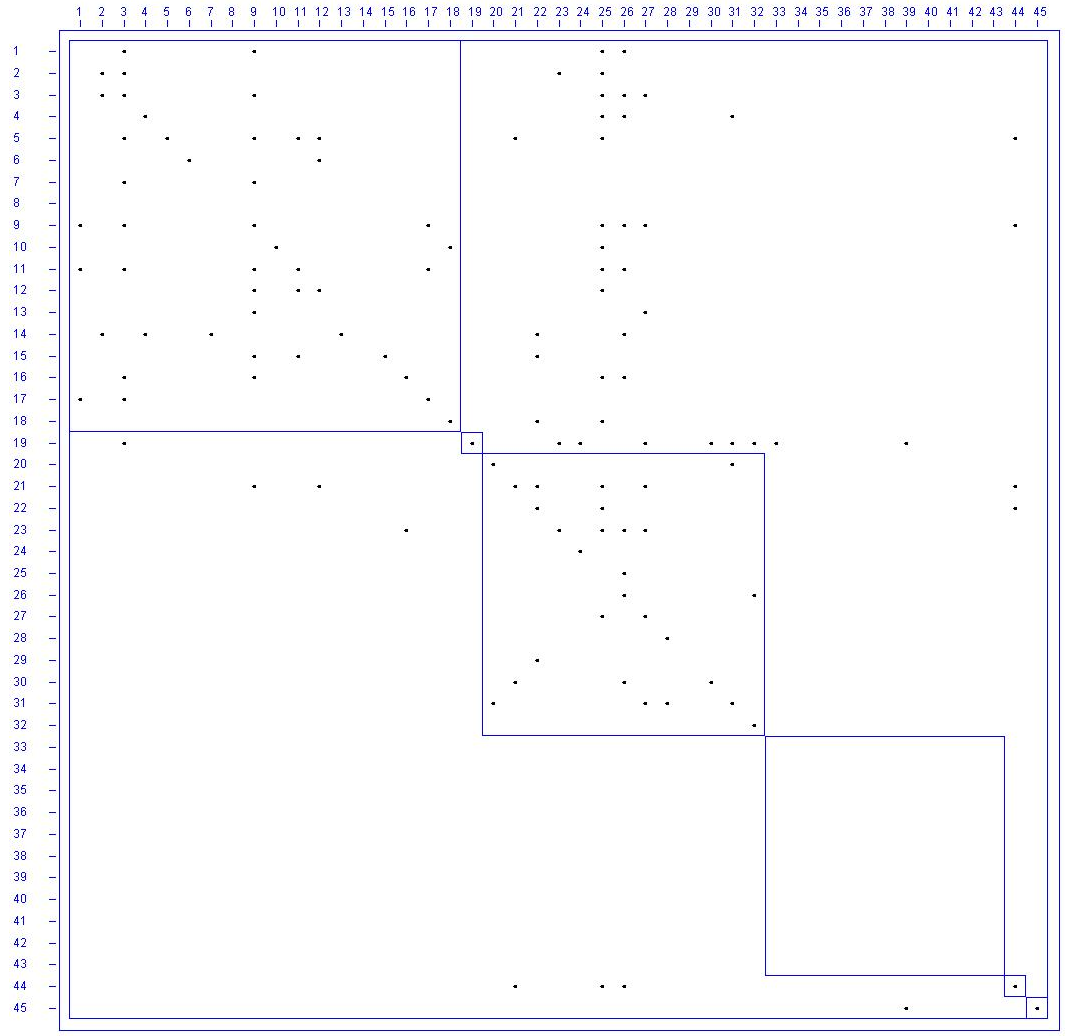


Abbildung D.23.: DSM Linux Version 0.01

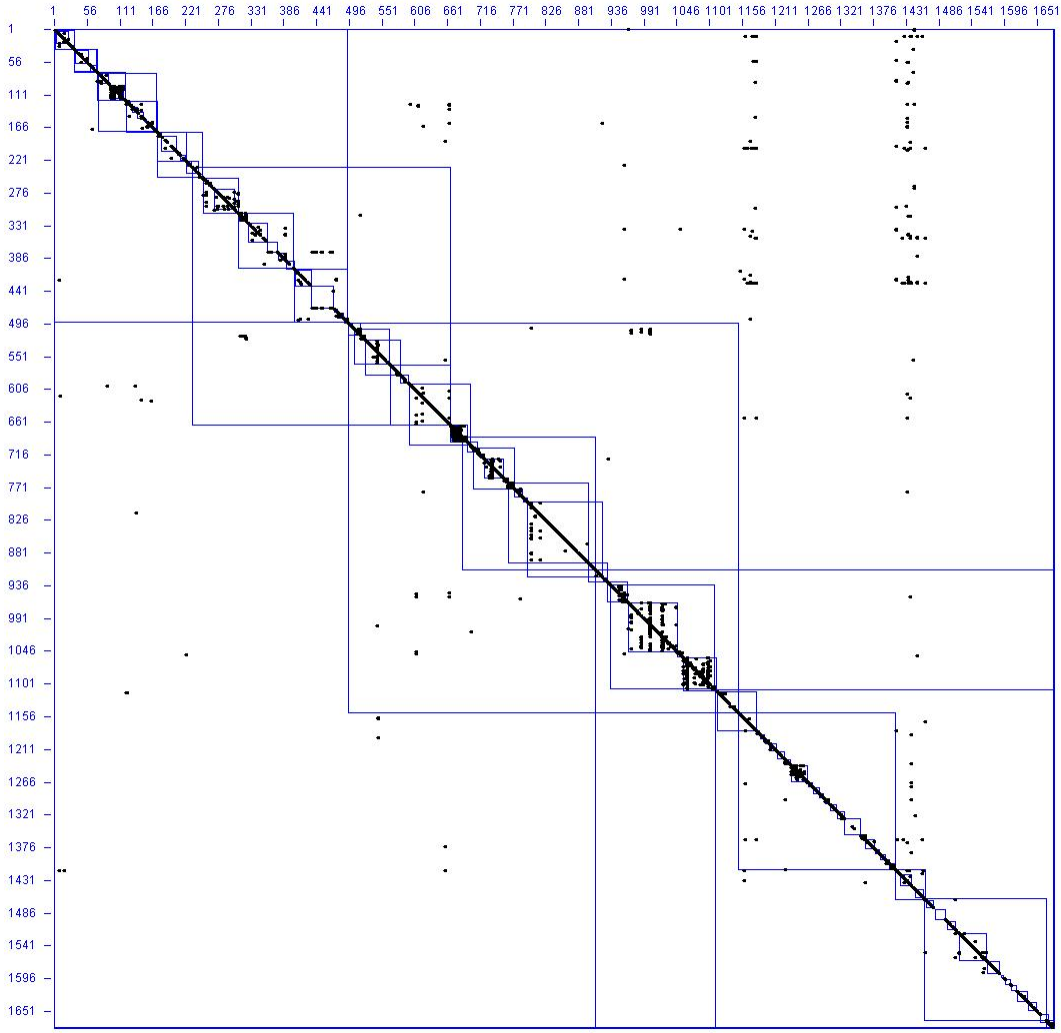


Abbildung D.24.: DSM Linux Version 2.1.105

D.2. Project Overview

Dieses Kapitel zeigt die Project Overview der einzelnen Programme und der jeweiligen Versionen. Folgende Angaben sind der Abbildungsbezeichnung zu entnehmen:

- Name der Project Overview
- Programmnamen
- Version

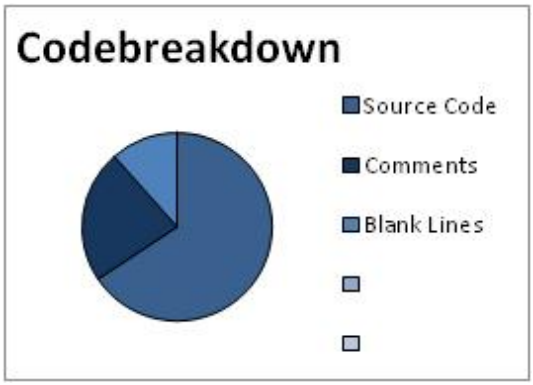


Abbildung D.25.: Code Breakdown JBoss-Portal Version 2.0-beta1

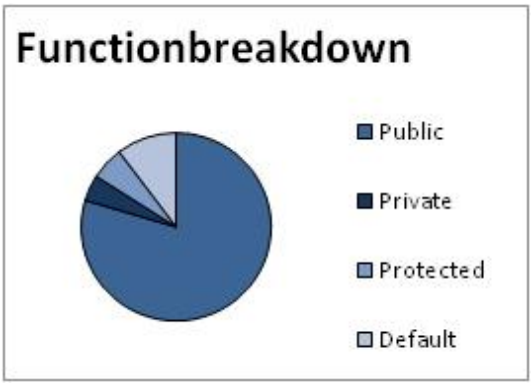


Abbildung D.26.: Function Breakdown JBoss-Portal Version 2.0-beta1

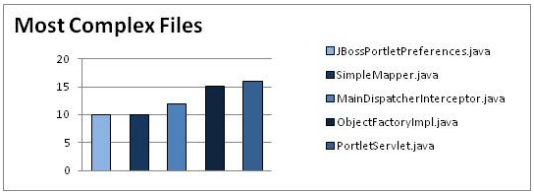


Abbildung D.27.: Most Complex Files JBoss-Portal Version 2.0-beta1

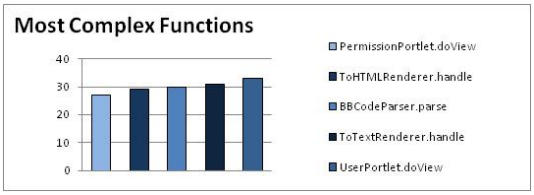


Abbildung D.28.: Most Complex Functions JBoss-Portal Version 2.0-beta1

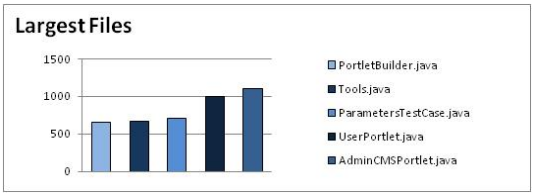


Abbildung D.29.: Largest Files JBoss-Portal Version 2.0-beta1

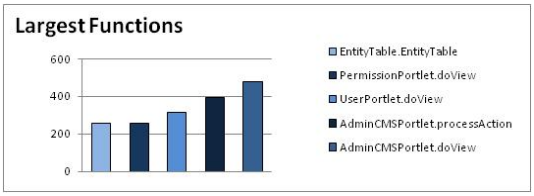


Abbildung D.30.: Largest Functions JBoss-Portal Version 2.0-beta1

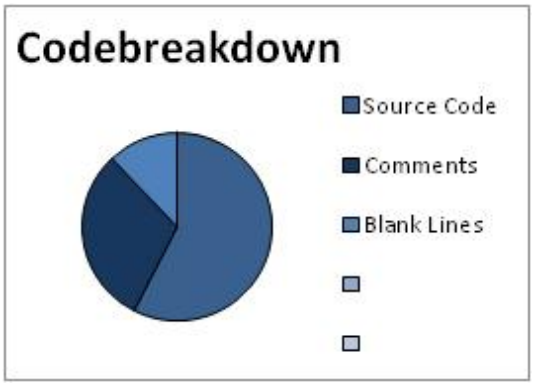


Abbildung D.31.: Code Breakdown JBoss-Portal Version 2.6.2.GA

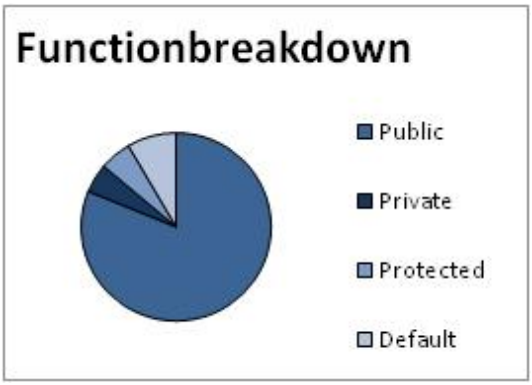


Abbildung D.32.: Function Breakdown JBoss-Portal Version 2.6.2.GA

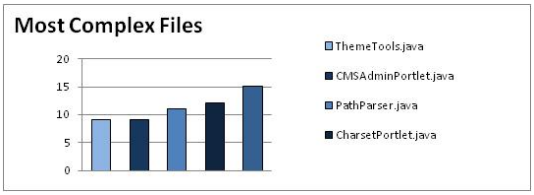


Abbildung D.33.: Most Complex Files JBoss-Portal Version 2.6.2.GA

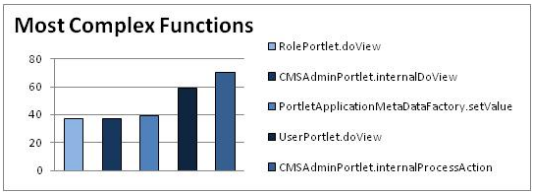


Abbildung D.34.: Most Complex Functions JBoss-Portal Version 2.6.2.GA

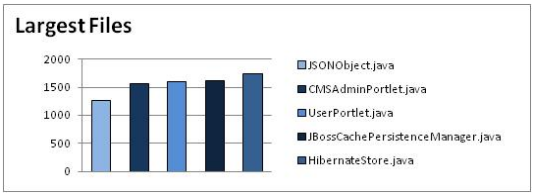


Abbildung D.35.: Largest Files JBoss-Portal Version 2.6.2.GA

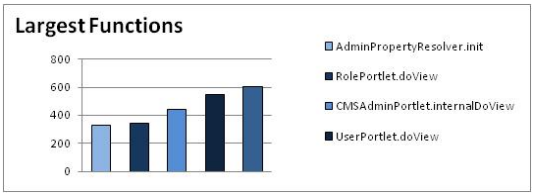


Abbildung D.36.: Largest Functions JBoss-Portal Version 2.6.2.GA

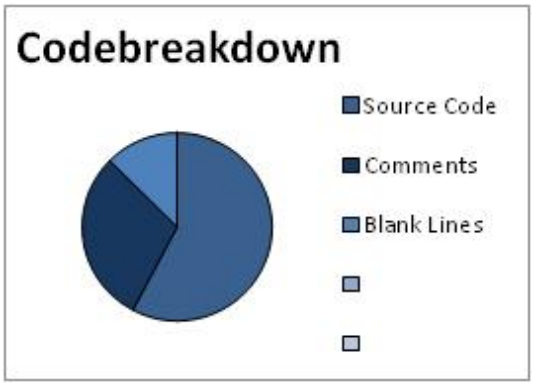


Abbildung D.37.: Code Breakdown JBoss-Portal Version 2.7.0.CR1

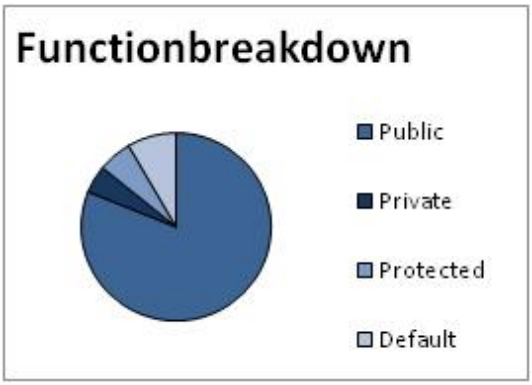


Abbildung D.38.: Function Breakdown JBoss-Portal Version 2.7.0.CR1

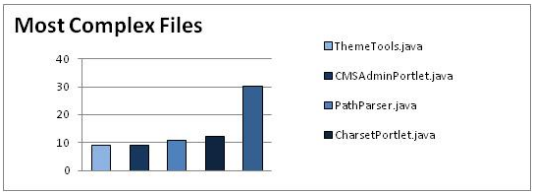


Abbildung D.39.: Most Complex Files JBoss-Portal Version 2.7.0.CR1

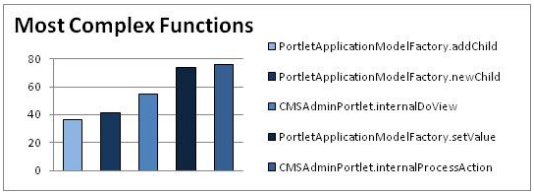


Abbildung D.40.: Most Complex Functions JBoss-Portal Version 2.7.0.CR1

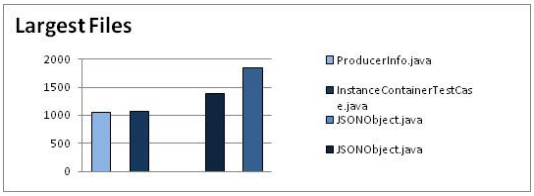


Abbildung D.41.: Largest Files JBoss-Portal Version 2.7.0.CR1

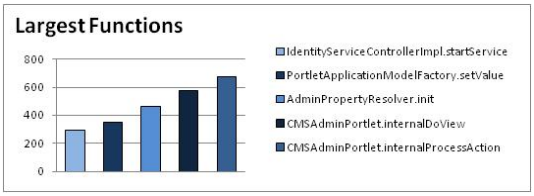


Abbildung D.42.: Largest Functions JBoss-Portal Version 2.7.0.CR1

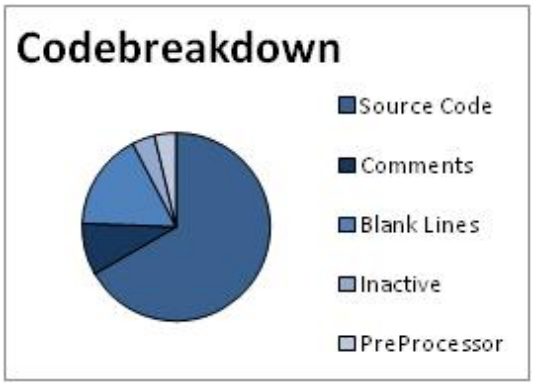


Abbildung D.43.: Code Breakdown Epiphany Version 2.22.3

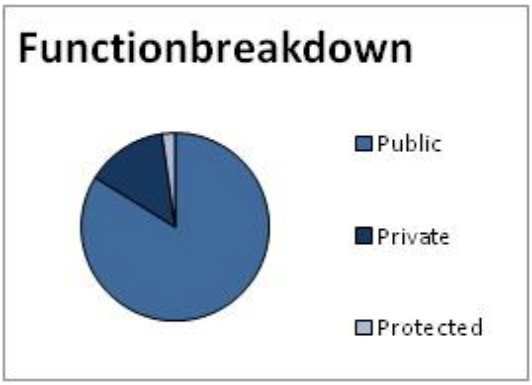


Abbildung D.44.: Function Breakdown Epiphany Version 2.22.3

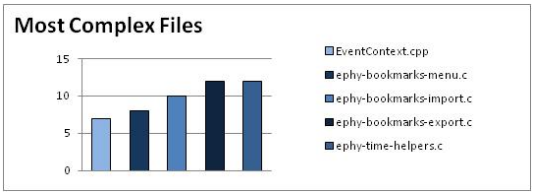


Abbildung D.45.: Most Complex Files Epiphany Version 2.22.3

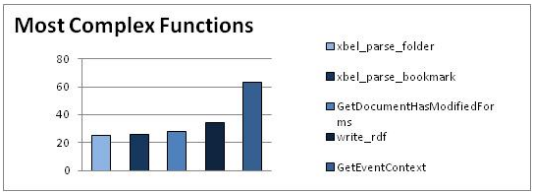


Abbildung D.46.: Most Complex Functions Epiphany Version 2.22.3

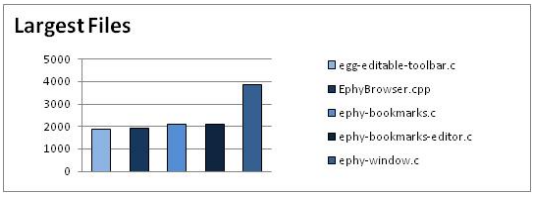


Abbildung D.47.: Largest Files Epiphany Version 2.22.3

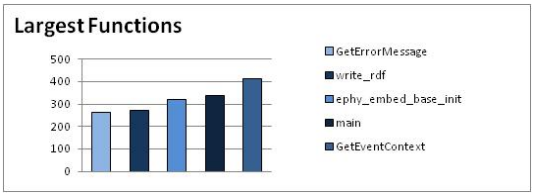


Abbildung D.48.: Largest Functions Epiphany Version 2.22.3

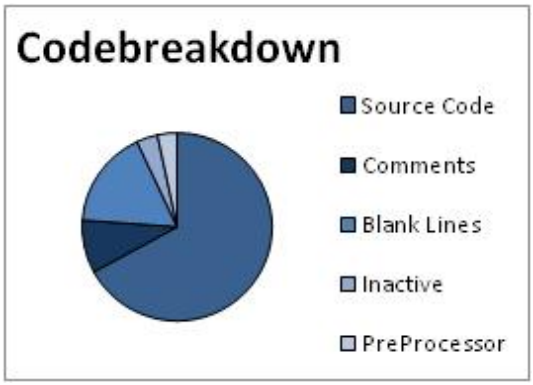


Abbildung D.49.: Code Breakdown Epiphany Version 2.26.1

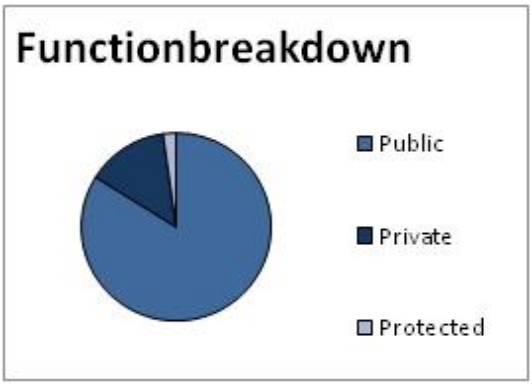


Abbildung D.50.: Function Breakdown Epiphany Version 2.26.1

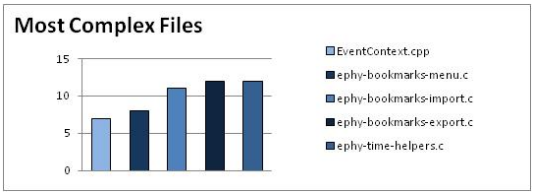


Abbildung D.51.: Most Complex Files Epiphany Version 2.26.1

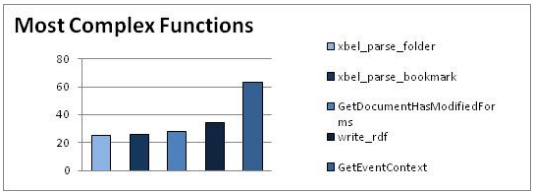


Abbildung D.52.: Most Complex Functions Epiphany Version 2.26.1

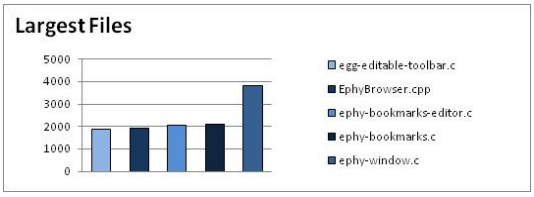


Abbildung D.53.: Largest Files Epiphany Version 2.26.1

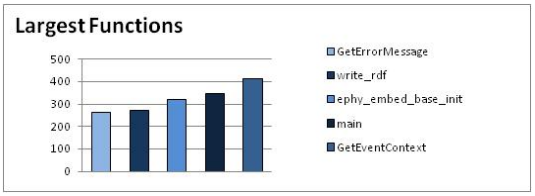


Abbildung D.54.: Largest Functions Epiphany Version 2.26.1

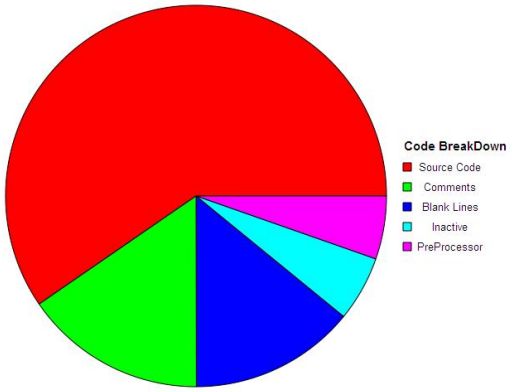


Abbildung D.55.: Code Breakdown Firefox Version 1.0

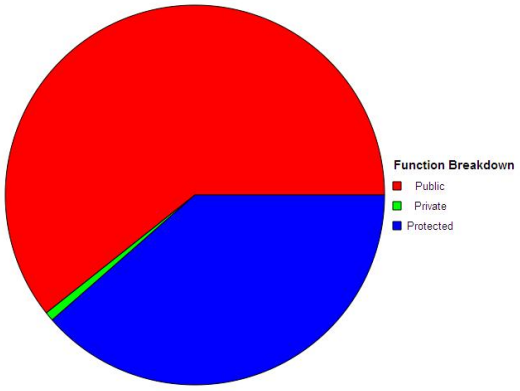


Abbildung D.56.: Function Breakdown Firefox Version 1.0

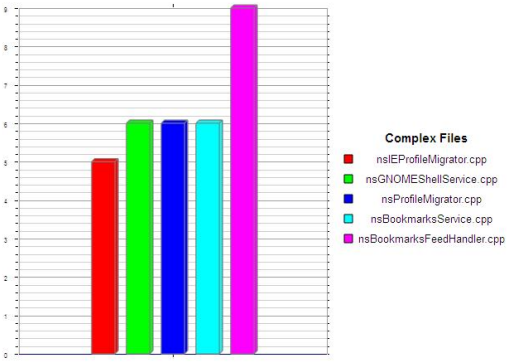


Abbildung D.57.: Most Complex Files Firefox Version 1.0

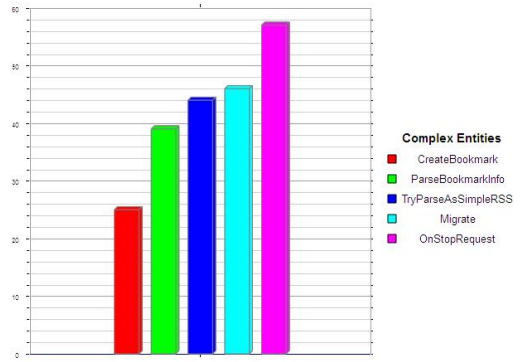


Abbildung D.58.: Most Complex Functions Firefox Version 1.0

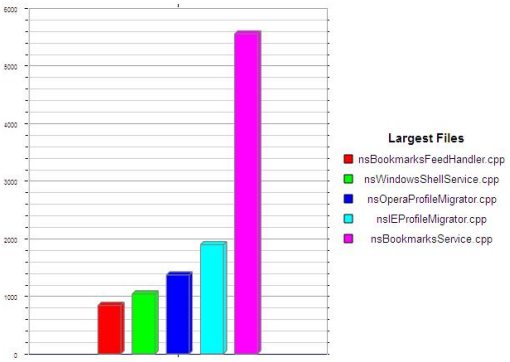


Abbildung D.59.: Largest Files Firefox Version 1.0

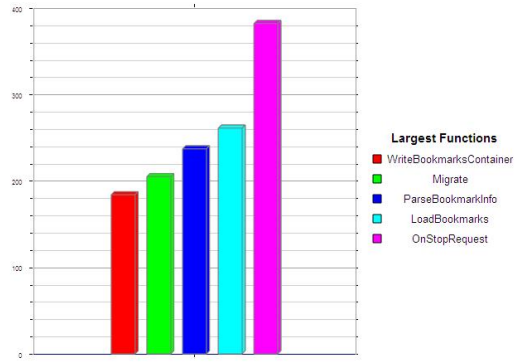


Abbildung D.60.: Largest Functions Firefox Version 1.0

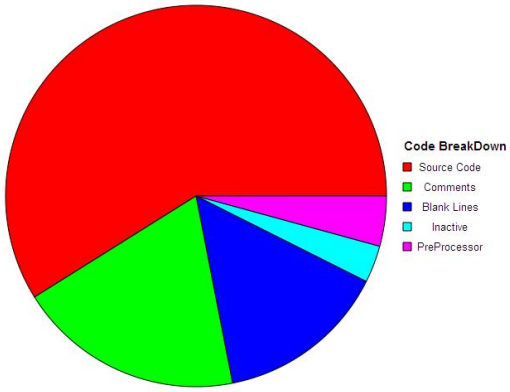


Abbildung D.61.: Code Breakdown Firefox Version 2.0

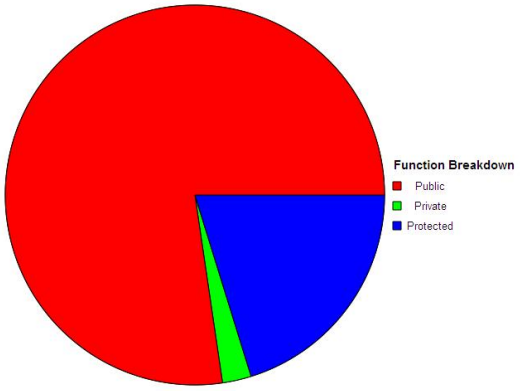


Abbildung D.62.: Function Breakdown Firefox Version 2.0

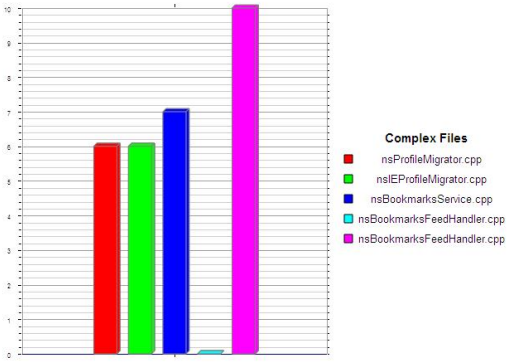


Abbildung D.63.: Most Complex Files Firefox Version 2.0

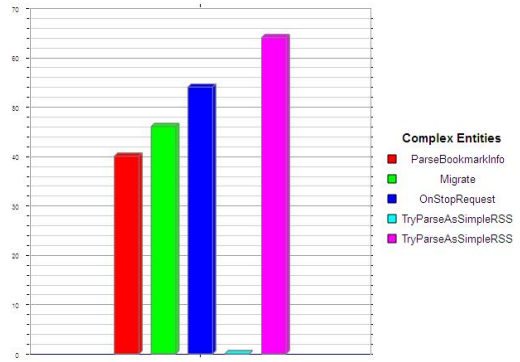


Abbildung D.64.: Most Complex Functions Firefox Version 2.0

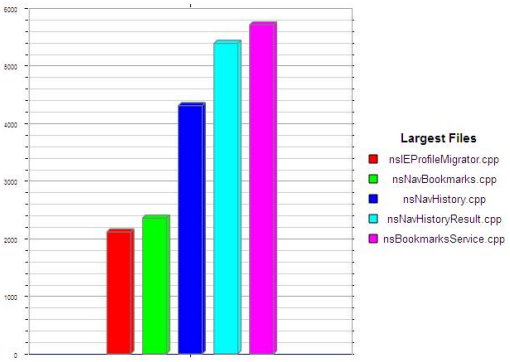


Abbildung D.65.: Largest Files Firefox Version 2.0

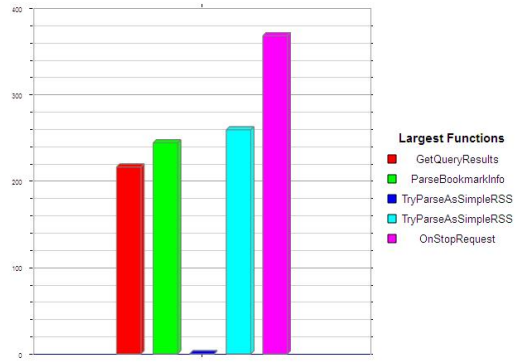


Abbildung D.66.: Largest Functions Firefox Version 2.0

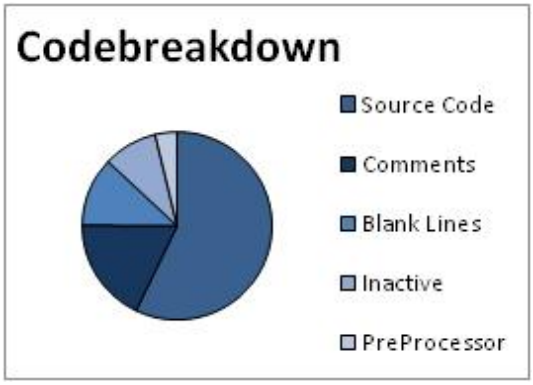


Abbildung D.67.: Code Breakdown Firefox Version 3.5

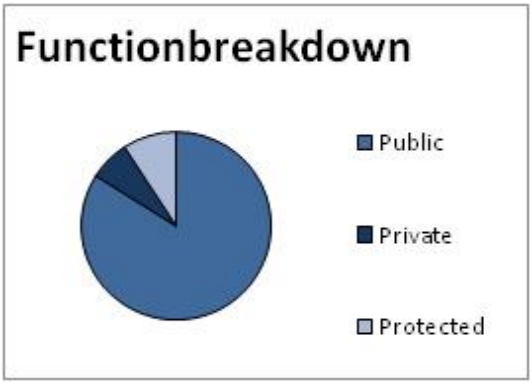


Abbildung D.68.: Function Breakdown Firefox Version 3.5

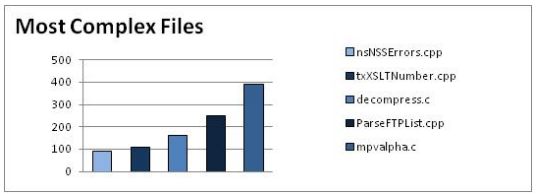


Abbildung D.69.: Most Complex Files Firefox Version 3.5

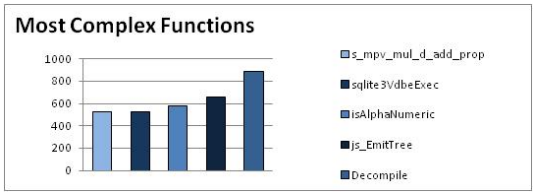


Abbildung D.70.: Most Complex Functions Firefox Version 3.5

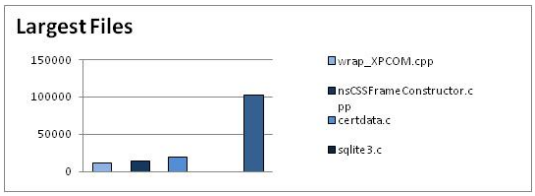


Abbildung D.71.: Largest Files Firefox Version 3.5

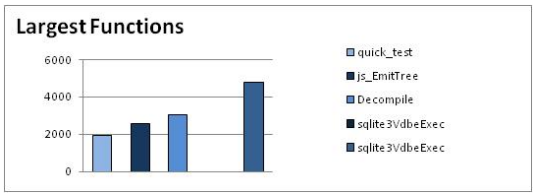


Abbildung D.72.: Largest Functions Firefox Version 3.5

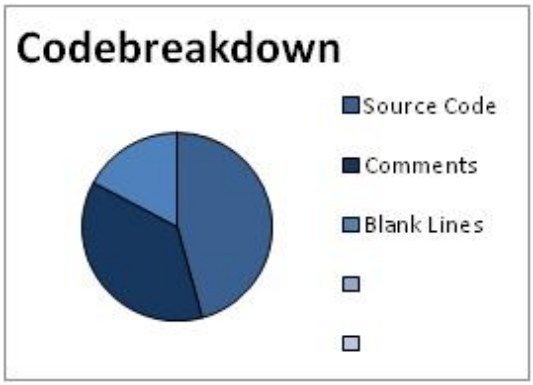


Abbildung D.73.: Code Breakdown Hipergate Version 1.0.4-beta

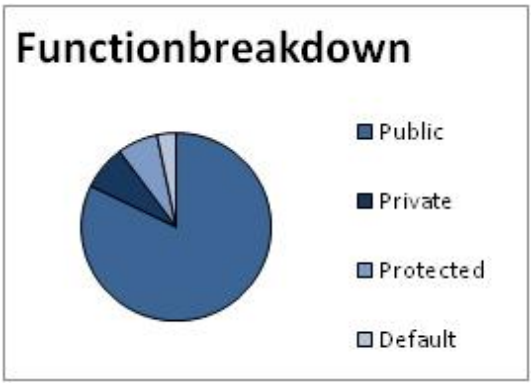


Abbildung D.74.: Function Breakdown Hipergate Version 1.0.4-beta

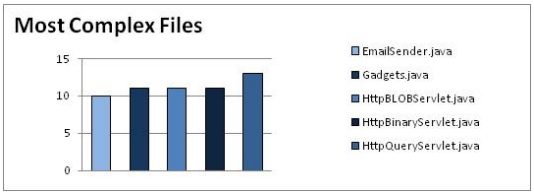


Abbildung D.75.: Most Complex Files Hipergate Version 1.0.4-beta

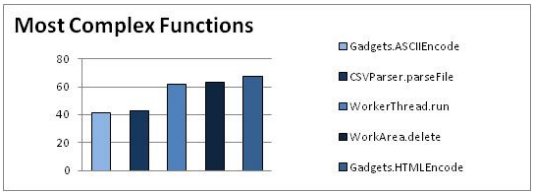


Abbildung D.76.: Most Complex Functions Hipergate Version 1.0.4-beta

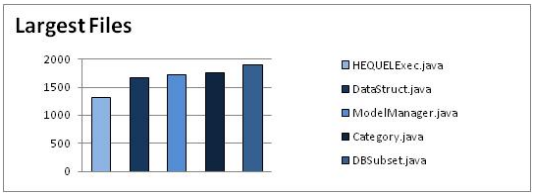


Abbildung D.77.: Largest Files Hipergate Version 1.0.4-beta

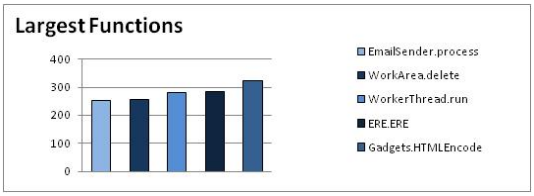


Abbildung D.78.: Largest Functions Hipergate Version 1.0.4-beta

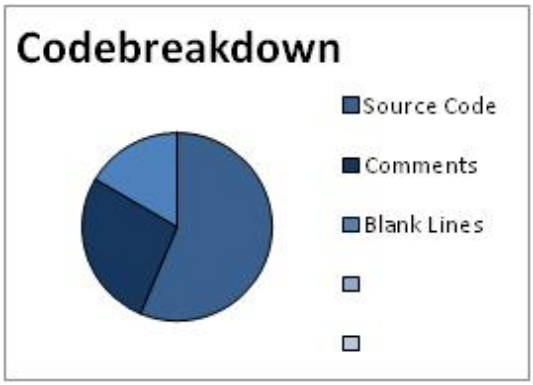


Abbildung D.79.: Code Breakdown Hipergate Version 3.0

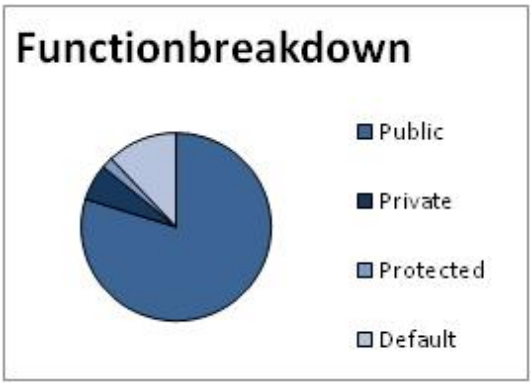


Abbildung D.80.: Function Breakdown Hipergate Version 3.0

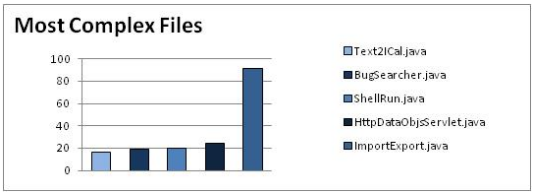


Abbildung D.81.: Most Complex Files Hipergate Version 3.0

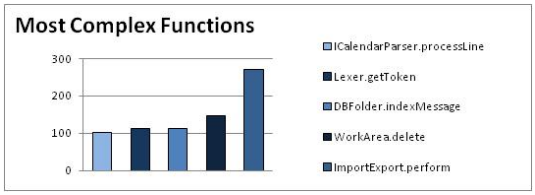


Abbildung D.82.: Most Complex Functions Hipergate Version 3.0

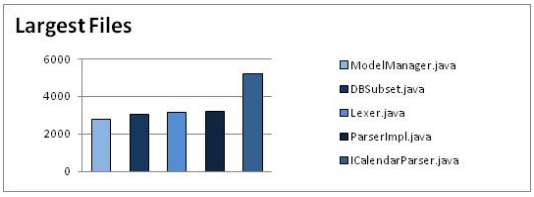


Abbildung D.83.: Largest Files Hipergate Version 3.0

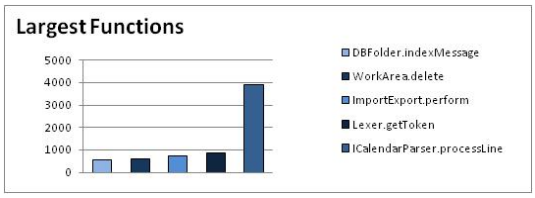


Abbildung D.84.: Largest Functions Hipergate Version 3.0

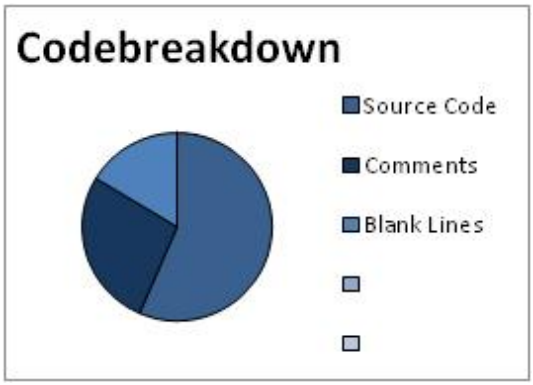


Abbildung D.85.: Code Breakdown Hipergate Version 4.0

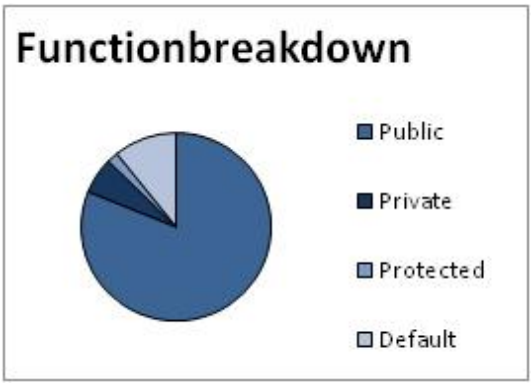


Abbildung D.86.: Function Breakdown Hipergate Version 4.0

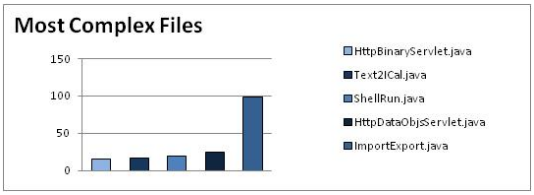


Abbildung D.87.: Most Complex Files Hipergate Version 4.0

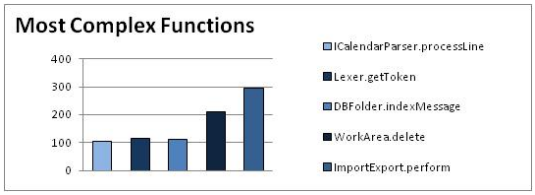


Abbildung D.88.: Most Complex Functions Hipergate Version 4.0

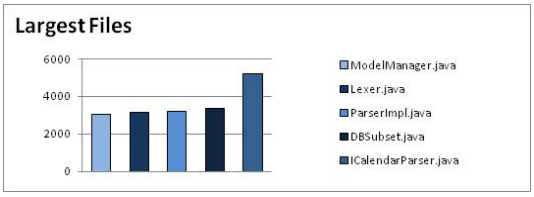


Abbildung D.89.: Largest Files Hipergate Version 4.0

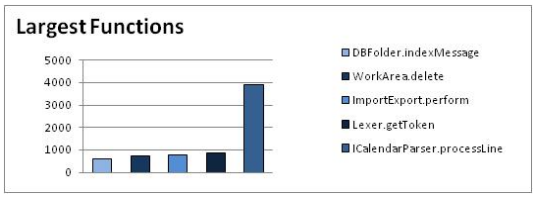


Abbildung D.90.: Largest Functions Hipergate Version 4.0

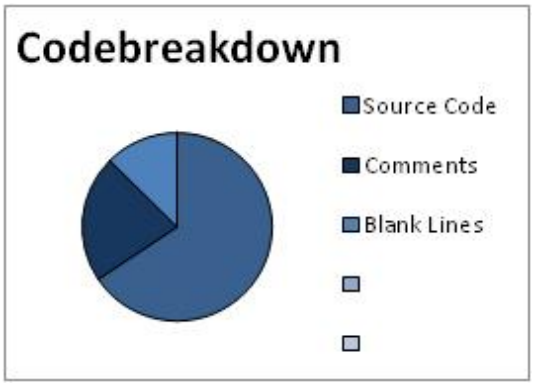


Abbildung D.91.: Code Breakdown Opentaps Version 0.8.5

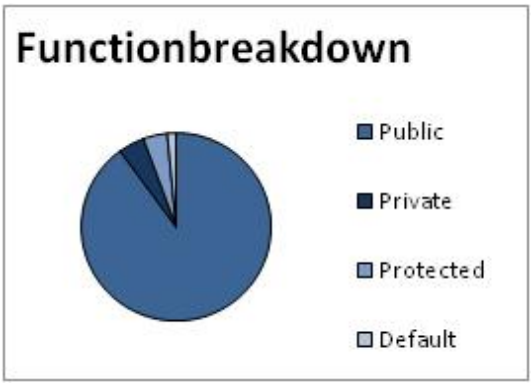


Abbildung D.92.: Function Breakdown Opentaps Version 0.8.5

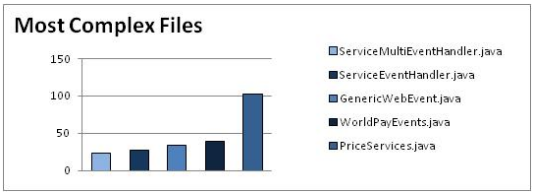


Abbildung D.93.: Most Complex Files Opentaps Version 0.8.5

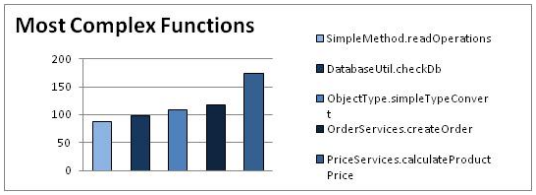


Abbildung D.94.: Most Complex Functions Opentaps Version 0.8.5

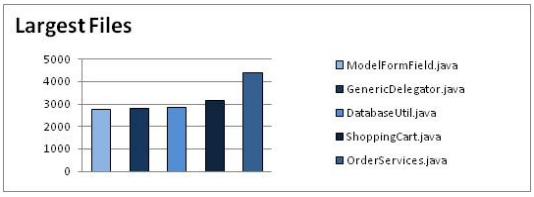


Abbildung D.95.: Largest Files Opentaps Version 0.8.5

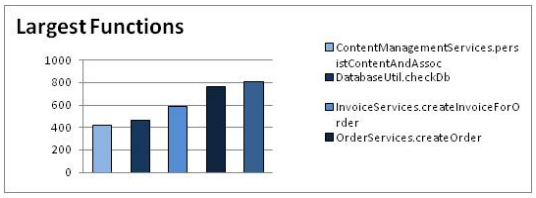


Abbildung D.96.: Largest Functions Opentaps Version 0.8.5

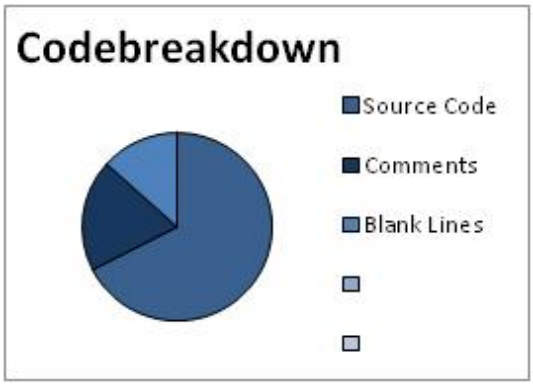


Abbildung D.97.: Code Breakdown Opentaps Version 1.0.4

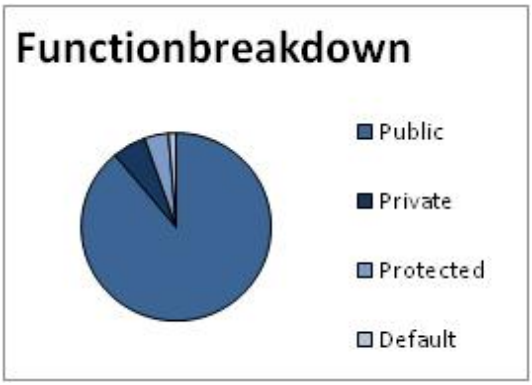


Abbildung D.98.: Function Breakdown Opentaps Version 1.0.4

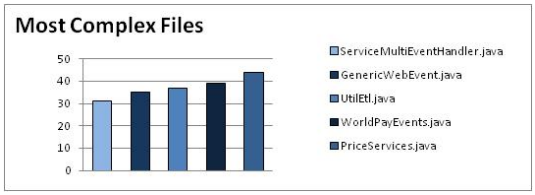


Abbildung D.99.: Most Complex Files Opentaps Version 1.0.4

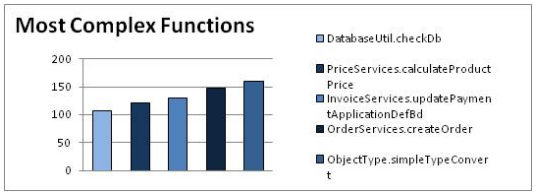


Abbildung D.100.: Most Complex Functions Opentaps Version 1.0.4

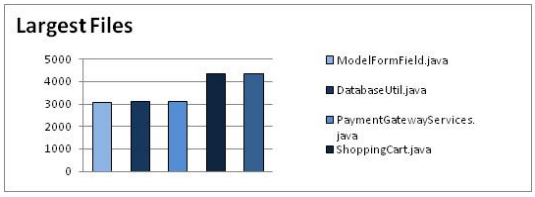


Abbildung D.101.: Largest Files Opentaps Version 1.0.4

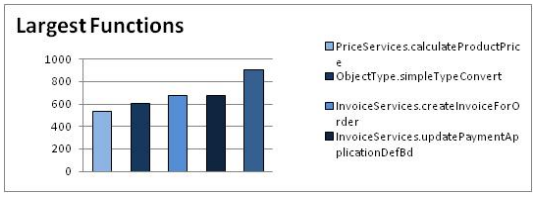


Abbildung D.102.: Largest Functions Opentaps Version 1.0.4

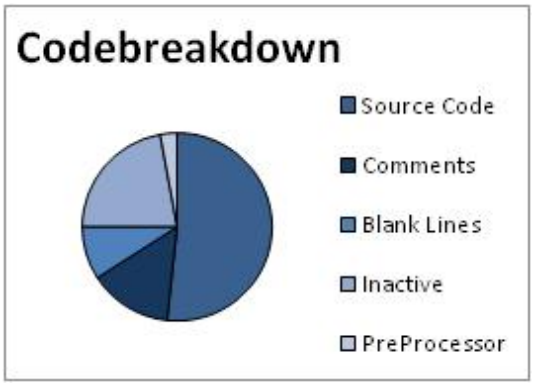


Abbildung D.103.: Code Breakdown MySQL Version 4.1.22

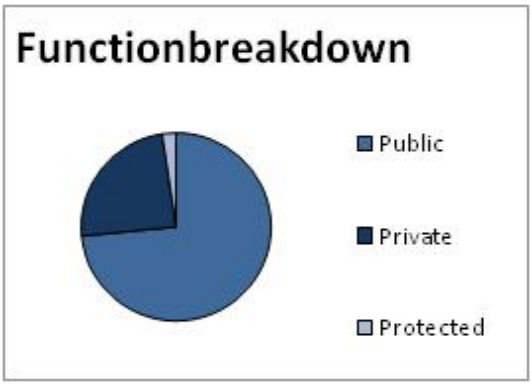


Abbildung D.104.: Function Breakdown MySQL Version 4.1.22

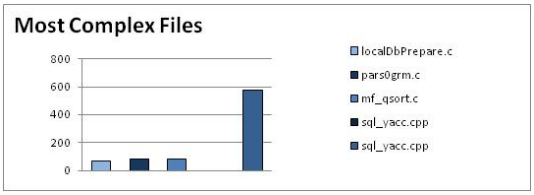


Abbildung D.105.: Most Complex Files MySQL Version 4.1.22

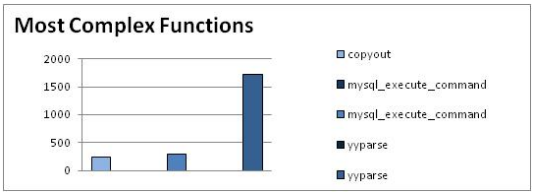


Abbildung D.106.: Most Complex Functions MySQL Version 4.1.22

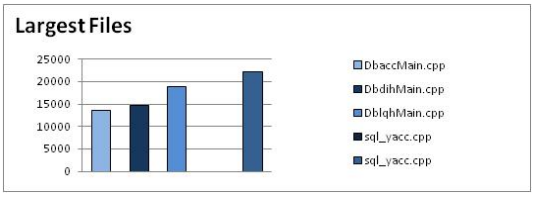


Abbildung D.107.: Largest Files MySQL Version 4.1.22

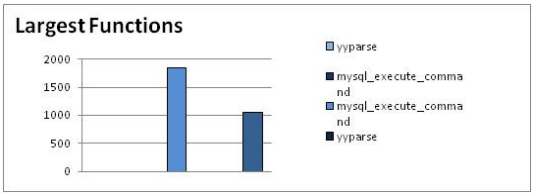


Abbildung D.108.: Largest Functions MySQL Version 4.1.22

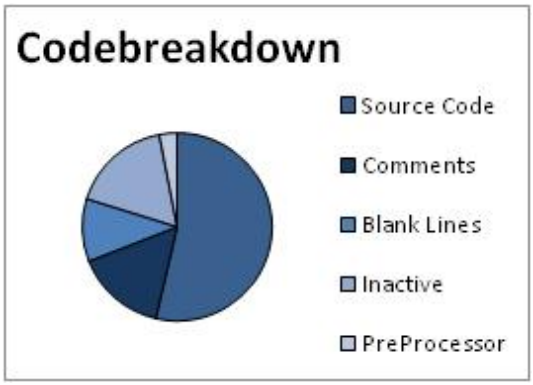


Abbildung D.109.: Code Breakdown MySQL Version 5.0.83

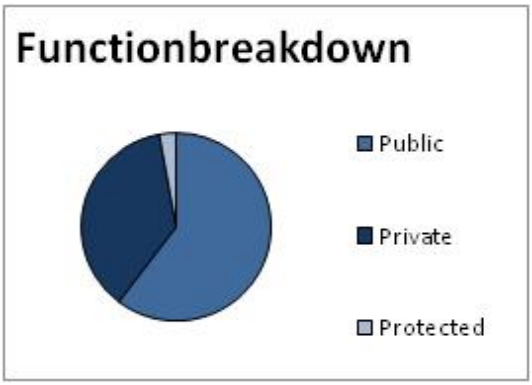


Abbildung D.110.: Function Breakdown MySQL Version 5.0.83

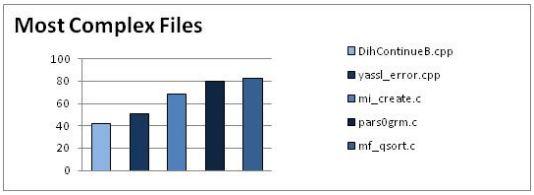


Abbildung D.111.: Most Complex Files MySQL Version 5.0.83

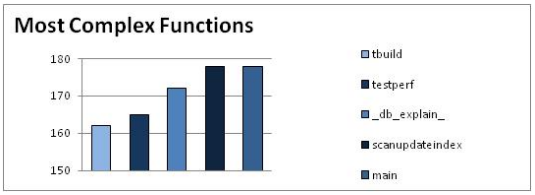


Abbildung D.112.: Most Complex Functions MySQL Version 5.0.83

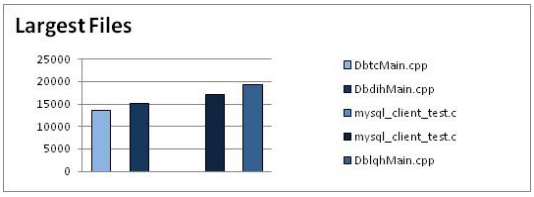


Abbildung D.113.: Largest Files MySQL Version 5.0.83

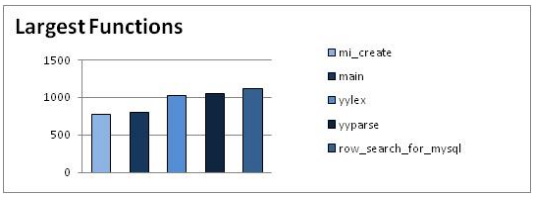


Abbildung D.114.: Largest Functions MySQL Version 5.0.83

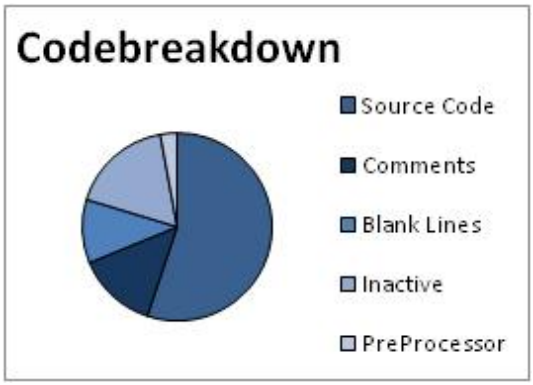


Abbildung D.115.: Code Breakdown MySQL Version 5.4.1-beta

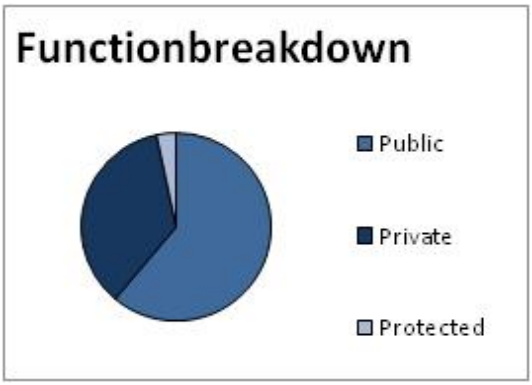


Abbildung D.116.: Function Breakdown MySQL Version 5.4.1-beta

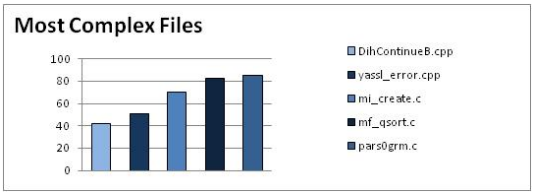


Abbildung D.117.: Most Complex Files MySQL Version 5.4.1-beta

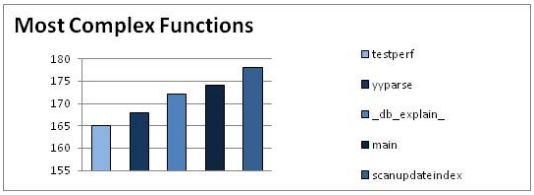


Abbildung D.118.: Most Complex Functions MySQL Version 5.4.1-beta

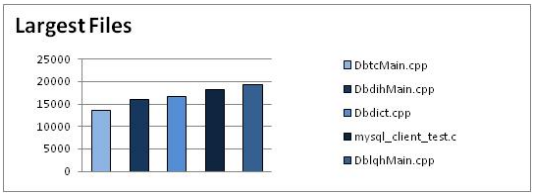


Abbildung D.119.: Largest Files MySQL Version 5.4.1-beta

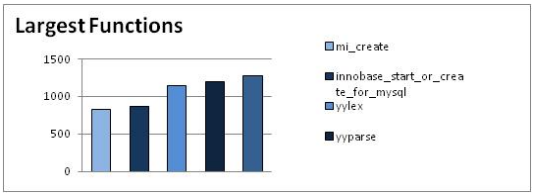


Abbildung D.120.: Largest Functions MySQL Version 5.4.1-beta

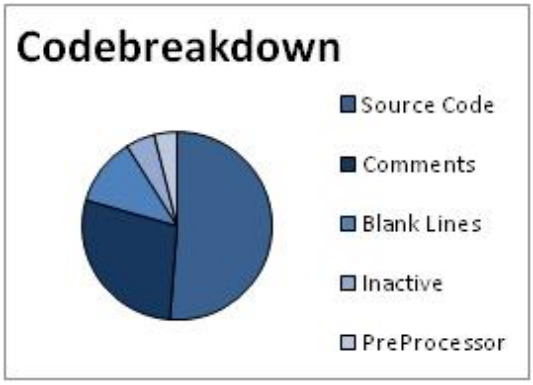


Abbildung D.121.: Code Breakdown Postgresql Version v60

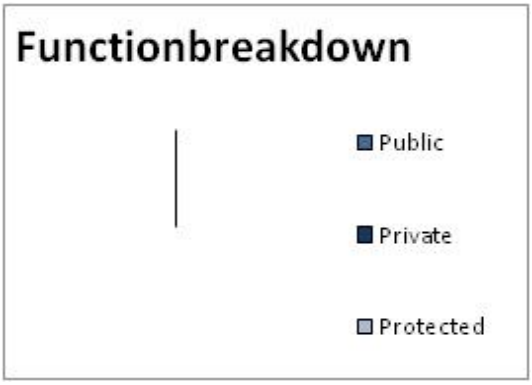


Abbildung D.122.: Function Breakdown Postgresql Version v60

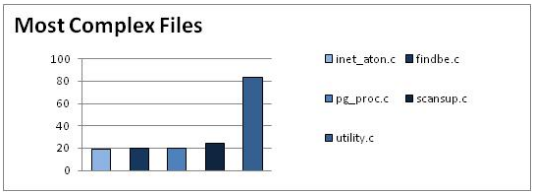


Abbildung D.123.: Most Complex Files Postgresql Version v60

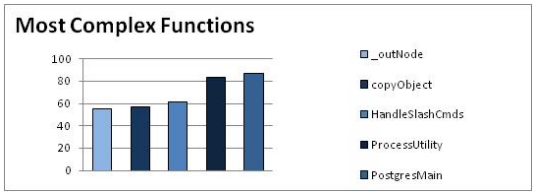


Abbildung D.124.: Most Complex Functions Postgresql Version v60

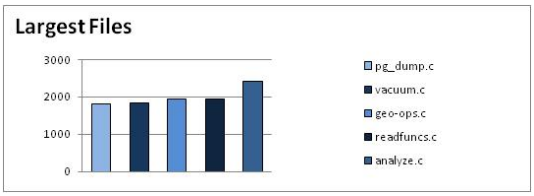


Abbildung D.125.: Largest Files Postgresql Version v60

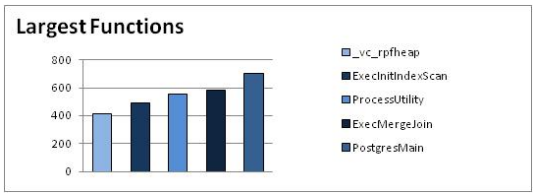


Abbildung D.126.: Largest Functions Postgresql Version v60

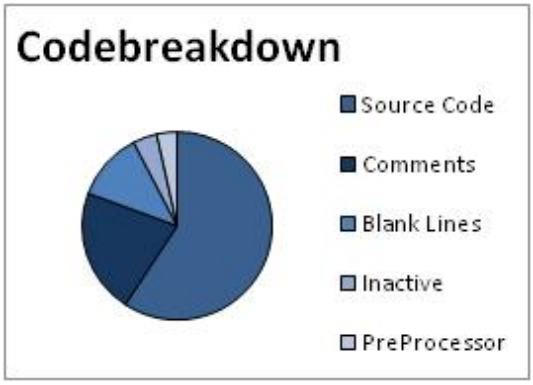


Abbildung D.127.: Code Breakdown Postgresql Version 8.0.19

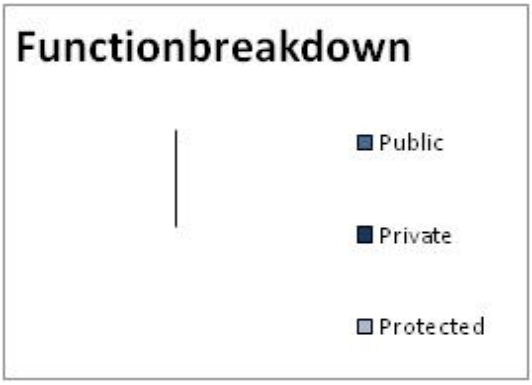


Abbildung D.128.: Function Breakdown Postgresql Version 8.0.19

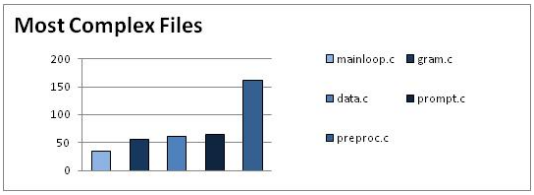


Abbildung D.129.: Most Complex Files Postgresql Version 8.0.19

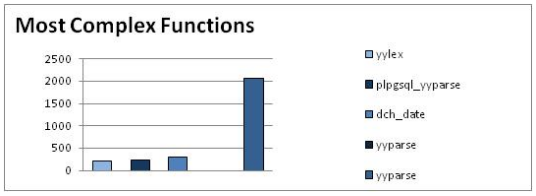


Abbildung D.130.: Most Complex Functions Postgresql Version 8.0.19

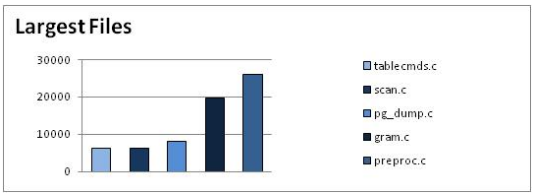


Abbildung D.131.: Largest Files Postgresql Version 8.0.19

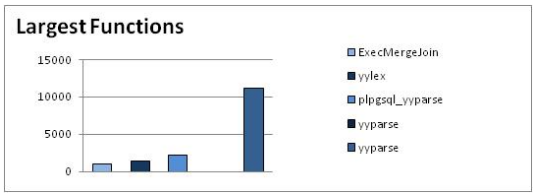


Abbildung D.132.: Largest Functions Postgresql Version 8.0.19

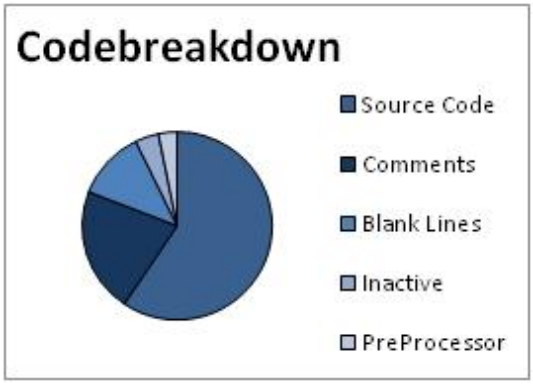


Abbildung D.133.: Code Breakdown Postgresql Version 8.5alpha1

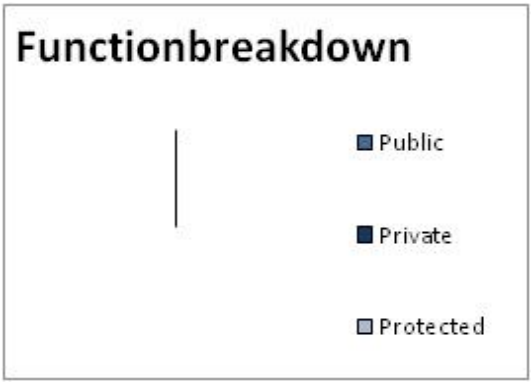


Abbildung D.134.: Function Breakdown Postgresql Version 8.5alpha1

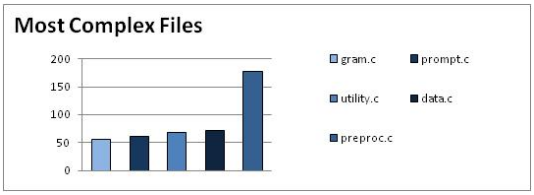


Abbildung D.135.: Most Complex Files Postgresql Version 8.5alpha1

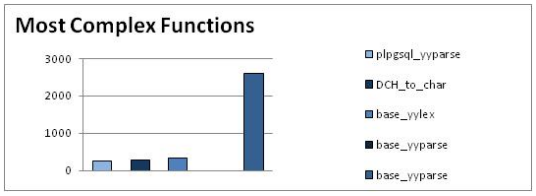


Abbildung D.136.: Most Complex Functions Postgresql Version 8.5alpha1

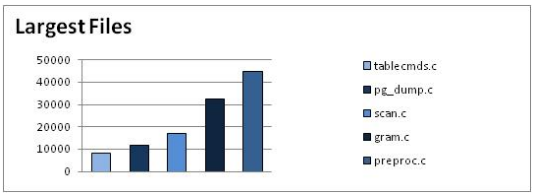


Abbildung D.137.: Largest Files Postgresql Version 8.5alpha1

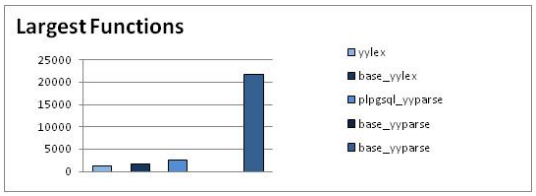


Abbildung D.138.: Largest Functions Postgresql Version 8.5alpha1

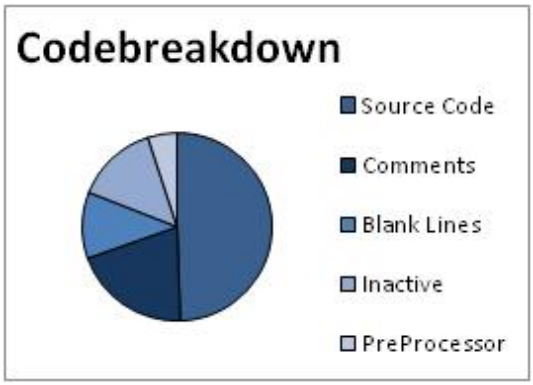


Abbildung D.139.: Code Breakdown Interbase Version 1.0.3.972

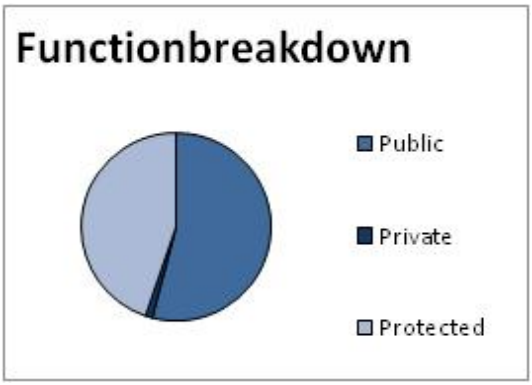


Abbildung D.140.: Function Breakdown Interbase Version 1.0.3.972

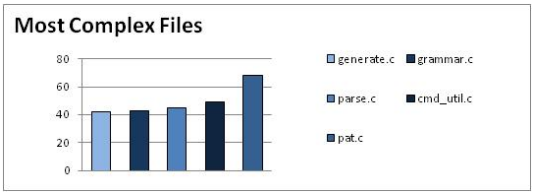


Abbildung D.141.: Most Complex Files Interbase Version 1.0.3.972

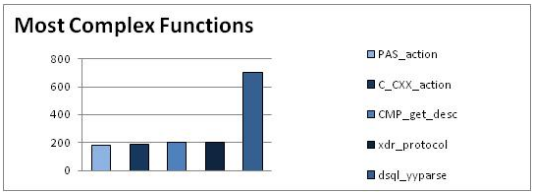


Abbildung D.142.: Most Complex Functions Interbase Version 1.0.3.972

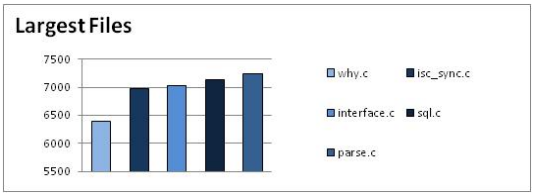


Abbildung D.143.: Largest Files Interbase Version 1.0.3.972

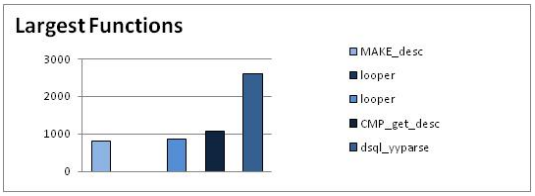


Abbildung D.144.: Largest Functions Interbase Version 1.0.3.972

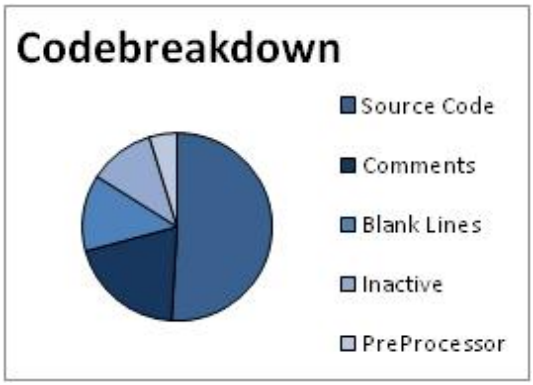


Abbildung D.145.: Code Breakdown Firebird Version 1.5.5.4926

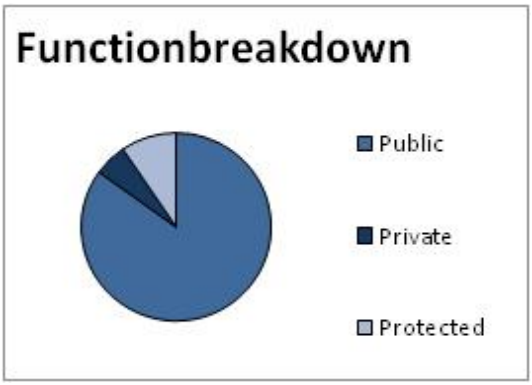


Abbildung D.146.: Function Breakdown Firebird Version 1.5.5.4926

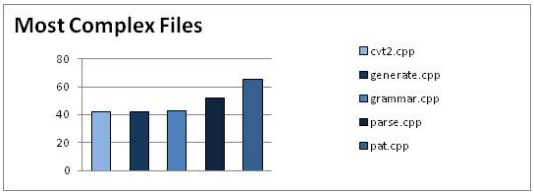


Abbildung D.147.: Most Complex Files Firebird Version 1.5.5.4926

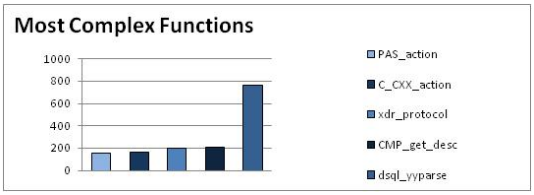


Abbildung D.148.: Most Complex Functions Firebird Version 1.5.5.4926

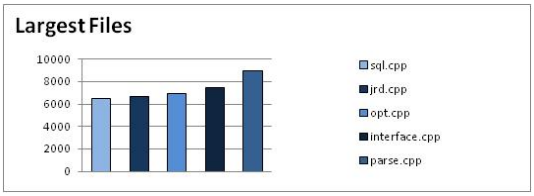


Abbildung D.149.: Largest Files Firebird Version 1.5.5.4926

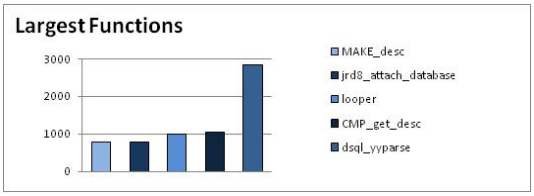


Abbildung D.150.: Largest Functions Firebird Version 1.5.5.4926

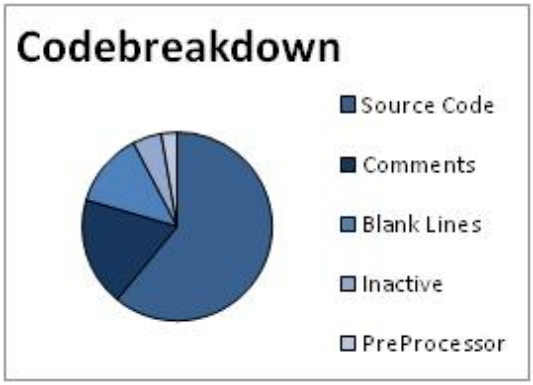


Abbildung D.151.: Code Breakdown Firebird Version 2.5.0.23247-Beta1

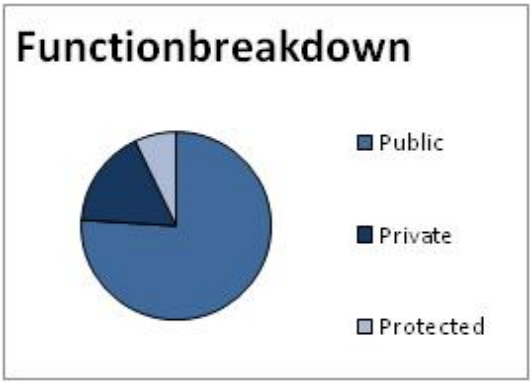


Abbildung D.152.: Function Breakdown Firebird Version 2.5.0.23247-Beta1

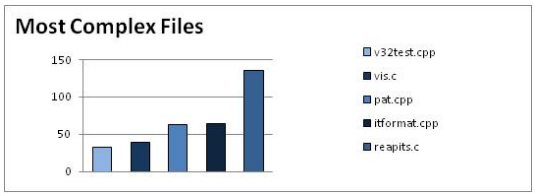


Abbildung D.153.: Most Complex Files Firebird Version 2.5.0.23247-Beta1

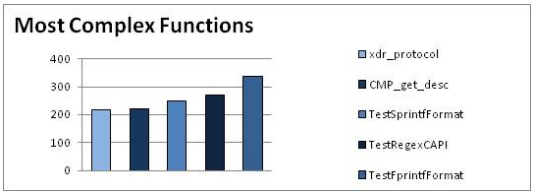


Abbildung D.154.: Most Complex Functions Firebird Version 2.5.0.23247-Beta1

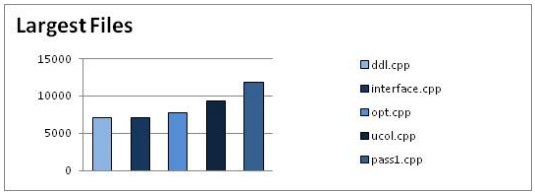


Abbildung D.155.: Largest Files Firebird Version 2.5.0.23247-Beta1

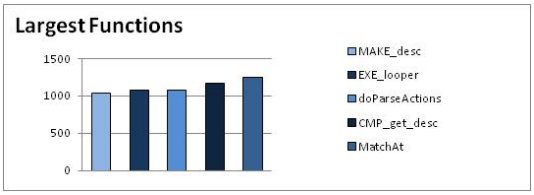


Abbildung D.156.: Largest Functions Firebird Version 2.5.0.23247-Beta1

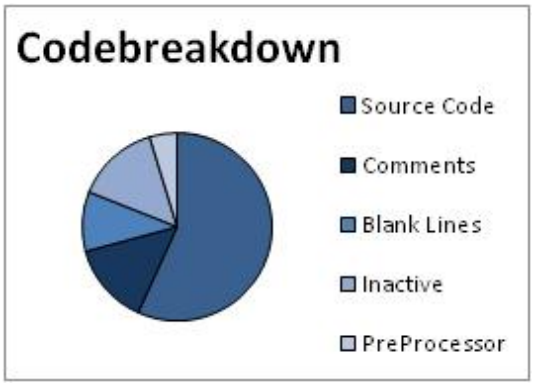


Abbildung D.157.: Code Breakdown Amanda Version 2.4.3b2

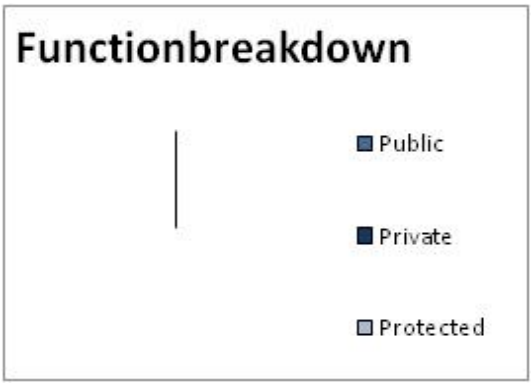


Abbildung D.158.: Function Breakdown Amanda Version 2.4.3b2

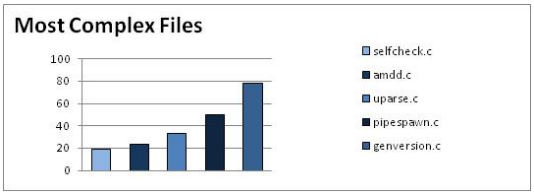


Abbildung D.159.: Most Complex Files Amanda Version 2.4.3b2

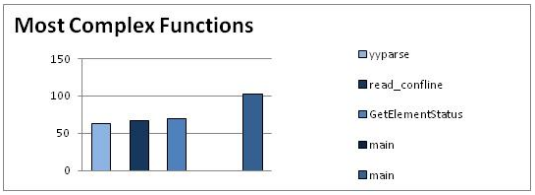


Abbildung D.160.: Most Complex Functions Amanda Version 2.4.3b2

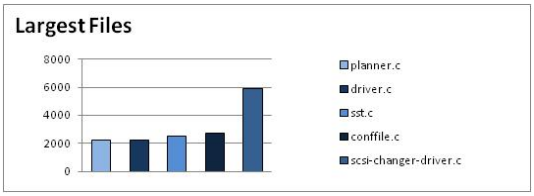


Abbildung D.161.: Largest Files Amanda Version 2.4.3b2

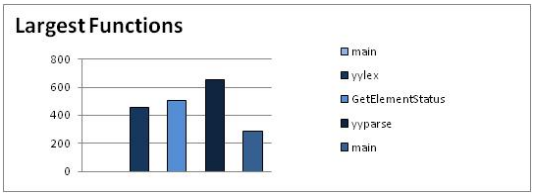


Abbildung D.162.: Largest Functions Amanda Version 2.4.3b2

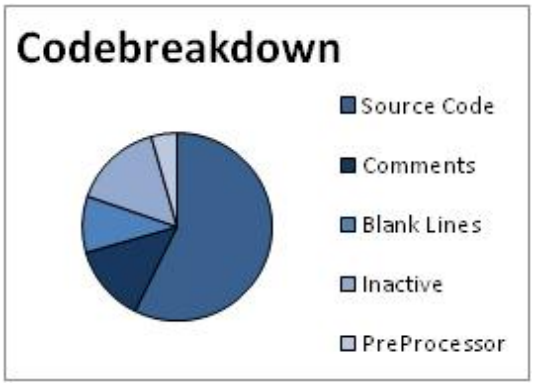


Abbildung D.163.: Code Breakdown Amanda Version 2.5.1p3

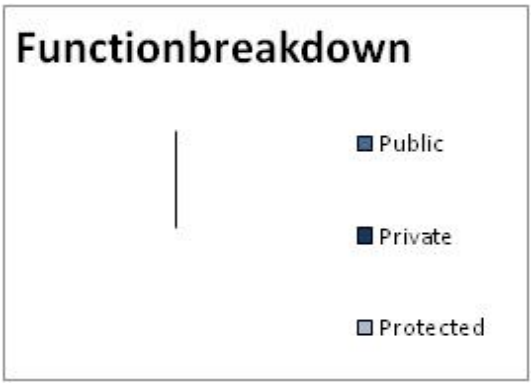


Abbildung D.164.: Function Breakdown Amanda Version 2.5.1p3

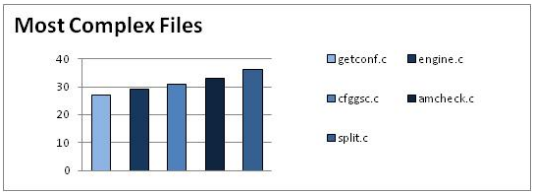


Abbildung D.165.: Most Complex Files Amanda Version 2.5.1p3

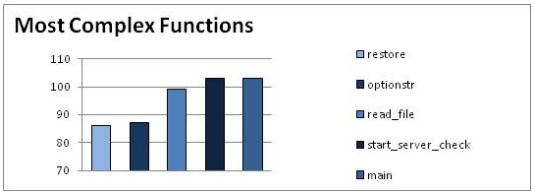


Abbildung D.166.: Most Complex Functions Amanda Version 2.5.1p3

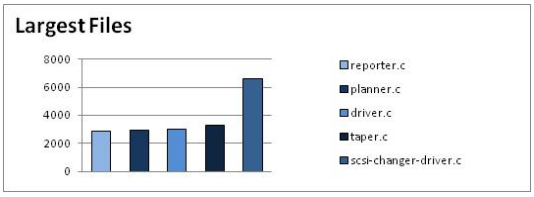


Abbildung D.167.: Largest Files Amanda Version 2.5.1p3

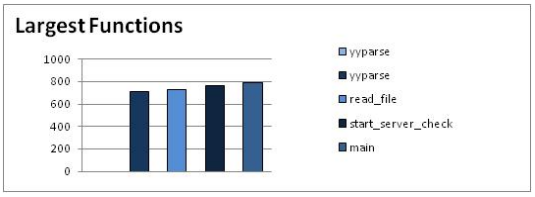


Abbildung D.168.: Largest Functions Amanda Version 2.5.1p3

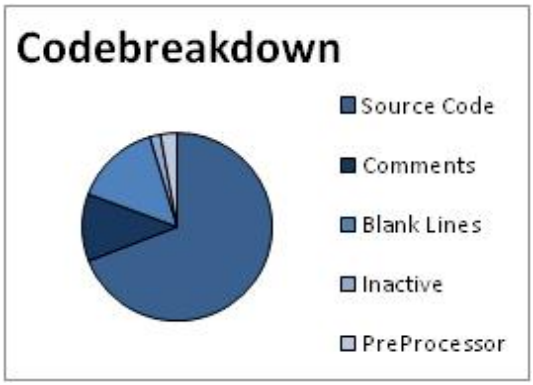


Abbildung D.169.: Code Breakdown Gnucash Version 1.8.4

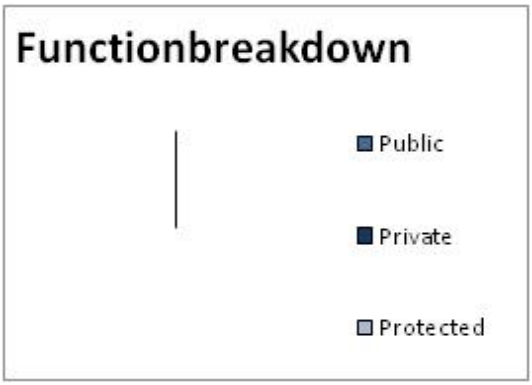


Abbildung D.170.: Function Breakdown Gnucash Version 1.8.4

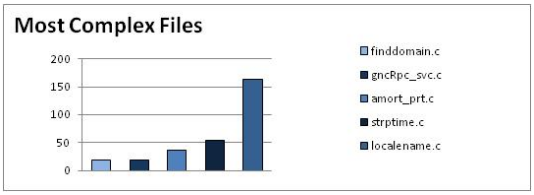


Abbildung D.171.: Most Complex Files Gnucash Version 1.8.4

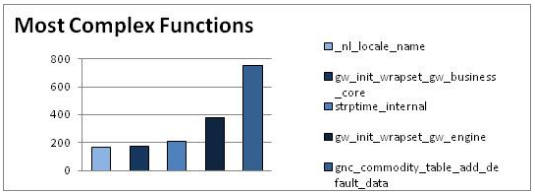


Abbildung D.172.: Most Complex Functions Gnucash Version 1.8.4

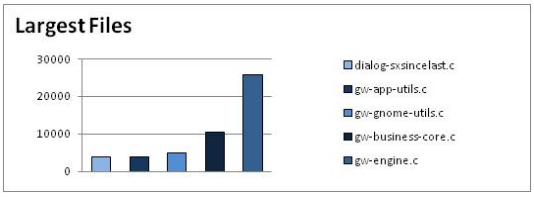


Abbildung D.173.: Largest Files Gnucash Version 1.8.4

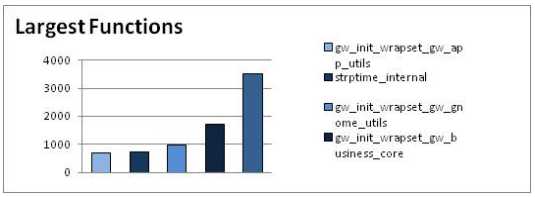


Abbildung D.174.: Largest Functions Gnucash Version 1.8.4

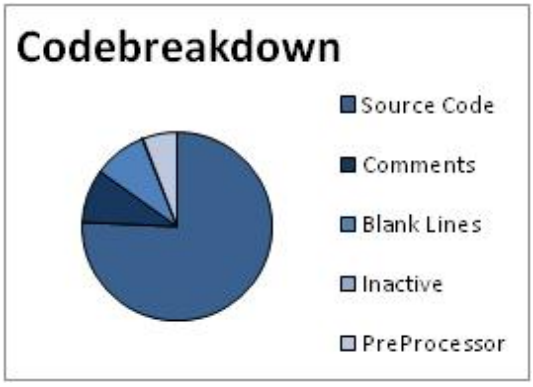


Abbildung D.175.: Code Breakdown Linux Version 0.01

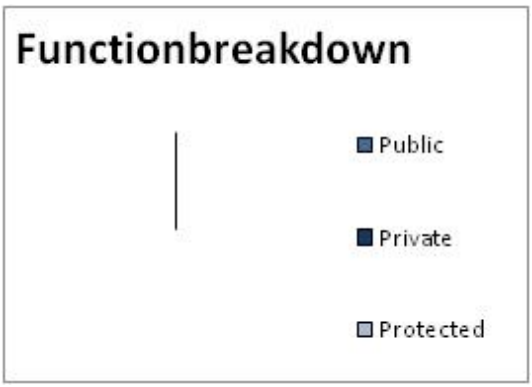


Abbildung D.176.: Function Breakdown Linux Version 0.01

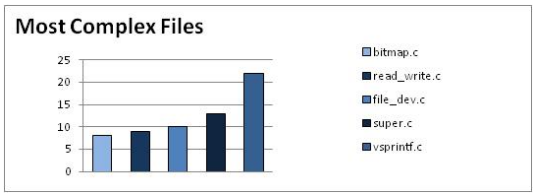


Abbildung D.177.: Most Complex Files Linux Version 0.01

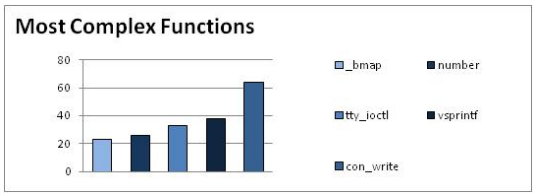


Abbildung D.178.: Most Complex Functions Linux Version 0.01

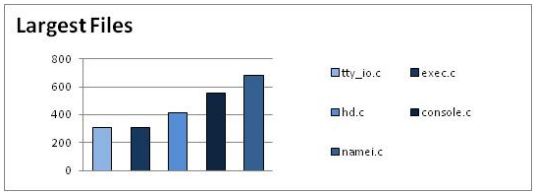


Abbildung D.179.: Largest Files Linux Version 0.01

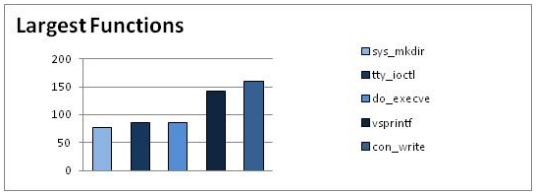


Abbildung D.180.: Largest Functions Linux Version 0.01

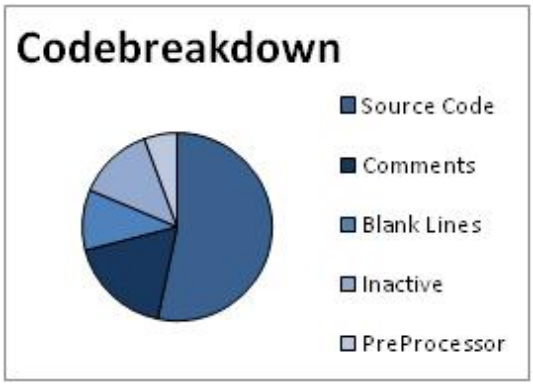


Abbildung D.181.: Code Breakdown Linux Version 2.1.105

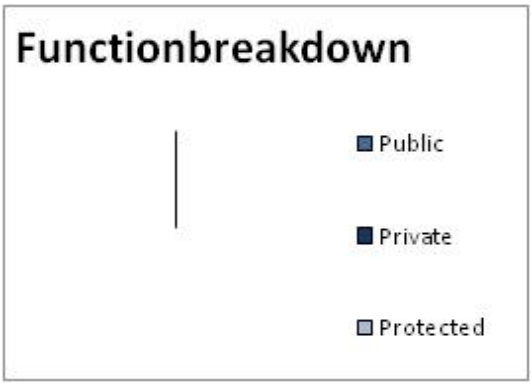


Abbildung D.182.: Function Breakdown Linux Version 2.1.105

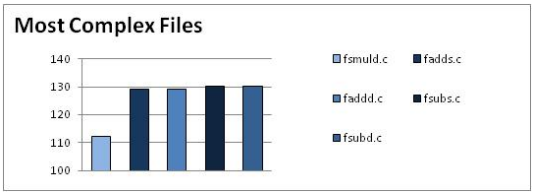


Abbildung D.183.: Most Complex Files Linux Version 2.1.105

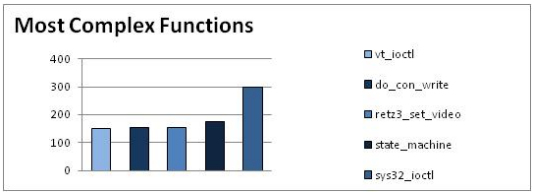


Abbildung D.184.: Most Complex Functions Linux Version 2.1.105

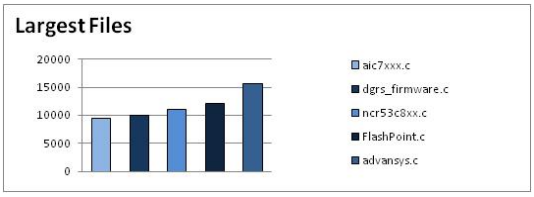


Abbildung D.185.: Largest Files Linux Version 2.1.105

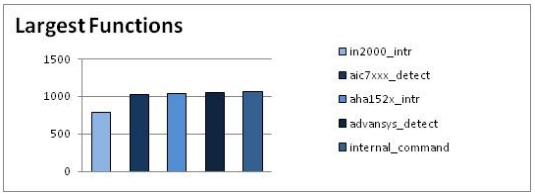


Abbildung D.186.: Largest Functions Linux Version 2.1.105

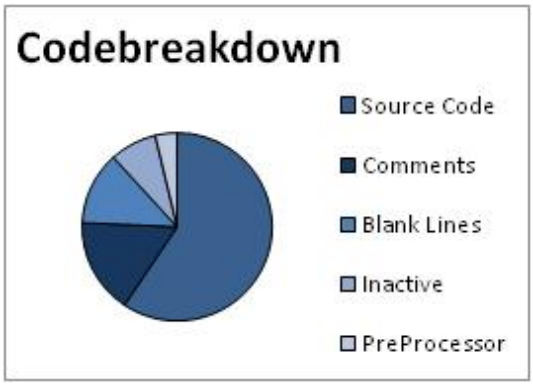


Abbildung D.187.: Code Breakdown Linux Version 2.6.30.2

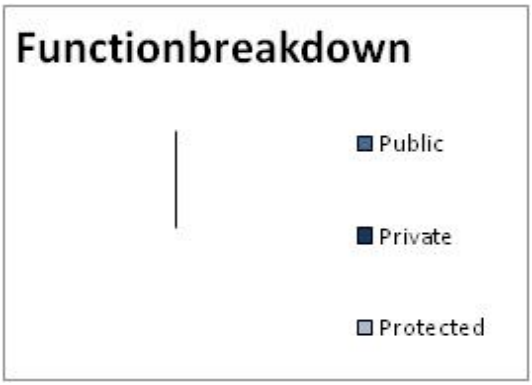


Abbildung D.188.: Function Breakdown Linux Version 2.6.30.2

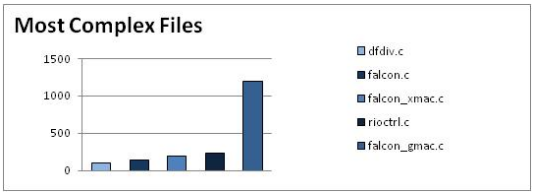


Abbildung D.189.: Most Complex Files Linux Version 2.6.30.2

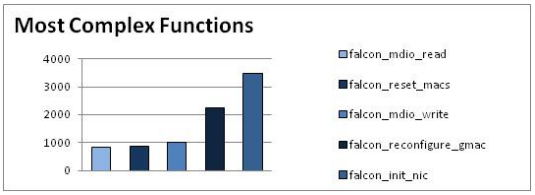


Abbildung D.190.: Most Complex Functions Linux Version 2.6.30.2

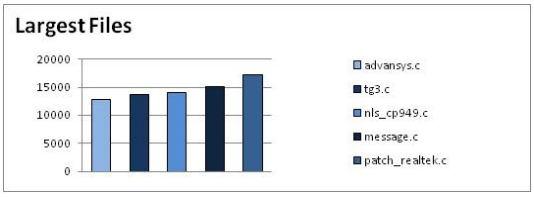


Abbildung D.191.: Largest Files Linux Version 2.6.30.2

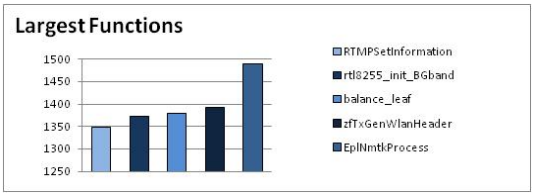


Abbildung D.192.: Largest Functions Linux Version 2.6.30.2

E. CD-ROM

Die beiliegende CD-ROM enthält folgende Informationen:

- Programme
  - Understand (Demo Version)
  - Lattix (Demo Version)
  - CsvToDSM (Vollversion inkl. Quellcode)
- SourceCode der analysierten OSS
- Analyse Ergebnisse

CD-ROM: