Universität Passau

Fakultät für Informatik und Mathematik

Masterarbeit

Aufstieg in der Linux-Entwickler-Community: Eine Analyse der Patch-Acceptance-Rate

Autor:

Manuel Donaubauer

9. Februar 2018

Betreuer:

Prof. Dr.-Ing. Sven Apel

Lehrstuhl für Software Engineering

Prof. Dr. Carolin Häussler

Lehrstuhl für Betriebswirtschaftslehre mit Schwerpunkt Organisation, Technologiemanagement und Entrepreneurship

Donaubauer, Manuel:
Aufstieg in der Linux-Entwickler-Community: Eine Analyse der Patch-Acceptance-Rate
Masterarbeit, Universität Passau, 2018.

Inhaltsangabe

Der Linux-Kernel ist der zentrale Bestandteil des Linux Betriebssystems. Dadurch, dass es sich bei Linux um Open-Source-Software handelt, arbeiten Entwickler aus der ganzen Welt an diesem Projekt. Um die große Anzahl an Entwicklern steuern zu können, nehmen bestimmte Personen eine leitende Position ein. Diese werden als Maintainer bezeichnet. Die vorliegende Arbeit versucht heraus zu finden, ob die Patch-Acceptance-Rate einen Einfluss darauf hat, ob und wie schnell ein Entwickler zum Maintainer aufsteigt. Die Patch-Acceptance-Rate beschreibt dabei, wie oft ein Entwickler seine Codeänderung (Patch) nachbessern muss, bevor der jeweilige Maintainer diese akzeptiert und im Linux-Kernel übernommen wird. Um zu Analysieren, ob die Patch-Acceptance-Rate einen Einfluss auf die Aufstiegschancen von Entwicklern hat, wurden Daten über die E-Mail Kommunikation, welche auch die Patches enthalten, und Daten aus dem Versionskontrollsystem zu einem Datensatz zusammengefügt.
Die zentrale Fragestellung ist, welche Einflussfaktoren es gibt, die den Aufstieg von Entwicklern in der Hierarchie der Linux-Kernel-Entwicklung begünstigen. In dieser Ausarbeitung wurden dabei vier Hypothesen aufgestellt. Diese besagen, dass die Aufstiegschancen von Entwicklern positiv durch eine große Anzahl an technischensowie den kommunikativen Beiträgen beeinträchtigt werden. Eine weitere Behauptung ist, dass eine hohe Akzeptanz der Patches eines Entwicklers zum schnelleren Aufstieg führt. Die letzte Hypothese besagt, dass Entwickler, die viel mit wichtigen Maintainern direkt kommunizieren, ebenfalls schneller in der Hierarchie aufsteigen.
Das Ergebnis der Analyse zeigt, dass die technischen Einflussfaktoren, wie eine große Anzahl an technischen Beiträgen und eine hohe Akzeptanz der Patches durch Maintainer, die Wahrscheinlichkeit, irgendwann Maintainer zu werden, positiv beeinflussen. Kommuniziert ein Entwickler viel über die Mailingliste, hat dies ebenfalls einen positiven Einfluss auf die Aufstiegschancen. Schreibt ein Entwickler viele E-Mails mit wichtigen Maintainer, kann ein positiver Effekt auf die Wahrscheinlichkeit, dass ein Entwickler irgendwann Maintainer wird, nicht bestätigt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

1. Einführung

Der Linux-Kernel ist ein Open-Source-Projekt, das für jeden zugänglich ist, und bei dem jeder mitwirken darf. Es ist der zentrale Bestandteil des Linux-Betriebssystems. Durch eine sehr große, weltweite Entwickler-Community, die ein öffentliches Versionskontrollsystem zur Source-Code-Verwaltung nutzt und über öffentliche Mailinglisten kommuniziert, eignet sich der Linux-Kernel bestens für wissenschaftliche Forschungen und Analysen der Aufstiegschancen von Entwicklern bei großen Softwareprojekten.

Beim Linux-Kernel gibt es unterschiedliche Arten von Entwicklern: die „normalen“ Entwickler, welche neue Features hinzuzufügen, Bugs fixen, oder Code Refactoring durchführen und sogenannte Maintainer, welche eine Steuerungsfunktion beim Projekt einnehmen. Maintainer entscheiden, ob eine Code-Änderung (Patch) eines Entwicklers ins Projekt übernommen wird, oder ob der jeweilige Entwickler nochmal nachbessern muss. Wie oft ein Entwickler nachbessern muss, wird dabei durch eine Patch-Acceptance-Rate beschrieben. Da das Einreichen von Patches wie die normale Kommunikation über die Mailingliste erfolgt, kann die Patch-Acceptance-Rate mithilfe des öffentlich zugänglichen Versionskontrollsystems und den darin enthaltenen Commits, berechnet werden. Ein Commit ist dabei ein Patch in der Mailingliste, der von einem Maintainer akzeptiert und ins Projekt übernommen wurde. Diese Ausarbeitung versucht herauszufinden, ob Faktoren, wie die Patch-Acceptance-Rate, einen Einfluss darauf haben, ob und wie schnell ein Entwickler den Maintainer-Status erlangt. Ein weiteres Ziel ist es, Probleme bei der Datenbeschaffung und der Aufbereitung zu identifizieren, um diese in weiterführenden Arbeiten beseitigen zu können.

Um die Fragestellung zu beantworten, werden vier Hypothesen aufgestellt. Diese besagen, dass die Aufstiegschancen von Entwickler, die viele technische- und kommunikative Beiträge zum Linux-Kernel-Projekt leisten, höher sind. Eine weitere Hypothese lautet, dass eine hohe Akzeptanz von technischen Beiträge durch Maintainer die Aufstiegschancen der Entwickler beschleunigt. Die letzte besagt, dass die direkte Kommunikation mit wichtigen Maintainern, wie zum Beispiel Linus Tor-

valds, ebenfalls den Aufstieg zum Maintainer beschleunigt.

Eine statistische Cox-Analyse zeigt, dass sich bei Entwicklern, die viele Commits in den master-Branch bekommen, die Aufstiegschancen irgendwann Maintainer zu werden, erhöhen. Das beutet, dass Hypothese 1 durch die Ergebnisse der Analyse bestätigt werden. Hypothese 2, dass eine hohe Akzeptanz der Patches durch Maintainer sich positiv auf Aufstiegschancen von Entwickler auswirkt, kann durch die statistische Auswertung ebenfalls bestätigt werden. Sendet ein Entwickler viele E-Mails an die Mailingliste, zeigt die Analyse, dass dieser auch erhöhte Chancen hat, irgendwann Maintainer zu werden. Hypothese 3 entspricht somit der Wahrheit. Die letzte Hypothese, dass direkte Kommunikation mit wichtigen Maintainern die Aufstiegschancen erhöhen, konnte nicht bestätigt werden.
Die Forschungsfrage, ob es Einflussfaktoren gibt, die die Aufstiegschancen von Entwicklern im Linux-Kernel positiv beeinflussen, kann mit ja beantwortet werden. Die untersuchten, technischen Faktoren wirken sich alle positiv darauf aus.
Nach dem Erklären der Grundlagen in Kapitel 2, wird auf die Forschungsfragen und Hypothesen im Kapitel 3 eingegangen. Danach folgt das Aufbereiten der Ausgangsdaten in Kapitel 4. Dazu gehören Release-Informationen, die Maintainer-Liste, die Mailingliste, die Commit-Liste und die zu berechnende Patch-Acceptance-Rate. In Kapitel 5 wird die statistische Cox-Analyse vorgestellt. Nach der Evaluierung in Kapitel 6 wird in Kapitel 7 auf verwandte und in Kapitel 8 zukünftige Arbeiten eingegangen.

2. Grundlagen

Das folgende Kapitel erläutert die Grundlagen, welche für das Verständnis dieser Ausarbeitung benötigt werden. Dabei wird zunächst auf das Linux-Betriebssystem und den Linux-Kernel eingegangen. Eine Erklärung, welche Entwickler als Maintainer bezeichnet werden und was Subsysteme sind, wird ebenfalls in diesem Kapitel aufgeführt.

2.1 Der Linux-Kernel

Linux ist ein Betriebssystem, ähnlich wie Windows oder macOS. Es ist kostenlos und wird von einer weltweiten Entwicklergemeinschaft, der jeder Beitreten kann, entwickelt und als Open-Source zum Download angeboten. Eine Besonderheit von Linux ist, dass es unterschiedliche Distributionen davon gibt. Eine Distribution ist dabei speziell für einen bestimmten Anwendungsbereich angepasste Version von Linux mit bereits vorinstallierten Tools. So gibt es zum Beispiel eine spezielle Version für den Rasperry Pi mit dem Namen Raspbian [ras]. Die Distribution Debian wird oft für Server verwendet, da es leicht zu installieren, fast beliebig anpassbar und viele Architekturen unterstützt [Bec]. Eine weitere beliebte Distribution ist ubuntu. Durch die gute Unterstützung von vielen Multimedia-Inhalten und einer hohen Benutzerfreundlichkeit ist es ein beliebtes Betriebssystem für den privaten Anwender [Bec].
Da Linux so vielfältig ist, sind die Einsatzmöglichkeiten nahezu grenzenlos. Um die Vielseitigkeit zu gewährleisten, wurde Linux in mehrere einzelne Bereiche unterteilt. Dabei unterscheidet man zwischen Bootloader, Daemons, Shell, Graphical Server, Desktop Environment, Applications und dem Kernel. Letzterer ist der Teil, der eigentlich den Namen Linux trägt und in jeder Distribution vorhanden ist. Der Linux-Kernel ist dabei das Kernstück des Betriebssystems und agiert mit der Hardware [Fou].

2.2 Maintainer und Subsysteme

Da es bei einem so großen Projekt, wie den Linux-Kernel, sehr viele Entwickler gibt, muss eine gewisse Organisation und Steuerung der Entwicklung stattfinden. Jeder Entwickler programmiert dabei lokal auf seinem Rechner. Soll eine Änderung in den master-Branch übernommen werden, muss der jeweilige Entwickler seine Änderung als Patch über eine passende Mailingliste einreichen. Um die Entwicklung zu Steuern und Entscheidungen darüber zu treffen, ob ein Patch sinnvoll ist und übernommen werden soll oder nicht, werden sogenannte Maintainer benötigt. Die Berufung zum Maintainer erfolgt immer im Zeitraum von einem Release und kann von einem anderen Maintainer, aber auch aus Eigeninitiative durch Eintragen in die Maintainer Liste, erfolgen. Die Tätigkeit als Maintainer setzt voraus, dass dieser sehr vertraut mit den einzelnen Komponenten des Linux-Kernels und deren Entwicklung ist. Welcher Entwickler zu welchem Zeitpunkt Maintainer war oder ist, kann aus einer Datei im Versionskontrollsystem ausgelesen werden (vgl. Abschnitt 4.2). Unter den einzelnen Maintainern gibt es eine bestimmte Hierarchie. Die wichtigste Person bei der Linux-Kernel-Entwicklung ist dabei Linus Torvalds. Er ist der Initiator des Linux-Kernels und wird als ranghöchster Maintainer von allen akzeptiert. Weitere wichtige Maintainer sind zum Beispiel Andrew Morton oder David S. Miller [Fou10]. Der Linux-Kernel wächst von Release zu Release. Bei Release 2.6.12 im Jahr 2005 waren knapp 5,5 Millionen Lines of Code vorhanden und 186 Maintainer aktiv.

Tabelle 2.1: Überischt über Maintaineranzahl und LoC für verschiedene Releases [Prob]

Release	Timestamp	Anzahl an Maintainer	Lines of Code
2.6.12	2005-06-17	186	5 404 072
2.6.17	2006-06-17	240	6 043 106
2.6.22	2007-07-08	328	6 721 395
2.6.27	2008-10-09	444	7 538 961
2.6.32	2009-12-02	584	9 771 588
2.6.37	2011-01-04	638	11 315 921
3.2	2012-01-04	708	12 201 423
3.7	2012-12-10	760	13 157 359
3.12	2013-11-03	826	14 539 764
3.17	2014-10-05	871	15 216 469
4.2	2015-08-30	954	16 354 974
4.7	2016-07-24	1039	17 411 340

Mittlerweile stieg die Anzahl an Maintainer bei Release 4.7 auf über 1000 an. Die Lines of Code haben sich im Zeitraum von Version 2.6.12 bis 4.7 mehr als verdreifacht (siehe Abbildung 2.1, Tabelle 2.1).

Abbildung 2.1: Grafische Darstellung der Anzahl an Maintainer und LoC für verschiedene Releases

Da ein Betriebssystem, wie der LINUX-Kernel, sehr komplex und umfangreich ist, wurde dieser in sogenannte Subsysteme eingeteilt. Ein Subsystem ist ein Teilsystem des LINUX-Kernels, das ähnliche Funktionalitäten gruppiert. Diese können unterschiedliche Größe und Wichtigkeit besitzen. Es gibt sieben große Haupt-Subsysteme.

- System Call Interface (SCI)
- Process Management (PM)
- Memory Management (MM)
- Virtual File System (VFS)
- Network Stack
- Device Drivers (DD)
- Architecture-Dependent Code

Das *SCI-Subsystem* sorgt dafür, dass Funktionsaufrufe von der Benutzerebene in den Kernel gelangen. Dabei kann dieses Interface architekturabhängig sein. Die Implementierung befindet sich in den Pfaden `./linux/kernel` und `./linux/arch`.
Das Subsystem *Process Management* ist für das Ausführen von Prozessen verantwortlich. Ein Scheduler, der die Prozesse auf der geteilten CPU plant, ist ebenfalls in diesem Subsystem enthalten. Die Resourcen des Process Management Subsystems sind unter den Pfaden `./linux/kernel` und `./linux/arch` zu finden.
Ein weiteres, wichtiges Subsystem des LINUX-Kernels ist das *Memory Management*.

Dieses Subsystem verwaltet sämtliche Funktionalitäten rund um die Speicherverwaltung wie zum Beispiel das Swapping. Die Implementierung dieses Subsystems kann im Pfad `./linux/mm` gefunden werden.
Das *Virtual File System* Subsystem bietet eine Interface-Abstraktion für das File System. Dabei stellt es eine Vermittlungsschicht zwischen dem SCI und den vom Kernel unterstützen File Systemen zur Verfügungen. Die Implementierung dieses Subsystems ist im Pfad `./linux/fs` zu finden.
Im *Network Stack* Subsystem werden sämtliche Netzwerkprotokolle implementiert. Dazu gehören Protokolle wie zum Beispiel IP, UDP, TCP, ... Der Aufbau des Subsystems orientiert sich dabei an der Architektur der Netzwerk Schichten. Im Pfad `./linux/net` befinden sich sämtliche Implementierungen, die zu diesem Subsystem gehören.
Damit der Linux-Kernel mit der unterschiedlichsten Hardware kompatibel ist, werden Treiber benötigt. Diese sind im Subsystem *Device Drivers* enthalten. Dieses nimmt dabei einen sehr großen Teil des Kernels ein. Unter `./linux/drivers` ist die Implementierung der Treiber zu finden.
Das letzte große Subsystem *Architecture-Dependent Code* beinhaltet architekturabhängigen Code und befindet sich unter den Pfad `./linux/arch` [Jon].

Da diese großen, umfangreichen Subsysteme nicht von einem einzigen Maintainer verwaltet werden können, wurden diese nochmals in kleinere Subsysteme unterteilt. Diese werden im folgenden Verlauf der Arbeit auch als Subsysteme bezeichnet.
So ist zum Beispiel Clemens Ladisch Maintainer für ein sehr kleines Subsystem (BT87X Audio Treiber) mit nur zwei Dateien verantwortlich. Dieses kapselt die Funktionalität der Midi-Wiedergabe bei Linux [proa].

```
1 BT87X AUDIO DRIVER
2 M:  Clemens Ladisch <clemens@ladisch.de>
3 L:  alsa-devel@alsa-project.org (moderated for non-subscribers)
4 T:  git git://git.alsa-project.org/alsa-kernel.git
5 S:  Maintained
6 F:  Documentation/sound/alsa/Bt87x.txt
7 F:  sound/pci/bt87x.c
```

Listing 2.1: Maintainer-Daten von Clemens Ladisch im Subsystem BT87X AUDIO DRIVER

Das Subsystem ARM/ACTIONS SEMI ARCHITECTURE ist deutlich größer als das vorherige. Es umfasst komplette Ordner und nicht nur einzelne Dateien, wie im Subsystem BT87X AUDIO DRIVER. Dieses Subsystem sorgt für die Unterstützung ARM Prozessoren im Linux-Kernel und ist damit auch viel wichtiger als die Wiedergabe von Midi-Dateien. Es ist Teil des Haupt-Subsystems Architecture-Dependent Code.

Eine genaue Beschreibung der einzelnen Informationen erfolgt in Abschnitt 4.2.

```
1  ARM/ACTIONS SEMI ARCHITECTURE
2  M:  Andreas Farber <afaerber@suse.de>
3  L:  linux-arm-kernel@lists.infradead.org
4          (moderated for non-subscribers)
5  S:  Maintained
6  N:  owl
7  F:  arch/arm/mach-actions/
8  F:  arch/arm/boot/dts/owl-*
9  F:  arch/arm64/boot/dts/actions/
10 F:  drivers/clocksource/owl-*
11 F:  drivers/soc/actions/
12 F:  include/dt-bindings/power/owl-*
13 F:  include/linux/soc/actions/
14 F:  Documentation/devicetree/bindings/arm/actions.txt
15 F:  Documentation/devicetree/bindings/power/actions,owl-sps.txt
16 F:  Documentation/devicetree/bindings/timer/actions,owl-timer.txt
```

Listing 2.2: Maintainer-Daten von Andreas Färber im Subsystem ARM/ACTIONS SEMI ARCHITECTURE

3. Forschungsfrage und Hypothesen

Bei großen Open-Source-Projekten, wie dem Linux-Kernel, arbeiten viele Entwickler zusammen. Damit die Zusammenarbeit möglichst effizient und geregelt abläuft, nehmen sogenannte Maintainer eine leitende Position ein und entscheiden darüber, ob die Änderungen der Entwickler übernommen werden. Entwickler können technische- aber auch durch kommunikative Beiträge leisten. Da es nicht jeder Entwickler zum Maintainer schafft, muss es gewisse Einflussfaktoren geben, die Berufung zum Maintainer beeinflussen. Dadurch komme ich zu meiner Forschungsfrage:
Welche Einflussfaktoren gibt es, die den Aufstieg von Entwicklern in der Hierarchie der Linux*-Kernel-Entwicklung begünstigen?*

Maintainer bewerten Patches von Entwicklern und entscheiden darüber, ob eine Änderung in den master-Branch übernommen wird oder nicht. Aus diesem Grund müssen Maintainer technisch kompetent sein und sich in ihrem Subsystem bestens auskennen. Um dieses Wissen über ein bestimmtes Subsystem zu erlangen, muss sich ein Entwickler intensiv mit dem Subsystem auseinandersetzen. Dies kann durch viele technische Beiträge in Form von Commits stattfinden. Dadurch erscheinen solche Entwickler glaubwürdiger und vertrauenswürdiger [DO11]. Dies erhöht die Aufstiegschancen des Entwicklers, zu einen Maintainer aufzusteigen. Dies führt uns zu unserer ersten Hypothese:
Je mehr technische Beiträge ein Entwickler zum Linux*-Kernel beiträgt, desto größer ist die Wahrscheinlichkeit, dass er irgendwann Maintainer wird.*

In Abbildung 3.1 wird der positive Einfluss der technischen Beiträge in H1 (+) dargestellt.

Wenn ein Entwickler einen Patch über eine Mailingliste einreicht, kann es sein, dass ein Maintainer diesen nicht sofort akzeptiert und der Patch nachgebessert werden muss. Der Kehrwert der Anzahl an Anläufe bis der Patch im master-Branch ist,

wird dabei als Patch-Acceptance-Rate bezeichnet. Wenn die Patches eines Entwicklers mit wenigen Nachbesserungen von einem Maintainer übernommen werden, ist die Patch-Acceptance-Rate hoch. Dies kann ein Zeichen für gute Qualtität der technischen Beiträge sein und zeigt aber vor allem die Akzeptanz der Patches durch die Maintainer [RR16]. Hohe Akzeptanz bei den Patches eines Entwicklers kann dabei zu einem schnelleren Aufstieg zum Maintainer führen. Dies führt zur zweiten Hypothese:
Hohe Akzeptanz von technischen Beiträgen durch Maintainer beschleunigt die Aufstiegschancen von Entwickler.

Die Visualisierung des positiven Einflusses der unabhängigen Variable wird in Abbildung 3.1 mit H2 (+) bezeichnet.

Neben den technischen- sind die kommunikativen Beiträge bei der Entwicklung des Linux-Kernels ein wichtiger Bestandteil. Über die Mailinglisten werden während der Arbeit Fragen von Entwicklern gestellt und auch beantwortet. Dies fördert den Lernvorgang der Entwickler. Schreibt ein Entwickler viele E-Mails, teilt er sein Wissen mit anderen und hilft diesen weiter. Das führt zu höheren Ansehen in der Entwickler-Community [DO11]. Dadurch steigt auch die Wahrscheinlichkeit, dass der Entwickler zum Maintainer aufsteigt. Dies führt zur nächsten Hypothese:
Je mehr ein Entwickler über die Mailingliste kommuniziert, desto größer ist die Wahrscheinlichkeit, dass er irgendwann Maintainer wird.

Beim Hypothesenmodell in Abbildung 3.1 wird der positive Einfluss der unabhängigen Variable mit H3 (+) bezeichnet.

Wie in Hypothese 3 bereits erwähnt, ist die Kommunikation bei der Entwicklung von Open-Source-Software ein wichtiger Bestandteil. Kommuniziert ein Entwickler direkt mit ranghöheren Maintainern, können diese ein engere Bindung aufbauen [JH08]. Dadurch erhöht sich die Wahrscheinlichkeit, dass diese Entwickler schneller zu einem Maintainer werden. Dies führt zur vierten und letzten Hypothese:
Die direkte Kommunikation mit wichtigen Maintainern beschleunigt die Aufstiegschancen von Entwicklern in der Linux*-Projekthierarchie.*

Der positive Einfluss dieser unabhängigen Variable auf die Wahrscheinlichkeit, den Maintainer Status irgendwann zu erreichen, wird in Abbildung 3.1 als H4 (+) visualisiert.

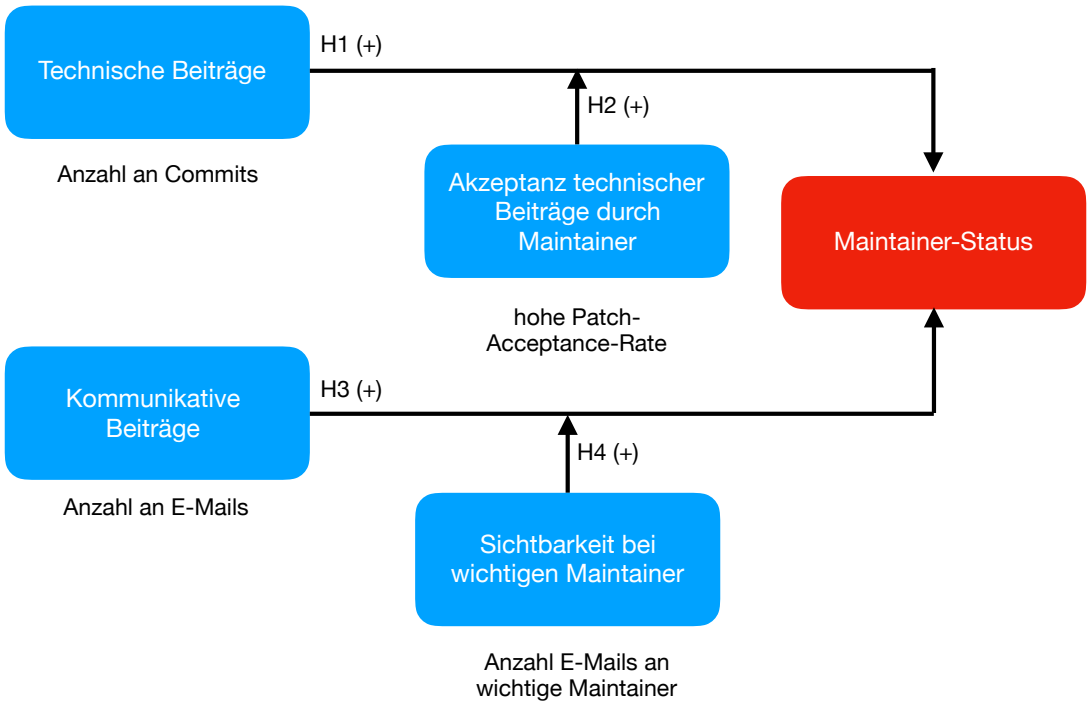

Abbildung 3.1: Hypothesenmodell

Die abhängige Variable wird im roten Kasten dargestellt.
Die unabhängigen Variablen werden durch die blauen Kasten visualisiert.
Durch welche messbare Größen die jeweiligen unabhängigen Variablen umgesetzt werden, wird unterhalb der blauen Kasten beschrieben.
Welche Hypothese durch welche Variable abgedeckt wird, wird durch die Bezeichnung H1 - H4 dargestellt.
Durch (+) wird ein positiver Einfluss der jeweiligen Variable visualisiert.

4. Datenaufbereitung

4.1 Releases

Wie bei jeder Software gibt es auch beim Linux-Kernel unterschiedliche Releases mit dazugehörigen Versionsnummern. Da sich der Status eines Entwicklers, ob er ein Maintainer ist oder nicht, von Version zu Version ändern kann, werden die genauen Start- und Endzeitpunkte für jede Version für die Analyse benötigt. Um diese Informationen zu erhalten wurde folgender Git-Befehl im Terminal auf dem Repository ausgeführt.

```
1 git log --tags --simplify-by-decoration --pretty="format:%ai %d"
```

Listing 4.1: Git Befehl zum Auslesen der Releaseinformationen.

Man erhält den genauen Zeitpunkt (Datum, Uhrzeit und Zeitzone) wann jeder Release gestartet wurde. Um die genaue Release-Bezeichnung zu erhalten wurde `tags` in den Befehl mit aufgenommen. Die älteste Version ist dabei 2.6.12, da zuvor das Git Repository nicht existierte. Release Candidates (Tags mit -rc) oder Releases ohne Tags werden bei der Aufbereitung nicht beachtet. Da zum Zeitpunkt der Analyse die zur Verfügung stehenden Daten nur Informationen bis Version 4.7 enthalten, sind die Releases 4.8 und 4.9 nicht relevant. Sie wurden nur wegen der Vollständigkeit aufgelistet.
So wurde zum Beispiel das Release mit der Versionsnummer 4.7 am 24.07.2016 um 12:23:50 in der Zeitzone -0700 gestartet. Der Endzeitpunkt ist der Release der nächst größeren Version 4.8 minus einer Sekunde, also 02.10.2016 16:24:32 -0700.
Die gekürzte Ausgabe sieht wie folgt aus.

```
2016-11-13 10:32:32 -0800  (tag: v4.9-rc5)
2016-11-05 16:23:36 -0700  (tag: v4.9-rc4)
2016-10-29 13:52:02 -0700  (tag: v4.9-rc3)
2016-10-23 17:10:14 -0700  (tag: v4.9-rc2)
2016-10-15 12:17:50 -0700  (tag: v4.9-rc1)
2016-10-05 14:50:51 -0700
2016-09-19 12:29:33 +0200
2014-08-11 15:29:23 +0800
2016-10-02 16:24:33 -0700  (tag: v4.8)
2016-09-25 18:47:13 -0700  (tag: v4.8-rc8)
2016-09-18 17:27:41 -0700  (tag: v4.8-rc7)
...
2016-07-24 12:23:50 -0700  (tag: v4.7)
...
2005-06-17 12:48:29 -0700  (tag: v2.6.12)
2005-06-06 08:22:29 -0700  (tag: v2.6.12-rc6)
...
(END)
```

Listing 4.2: Auszug der Terminalausgabe zum Auslesen der Releaseinformationen.

Damit die generierten Informationen vernünftig weiterverarbeitet werden können, müssen diese in ein passendes Format konvertiert werden. Durch hinzufügen von Kommas hinter der Zeitzonenangabe und den Spaltennamen erhält man eine CSV-Datei. Ein Auszug, der aufbereiteten Ausgabe kann in der folgenden Tabelle eingesehen werden.

Tabelle 4.1: Auszug der aufbereiteten Release Liste.

date	**release**
2016-10-02 16:24:33 -0700	v4.8
2016-07-24 12:23:50 -0700	v4.7
2016-05-15 15:43:13 -0700	v4.6
2016-03-13 21:28:54 -0700	v4.5
2016-01-10 15:01:32 -0800	v4.4
2015-11-01 16:05:25 -0800	v4.3
...	...
2005-06-17 12:48:29 -0700	v2.6.12

Unnötige Informationen, wie zum Beispiel Releases ohne tag oder Release Candidates wurden entfernt und von jedem tag nur die eigentliche Versionsnummer übernommen. In den späteren Schritten wird diese aufbereitete Datei als Input verwendet um den jeweiligen Zeitpunkten der Commits und E-Mails einen Release zuordnen zu können.

4.2 Maintainer

Um eine Analyse erfolgreich durchführen zu können, wird eine Auflistung aller Maintainer benötigt. Diese Informationen lassen sich ebenfalls aus dem Git-Repository generieren. Durch Ausführen eines kleinen Shell Skripts erhält man für jede Version ei-

ne Liste mit den dazugehörigen E-Mail Adressen und optional den Namen der Maintainer. Der Dateiname hat dabei folgenden Aufbau: `maintainers.<Versionsnummer>`. Das folgende Listing zeigt einen kleinen Ausschnitt der Maintainer von Version 4.7 (`maintainers.v4.7`).

```
1  ...
2  Christian Lamparter <chunkeey@googlemail.com>
3  Olof Johansson <olof@lixom.net>
4  Olof Johansson <olof@lixom.net>
5  Steffen Klassert <steffen.klassert@secunet.com>
6  Harald Welte <laforge@gnumonks.org>
7  David Howells <dhowells@redhat.com>
8  Willy Tarreau <willy@haproxy.com>
9  Ksenija Stanojevic <ksenija.stanojevic@gmail.com>
10 Sudip Mukherjee <sudipm.mukherjee@gmail.com>
11 Sudip Mukherjee <sudip@vectorindia.org>
12 Jeremy Fitzhardinge <jeremy@goop.org>
13 Chris Wright <chrisw@sous-sol.org>
14 Alok Kataria <akataria@vmware.com>
15 Rusty Russell <rusty@rustcorp.com.au>
16 Tim Waugh <tim@cyberelk.net>
17 "James␣E.␣J.␣Bottomley" <jejb@parisc-linux.org>
18 Helge Deller <deller@gmx.de>
19 Jim Cromie <jim.cromie@gmail.com>
20 ...
```

Listing 4.3: Auszug der Maintainer-Datei von Version 4.7.

Da diese Auflistungen noch nicht vereinheitlicht ist und noch viele Entwickler mehrere Einträge mit unterschiedlichen E-Mail Adressen haben, müssen diese vor der Analyse noch weiter aufbereitet werden. So müssen zum Beispiel die Einträge vom Entwickler `Olof Johansson` (Zeilen 3 und 4) zu einem zusammengefasst werden. Anführungszeichen (Zeile 17) bei Namen oder die spitzen Klammern bei den E-Mail Adressen sind ebenfalls noch zu entfernen.
Dazu wurde ein kleines Java-Programm implementiert, dass die unstrukturierten Dateien einlest und jede Zeile in Name und E-Mail Adresse aufspaltet. Um zu vermeiden, dass ein Entwickler in mehrere Zeilen auftaucht und dadurch bei der Analyse eine Person als zwei unterschiedliche Entwickler betrachtet wird, müssen diverse Einträge zusammengefasst werden. Durch Iterieren und Überprüfen, ob die E-Mail Adresse beziehungsweise der Name eines Entwicklers bereits vorhanden ist, werden Duplikate entfernt [WSS+16]. Enthält ein Duplikat eine noch fehlende Information, wird diese bei dem bereits vorhandenen Eintrag hinzugefügt.
Weitere Informationen, wie das Datum und die Version an dem der Entwickler zum ersten mal Maintainer wurde, werden ebenfalls hinzugefügt. Dazu wird die zuvor generierte `Releases.csv`-Datei eingelesen und anhand des Dateinamens der jeweiligen Maintainer-Liste die Version gematcht. So können die Felder `first_time_maintainer_release` und `first_time_maintainer_date` gefüllt werden. Die generierte CSV-Datei, in der alle Maintainer von allen Versionen enthalten sind, kann in der folgenden Tabelle eingesehen werden.

So wurde zum Beispiel der Entwickler `enver hasse` zum ersten Mal in der Version v2.6.12 ein Maintainer. Das Programm hat also in der Datei `maintainer.v2.6.12`

Tabelle 4.2: Auszug der generierten Liste von Maintainern am Linux-Kernel.

name	**email**	**first time maintainer date**	**first time maintainer release**
enver haase	a2232@gmx.net	2005/06/17 19:48:29	v2.6.12
adaptec oem raid solutions	aacraid@adaptec.com	2007/04/26 03:08:32	v2.6.21
alex elder	ambx1@neo.rr.com	2005/06/17 19:48:29	v2.6.12
	aelder@sgi.com | elder@inktank.com | elder@kernel.org	2009/12/03 03:51:21	v2.6.32
...	...	...	...

den Namen oder die E-Mail Adresse von `enver haase` gefunden und die fehlende Information in der Maintainer-Auflistung gefüllt. Falls die Angabe des Namens eines Entwicklers nicht gefunden wird, bleibt das Feld `name` in der jeweiligen Zeile frei. Dieses wird eventuell später bei der E-Mail Analyse gefüllt, wenn eine Nachricht des Maintainers in der mbox enthalten ist und das Namensfeld in der E-Mail gefüllt ist (zum Beispiel bei der E-Mail Adresse `ambx1@neo.rr.com`). Sollte ein Maintainer mehrere E-Mail Adressen oder Namen verwenden, werden diese durch eine Pipe | getrennt und im jeweiligen Feld aufgelistet. Dies ist zum Beispiel bei `alex elder` der Fall.

Da noch wichtige Informationen, wie die Namen und Pfade der Subsysteme, für die der Entwickler verantwortlich war als er das erste mal Maintainer wurde, fehlen, werden diese im nächsten Schritt hinzugefügt. Ein kleines Kommandozeilen Programm iteriert über alle Versionen des Linux-Kernels und führt das Perl-Skript `get_maintainer.pl` im Repository aus. Dieses generiert für jeden Pfad in einem Subsystem eine Zeile im Terminal mit dem Namen, der E-Mail Adresse des Maintainers, den Namen und den Pfad des Subsystems. Das Listing 4.4 zeigt einen kurzen Auszug der Ausgabe im Terminal.

```
1  ...
2  HEAD is now at 9ee1c93... Linux 2.6.12
3  get_maintainer.pl script not found.
4  git checkout v2.6.13
5  ...
6  git checkout v2.6.34
7  Checking out files: 100% (11171/11171), done.
8  Previous HEAD position was 60b341b... Linux 2.6.33
9  HEAD is now at e40152e... Linus 2.6.34
10 ...
11 Peter Zijlstra <a.p.zijlstra@chello.nl>: arch/sh/kernel/cpu/sh4a/
       perf_event.c
12 Peter Zijlstra <a.p.zijlstra@chello.nl>: arch/x86/kernel/cpu/
       perf_event.c
13 Steven Rostedt <rostedt@goodmis.org>: arch/arm/kernel/ftrace.c
14 Steven Rostedt <rostedt@goodmis.org>: arch/blackfin/kernel/ftrace.c
15 Steven Rostedt <rostedt@goodmis.org>: arch/ia64/kernel/ftrace.c
16 Steven Rostedt <rostedt@goodmis.org>: arch/microblaze/kernel/ftrace.c
17 Steven Rostedt <rostedt@goodmis.org>: arch/mips/kernel/ftrace.c
18 ...
19 git checkout v4.7
20 Checking out files: 100% (9922/9922), done.
21 Previous HEAD position was 2dcd0af... Linux 4.6
22 HEAD is now at 523d939... Linux 4.7
23 ...
24 Robert Jarzmik <robert.jarzmik@free.fr> (maintainer:PXA GPIO DRIVER):
       drivers/gpio/gpio-pxa.c
25 William Breathitt Gray <vilhelm.gray@gmail.com> (maintainer:
       WINSYSTEMS WS16C48 GPIO DRIVER): drivers/gpio/gpio-ws16c48.c
26 David Airlie <airlied@linux.ie> (maintainer:DRM DRIVERS): drivers/gpu
       /drm/
27 Alex Deucher <alexander.deucher@amd.com> (supporter:RADEON and AMDGPU
       DRM DRIVERS): drivers/gpu/drm/amd/
28 ...
29 James Hartley <james.hartley@imgtec.com> (maintainer:PISTACHIO SOC
       SUPPORT): arch/mips/configs/pistachio_defconfig
30 James Hartley <james.hartley@imgtec.com> (maintainer:PISTACHIO SOC
       SUPPORT): arch/mips/include/asm/mach-pistachio/
31 James Hartley <james.hartley@imgtec.com> (maintainer:PISTACHIO SOC
       SUPPORT): arch/mips/pistachio/
32 ...
```

Listing 4.4: Generierte Ausgabe des Perl-Skripts `get_maintainer.pl`

In den früheren Versionen des Linux-Kernels (zum Beispiel v2.6.12) gab es das Pearl-Skript, zum Auflisten der Subsystem Pfad Informationen, noch nicht. In diesem Fall überspringt das Programm den Release und die nächsten Version wird ausgecheckt (Zeile 2 bis 4). Die Informationen zu einem Subsystem-Pfad können auch unvollständig sein. So kann zum Beispiel der Name des Subsystems fehlen (Zeile 13 bis 17). In den neueren Releases sind die Informationen meistens vollständig. Die Zeilen 19 bis 30 zeigen einen Auszug von Version v4.7. Der Entwickler `James Hartley` mit der E-Mail Adresse `james.hartley@imgtec.com` ist in Version v4.7 Maintainer im Subsystem PISTACHIO SOC SUPPORT und für die Pfade `arch/-mips/configs/pistachio_defconfig`, `arch/mips/include/asm/mach-pistachio`

und `arch/mips/pistachio` verantwortlich.

Da diese Auflistung in dieser Formatierung nicht weiterverarbeitet werden kann, muss diese ebenfalls aufbereitet werden. Ein JAVA-Programm, dass als Input die zuvor generierte Maintainer CSV-Datei und die generierte Ausgabe des Shell-Skripts verwendet, erzeugt eine Ausgabe wie in der folgenden Tabelle.
Das Programm iteriert über die Maintainer CSV-Datei und durchsucht die Subsystem-Auflistung. Dabei erkennt es anhand des eines git-Checkout-Befehls (`git checkout`) die Version. Stimmt die E-Mail Adresse oder der Name eines Entwicklers aus der Maintainer-Datei mit einem Eintrag in der Subsystem-Auflistung zusammen, werden die fehlenden Informationen hinzugefügt. So wurde zum Beispiel die Zeile vom Entwickler `alan ott` in der Maintainer CSV-Datei um die Informationen, dass er für das Subsystem MRF24J40 IEEE 802... mit dem Pfad `drivers/net/ieee802154/mrf24j40.c` verantwortlich war, als er das erste mal Maintainer beim Release v3.17 wurde.
Ist ein Entwickler für mehrere Subsystem Pfade verantwortlich, wird, wie bereits beim Namen, oder der E-Mail Adresse, ein eindeutiges Trennzeichen | verwendet. Dies ist zum Beispiel beim Maintainer `davidlohr bueso` der Fall. Er ist im Subsystem GUID PARTITION TA... für die Pfade `block/partitions/efi.c` und `block/partitions/efi.h` verantwortlich.
Fehlt eine Information, wie zum Beispiel der Name des Subsystems beim Entwickler `dustin kirkland`, bleibt das Feld frei.
Um einen Überblick über die Maintainer-Aktivität eines Entwicklers zu bekommen, wird eine zusätzliche Spalte `all releases as maintainer` hinzugefügt. Dieses Feld enthält alle Versionsnummern bei denen der jeweilige Entwickler Maintainer ist. Aus Platzgründen wurde diese Spalte in der vorher erwähnten Tabelle nicht aufgelistet.

Tabelle 4.3: Auszug der aufbereiteten Liste von Maintainern am Linux-Kernel.

name	**email**	**first time maintainer date**	**first time maintainer release**	**first time subsystem name**	**first time maintainer paths**
alan ott	alan@signal11.us	Sun Oct 05 21:23:04 CEST 2014	v3.17	MRF24J40 IEEE 802...	drivers/net/ieee802154 /mrf24j40.c
alexander clouter	alex@digriz.org.uk	Thu May 19 06:06:34 CEST 2011	v2.6.39	ARM/Orion SoC/Tec...	arch/arm/mach-orion5x /ts78xx-fpga.h | arch/arm/mach-orion5x /ts78xx-setup.c
ariel elior	ariele@broadcom.com | ariel.elior@qlogic.com	Mon Jan 20 03:40:07 CET 2014	v3.13	BROADCOM BNX2X 10...	drivers/net/ethernet /broadcom/bnx2x/
davidlohr bueso	dave@stgolabs.net | davidlohr@hp.com	Mon Jan 20 03:40:07 CET 2014	v3.13	GUID PARTITION TA...	block/partitions/efi.c | block/partitions/efi.h
dimitri sivanich	sivanich@sgi.com	Mon Jul 01 00:13:29 CEST 2013	v3.10	SGI GRU DRIVER	drivers/misc/sgi-gru | drivers/misc/sgi-gru/
dustin kirkland	dustin.kirkland@ gazzang.com | kirkland@canonical.com	Thu Sep 10 00:13:59 CEST 2009	v2.6.31		fs/ecryptfs | fs/ecryptfs/
eddie wai	eddie.wai @broadcom.com	Mon Nov 04 00:41:51 CET 2013	v3.12	BROADCOM BNX2I 1/...	drivers/scsi/bnx2i/
gabriel krisman bertazi	krisman@linux .vnet.ibm.com	Mon May 16 00:43:13 CEST 2016	v4.6	JSM Neo PCI based serial card	drivers/tty/serial/jsm/
...	...	...	...	...	...

4.3 Mailingliste

Um die Kommunikation zwischen den einzelnen Entwicklern und Maintainern analysieren zu können, wird die mbox-Datei benötigt. Diese enthält alle, im Zeitraum von Januar 1996 bis August 2016, gesendeten E-Mails an die Mailingliste, die die Entwicklung des Linux-Kernels betreffen. Dabei sind die standard-E-Mail- und weitere Programm-, beziehungsweise Provider spezifische-, Header enthalten.
Die mbox-Datei generiert sich aus einem kleinen Python-Skript, dass auf den gmane-Server zugreift und von dort alle Nachrichten herunterlädt. Der eigentliche Inhalt der E-Mail Nachrichten wird bei dieser Analyse nicht berücksichtigt.
In der nachfolgenden Auflistung werden nur die für die Aufbereitung benötigten Header aufgelistet und erklärt.

Tabelle 4.4: Wichtige E-Mail Header zur Aufbereitung [NWGPR08].

Header	**Beschreibung**
From	Beinhaltet den Absender mit E-Mail Adresse und optional seinen Namen.
To	Enthält die E-Mail Adressen der direkten Empfänger.
Cc	In diesem Feld stehen die Adressen der Empfänger die eine Kopie der Nachricht erhalten sollen.
Date	Der Zeitpunkt an dem die Nachricht gesendet wurde.
Message-ID	Eindeutige ID der E-Mail. Bezogen auf die Linux-Mailing-List.
Subject	Betreff der Nachricht.

Der folgende Auszug aus der mbox-Datei zeigt den Aufbau einer Nachricht mit den dazugehörigen Headern und dem Inhalt der E-Mail. Unwichtige Stellen wurden gekürzt um eine bessere Übersicht zu erhalten.

```
1  From MAILER-DAEMON Wed Aug 10 17:36:58 2016
2  Path: news.gmane.org!not-for-mail
3  From: Ingo Molnar <mingo@kernel.org>
4  Newsgroups: gmane.linux.kernel.next,gmane.linux.kernel
5  Subject: Re: [PATCH] [v3] x86, pkeys: fix siginfo ABI breakage from
       new field
6  Date: Tue, 1 Mar 2016 08:40:52 +0100
7  Lines: 69
8  Approved: news@gmane.org
9  Message-ID: <20160301074052.GA7201@gmail.com>
10 References: <20160229221733.DC2C56B7@viggo.jf.intel.com>
11 NNTP-Posting-Host: plane.gmane.org
12 Mime-Version: 1.0
13 Content-Type: text/plain; charset=us-ascii
14 X-Trace: ger.gmane.org 1456818078 25933 80.91.229.3 (1 Mar 2016
       07:41:18 GMT)
15 X-Complaints-To: usenet@ger.gmane.org
16 NNTP-Posting-Date: Tue, 1 Mar 2016 07:41:18 +0000 (UTC)
17 Cc: linux-kernel@vger.kernel.org, dave.hansen@linux.intel.com,
18     sfr@canb.auug.org.au, akpm@linux-foundation.org,
19     tglx@linutronix.de, mingo@elte.hu, hpa@zytor.com,
20     peterz@infradead.org, linux-next@vger.kernel.org, deller@gmx.de
21 To: Dave Hansen <dave@sr71.net>
22 Original-X-From: linux-next-owner@vger.kernel.org Tue Mar 01 08:41:00
       2016
23 Return-path: <linux-next-owner@vger.kernel.org>
24 Envelope-to: glkn-linux-next@plane.gmane.org
25 Original-Received: /.../
26 Original-Received: /.../
27 DKIM-Signature: /.../
28 X-Google-DKIM-Signature: /.../
29 X-Gm-Message-State: /.../
30 X-Received: /.../
```

Listing 4.5: Auszug der Beispiel E-Mail, Teil 1

```
1  Original-Received: /.../
2  Content-Disposition: inline
3  In-Reply-To: <20160229221733.DC2C56B7@viggo.jf.intel.com>
4  User-Agent: Mutt/1.5.23 (2014-03-12)
5  Original-Sender: linux-next-owner@vger.kernel.org
6  Precedence: bulk
7  List-ID: <linux-next.vger.kernel.org>
8  X-Mailing-List: linux-next@vger.kernel.org
9  Xref: /.../
10 Archived-At: /.../
11
12
13 * Dave Hansen <dave@sr71.net> wrote:
14
15 >
16 > This responds to the feedback from Ingo that we should be using
17 > explicitly-sized types and fixes a typo in the patch description.
18 >
19 > ---
20 >
21 > From: Dave Hansen <dave.hansen@linux.intel.com>
22 >
23
24 /.../
25
26 This checking mechanism could then be extended to other user ABI
      definitions as
27 well, such as include/uapi/linux/perf_event.h.
28
29 Thanks,
30
31     Ingo
```

Listing 4.6: Auszug der Beispiel E-Mail, Teil 2

Die Beispiel E-Mail wurde von `Ingo Molnar` verfasst und von seiner Kernel Adresse `mingo@kernel.org` verschickt (`From`).
Die Nachricht hat den Betreff `Re: [PATCH] [v3] x86, pkeys: fix siginfo ABI breakage from new field` (`Subject`). Re: bedeutet, dass diese E-Mail eine Antwort auf eine andere Nachricht ist. Der zweite Teil des Betreffs zeigt den betroffenen Patch. Gefolgt von einer genauen Beschreibung was geändert wurde.
Im Header `Date` befindet sich der genaue Zeitpunkt an dem die Nachricht versendet wurde. Die Beispiel E-Mail wurde am Dienstag, den 1. März 2016 um 08:40:52 UTC+1h gesendet.
Das Feld `Message-ID` liefert einen eindeutigen String innerhalb der gmane-Linux-Datenbank. Die ID der Beispiel E-Mail ist `20160301074052.GA7201@gmail.com`. Der erste Teil ist bei dieser E-Mail der Timestamp der gesendeten Nachricht. Eine Message-ID endet immer mit einem @ gefolgt von einem Provider (zum Beispiel gmail.com).
Andere Header, wie zum Beispiel Path, Newsgroup, Original-Received, ... sind für die Aufbereitung und anschließende Analyse nicht von Bedeutung. Nach allen Headern folgt die eigentliche Nachricht, welche ebenfalls nicht in der Analyse berücksichtigt wird.

Um eine vernünftige statistische Auswertung der E-Mail Kommunikation durchführen zu können, müssen die Nachrichten aufbereitet werden. Dazu wurde ein kleines Java-Programm entwickelt, welches über die Nachrichten Iteriert, die benötigten Informationen sammelt und in eine CSV-Datei schreibt. Als Input verwendet das Java-Programm mehrere CSV-Dateien.

- **Maintainer_all.csv**
  Eine davon ist die `all.csv`. Diese Datei beinhaltet alle Maintainer von allen Versionen (v2.6.12 bis v4.7) mit deren Namen (`name`), E-Mail Adressen (`email`), dem Datum an dem die jeweilige Person das erste mal Maintainer war (`first time maintainer date`), Version bei der er das erste mal Maintainer war (`first time maintainer release`), den Namen (`first time subsystem name`) und Pfad (`first time maintainer paths`) für den die Person, als sie das erste mal Maintainer war, verantwortlich war und alle Versionen bei denen er Maintainer war (`all releases as maintainer`).
  Diese Datei wird benötigt um festzustellen ob der Absender einer Nachricht zum Zeitpunkt des Sendens Maintainer war oder nicht. Ein weiterer Grund ist, dass bei der mbox-Aufbereitung eine eindeutige ID für jeden Entwickler generiert wird und eine neue Version der `Maintainer_id.all` CSV-Datei mit der neuen Spalte (`maintainer_id`) erstellt wird.

- **Releases.csv** Diese Datei beinhaltet zwei Spalten. Die Erste listet dabei das Startdatum der jeweiligen Version (`date`) wobei die zweite Spalte die genaue Versionsnummer beinhaltet (`release`).
  Um eine E-Mail einem bestimmten Release zuordnen zu können, wird diese Datei benötigt.

- **mbox-Dateien** Da eine große mbox (ca. 16 GB) für die Aufbereitung zu unhandlich ist, wurde die Datei in mehrere kleine mboxen gesplittet. Dabei wurde nach Monat und E-Mail Adresse aufgeteilt.

Nach erfolgreichem Einlesen der zusätzlich benötigten Dateien (`Maintainer_all.csv` und `Releses.csv`) wird über jede Nachricht iteriert. Im ersten Schritt der Aufbereitung wird zunächst überprüft, ob dem Entwickler bereits eine ID zugeteilt wurde. Dies geschieht durch einen Vergleich der E-Mail Adressen. Falls es zu keiner Übereinstimmung kommt, wird der Name verglichen. Gibt es ebenfalls keine Übereinstimmung handelt es sich um einen neuen Entwickler und das Programm vergibt eine neue ID.
Anschließend wird die Versionsnummer vom Linux-Kernel zum Zeitpunkt des sendens der E-Mail hinzugefügt. Dafür wird über die `Releases.csv`-Datei iteriert und der passende Relese anhand des Datums ausgelesen.
Um herauszufinden, ob die E-Mail von einem Maintainer stammt, wird über die `Maintainer_all.csv` iteriert. Falls es eine Übereinstimmung anhand von dem Namen oder anhand der E-Mail Adresse gibt [WSS+16] und das Versanddatum der Nachricht nach dem Zeitpunkt an dem die Person das erste mal Maintainer wurde ist, handelt es sich bei dem E-Mail-Verfasser um einen Maintainer. Bei diesem Vorgehen werden auch die Identifier, die beim Auslesen der E-Mails generiert wurden, den passenden Maintainern zugeordnet.
Um eine Unterscheidung zwischen privaten und beruflichen Entwicklern beim Linux-Kernel zu gewährleisten, wurden die beliebtesten privaten E-Mail Provider (siehe Tabelle Private E-Mail-Anbieter) bei der Aufbereitung berücksichtigt. Sollte ein Absender auf einen solchen Anbieter nutzen, wird in der Output-Datei der Wert im Feld `private_email_provider` entsprechend gesetzt.

Tabelle 4.5: Beliebte private E-Mail Anbieter [Bis17][Coq]

Anbieter	**E-Mail Adressen Endung**
Apple iCloud	@icloud.com
Apple Mobile Me	@me.com
Google Mail	@gmail.com, @googlemail.com
Microsoft Outlook	@outlook.com
Microsoft Hotmail	@hotmail.com
Yahoo Mail	@yahoo.com
AOL	@aol.com
Zoho	@zoho.com
Mail.com	@mail.com
Yandex	@yandex.com
Protonmail	@protonmail.com, @protonmail.ch
GMX	@gmx.de, @gmx.net
Web.de	@web.de
T-Online	@t-online.de

Dies geschieht auch für die Top-Companies. Personen, die über eine berufliche E-Mail Adresse beim Linux-Kernel mitwirken und einer Top-Company angehören, wird

ein Feld mit dem Titel `top_company` dementsprechend gesetzt. Eine Liste von Top-Companies wurde dabei vom Linux-Kernel Development Report Dezember 2010 entnommen.

Tabelle 4.6: Top-Companies [Fou10]

Firmenname	**E-Mail Adressen Endung**
Red Hat	@redhat.com
Novell	@novell.com
IBM	@ibm.com
Intel	@intel.com
Oracle	@oracle.com
Renesas Technology	@renesas.com
The Linux Foundation	@linux.org
SGI	@sgi.com
Fujitsu	@fujitsu.com
Parallels	@parallels.com
Nokia	@nokia.com
Google	@google.com
Astaro	@astaro.com
Broadcom	@broadcom.com
QLogic	@qlogic.com
NTT	@ntt.co.jp

Um Entwickler von Bildungseinrichtungen unterscheiden zu können, wird auf die E-Mail-Endung .edu geprüft. Sollte ein Absender eine solche Adresse verwenden, wird der Eintrag `is_edu` mit true gefüllt.
Bestimmte Maintainer haben einen höheren Rang als andere. Diese werden als Top-Maintainer bezeichnet. Um einen Zusammenhang zwischen der Kommunikation mit Top-Maintainern und dem Erreichen des Maintainer-Status analysieren zu können, wird bei jeder Nachricht überprüft, ob diese direkt und nur an einen Top-Maintainer gerichtet ist. Die berücksichtigten Top-Maintainer werden in der folgenden Tabelle aufgelistet.

Tabelle 4.7: Top-Maintainer [Fou10]

Name	**E-Mail Adresse/n**
Linus Torvalds	torvalds@linux-foundation.org
Andrew Morton	akpm@linux-foundation.org
Greg Kroah-Hartman	gregkh@suse.de, greg@kroah.com, gregkh@linuxfoundation.org
Al Viro	viro@zeniv.linux.org.uk
Ingo Molnar	mingo@elte.hu, mingo@redhat.com, mingo@kernel.org
David S. Miller	davem@davemloft.net
Takashi Iwai	tiwai@suse.com, tiwai@suse.de
Bartlomiej Zolnierkiewicz	b.zolnierkie@samsung.com, b.zolnierkiewicz@elka.pw.edu.pl

Die Liste von Top-Maintainern wurde dabei aus dem Linux-Kernel-Report entnommen. Diese Maintainer tragen am meisten zu der Entwicklung des Linux Kernels bei.

Um herauszufinden ob das Aufsteigen zu einem Maintainer mit dem Helfen beziehungsweise Antworten auf E-Mails zusammenhängt, wird der Betreff der Nachrichten analysiert. Beginnt der Betreff mit `aw, wg, fw` oder `re` handelt es sich bei der Nachricht um eine Antwort beziehungsweise um eine Weiterleitung und somit um eine unterstützende E-Mail. Anderenfalls ist die Nachricht eine initiale E-Mail und das Feld `initial_mail` wird auf true gesetzt. Bei diesem Schritt wird auch das Feld `thread` mit dem Betreff der Nachricht ohne Prefix gefüllt. Dies wird benötigt um mehrere Nachrichten einer Unterhaltung zuordnen zu können. Betrachtet man beispielsweise folgende drei Betreffe `[PATCH] fbdev: colormap fixes, Re: [PATCH] fbdev: colormap fixes` und `Re: Re: [PATCH] fbdev: colormap fixes`. Diese hätten den Thread `[PATCH] fbdev: colormap fixes` und können somit durch matchen des Feldes `thread` einer Unterhaltung zugeordnet werden. Die Nachricht mit dem `Betreff [PATCH] fbdev: colormap fixes` wäre dabei die initial E-Mail.
Zur Vollständigkeit werden sämtlich auslesbaren Felder der E-Mail ebenfalls in die Ausgabe-Datei angefügt. Counter für zählbare E-Mail-Attribute werden ebenfalls ergänzt.
Das Java-Programm zur Aufbereitung der mbox-Datei liefert mehrere Ausgabe-Dateien. Die wichtigste ist dabei eine Auflistung aller Nachrichten mit den zuvor beschriebenen zusätzlichen Informationen. Dabei handelt es sich um eine CSV-Datei (`emailRecords.csv`) bei der jede Zeile eine E-Mail beschreibt. Die folgende Tabelle liefert einen kleinen Auszug.

Tabelle 4.8: Auszug der aufbereiteten mbox-Datei, Teil 1.

message_id	developer_id	timestamp	developer_name	developer_email	release
472F6784.9080900@gmx.net	1	Mon Nov 05 19:57:08 CET 2007	Enver Haase	a2232@gmx.net	v2.6.23
57A06420.3070508@oracle.com	2	Tue Aug 02 11:13:04 CEST 2016	Vegard Nossum	vegard.nossum@oracle.com	v4.7
20160801130655.15004-1-vigneshr@ti.com	3	Mon Aug 01 15:06:55 CEST 2016	Vignesh R	vigneshr@ti.com	v4.7
87wpjrob24.fsf@vitty.brq.redhat.com	4	Mon Aug 08 11:38:11 CEST 2016	Vitaly Kuznetsov	vkuznets@redhat.com	v4.7
1470225791-16613-4-git-send-email-wagi@monom.org	5	Wed Aug 03 14:03:10 CEST 2016	Daniel Wagner	wagi@monom.org	v4.7
...	...	...	...	...	...

Tabelle 4.9: Auszug der aufbereiteten mbox-Datei, Teil 2.

message_id	private_mail_provider	top_company	is_edu	mail_to_torvalds	...	initial_mail	maintainer	count_cc
472F6784.9080900@gmx.net	1	0	0	0	...	0	1	2
57A06420.3070508@oracle.com	0	1	0	0	...	0	1	5
20160801130655.15004-1-vigneshr@ti.com	0	0	0	0	...	1	0	7
87wpjrob24.fsf@vitty.brq.redhat.com	0	1	0	0	...	0	0	3
1470225791-16613-4-git-send-email-wagi@monom.org	0	0	0	0	...	1	0	10
...	...	...	...	...	...	...	...	...

Tabelle 4.10: Auszug der aufbereiteten mbox-Datei, Teil 3.

messageId	count_to	...	subject	thread
<472F6784.9080900@gmx.net>	1	...	Re: [PATCH] ser_a2232.c bug ...	[patch] ser_a2232.c bug ...
<57A06420.3070508@oracle.com>	1	...	Re: Hang in 9p/virtio	hang in 9p/virtio
<20160801130655.15004-1-vigneshr@ti.com>	1	...	ARM: dts: dra7xx: ...	arm: dts: dra7xx: ...
<87wpjrob24.fsf@vitty.brq.redhat.com>	1	...	Re: [PATCH 2/4] Drivers: hv ...	drivers: hv ...
<1470225791-16613-4-git-send-email-wagi@monom.org>	1	...	i2c: brcmstb: Use ...	i2c: brcmstb: use ...
...	...	...	...	...

Aus Platzgründen wurden bei den drei vorherigen Tabellen einige Spalten weggelassen. So wurde zum Beispiel auf die Auflistung aller Spalten, ob eine Nachricht direkt an diverse Top-Maintainer gerichtet ist, nicht aufgelistet.

Des weiteren wird eine neue Version der `Maintainer.csv`-Datei erstellt. Der Unterschied zur Input-Datei liegt darin, dass ein zusätzliches Feld mit der bei der Analyse generierten Developer-ID hinzugefügt wurde. Die nachfolgende Tabelle zeigt einen kurzen Ausschnitt dieser neu generierten Datei.

Tabelle 4.11: Neu generierte `Maintainers_all.csv`-Datei mit IDs

maintainer_id	**maintainer_name**	**maintainer_email**	**...**
74459a75-293a-46a0-8a2f-943aaf4aaf0a	enver haase	a2232@gmx.net	...
4f4e8653-2f43-401f-90e5-1b092fbb5c51	antonino daplas	adaplas@gmail.com	...
df794aef-d0a3-4d6a-aa44-a460aa480ce9	mimi zohar	zohar@linux.vnet.ibm.com | zohar@us.ibm.com	...
193bc336-9332-4208-a0aa-6373397e45bd	ross zwisler	ross.zwisler@linux.intel.com	...
...	...	...	...

Aus Platzgründen wurden hier ebenfalls nur die wichtigsten Spalten abgebildet. Die weggelassenen Informationen sind identisch mit der Input `Maintainer_all.csv`-Datei.

Die letzte Ausgabe-Datei ist die `Developer.csv`. Sie enthält alle Entwickler mit deren IDs, Namen und E-Mail Adressen, die eine E-Mail an die Mailingliste geschrieben haben.

Tabelle 4.12: Auszug aus der generierte `Developer.csv`-Datei

developer_id	**developer_name**	**developer_email**
1	Antonino A. Daplas	adaplas@gmail.com
2	Enver Haase	a2232@gmx.net
3	Vladislav Bolkhovitin	vst@vlnb.net
4	Willy Tarreau	w@1wt.eu
...	...	...

Eine weitere interessante Information ist die Antwortzeit auf eine initiale E-Mail. Sendet ein Entwickler eine E-Mail an die Mailingliste, dauert es einen bestimmte Zeit, bis jemand darauf antwortet. Diese Zeit wird durch ein Java-Programm generiert, welches die `commitRecords.csv`-Datei als Input verwendet. Dieses Programm iteriert über alle Einträge und vergleicht dabei die Spalten `subject` und `thread`. Bei der Spalte `subject` handelt es sich um den Betreff der E-Mail. `thread` ist der Betreff der E-Mail ohne Antwort oder Weiterleitungs-Prefix. Dadurch kann die erste Antwort auf eine E-Mail gefunden werden und durch die Timestamp-Header die Antwortzeit ausgerechnet werden.
Eine Beispielausgabe des Programms kann in der Tabelle 4.13 und folgende eingesehen werden.

Tabelle 4.13: Antwortzeit auf initiale E-Mails, Teil 1.

subject	initial_message_id	initial_developer_id	initial_sender
Primary ide disk inaccessible. 2.6.11.7, x86_64	426B8D14.9050408@lboro.ac.uk	21	A.E.Lawrence
[PATCH][RFC] Linux VM hooks for advanced RDMA NICs	426E62ED.5090803@quadrics.com	53	David Addison
[PATCH -mm] Tell users to install xmlto, not stylesheets	20050430113519.GA9309@mipter.zuzino.mipt.ru	73	Alexey Dobriyan
[PATCH] 2.6.11.7 X.25 : x25tap	1114576764.4789.36.camel@localhost.localdomain	88	Andrew Hendry
New debugging patch was Re: x86-64 bad pmds in 2.6.11.6 II	20050427142343.GN13305@wotan.suse.de	145	Andi Kleen
2.6.12-rc3-mm1	20050429231653.32d2f091.akpm@osdl.org	202	Andrew Morton
2.6.12-rc3-mm2	20050430164303.6538f47c.akpm@osdl.org	202	Andrew Morton
IRQ Disabling	1114453001.19173.47.camel@morte.jpl.nasa.gov	589	Al Niessner
Multiple functionality breakages in 2.6.12rc3 IDE layer	1114703284.18809.208.camel@localhost.localdomain	600	Alan Cox
IDE problems with rmmod ide-cd	1114706653.18330.212.camel@localhost.localdomain	600	Alan Cox

Tabelle 4.14: Antwortzeit auf initiale E-Mails, Teil 2.

subject	initial_email	initial_timestamp	first_response_message_id
Primary ide disk inaccessible. 2.6.11.7, x86_64	A.E.Lawrence@lboro.ac.uk	Sun Apr 24 14:12:04 CEST 2005	426BC4C9.8060403@lboro.ac.uk
[PATCH][RFC] Linux VM hooks for advanced RDMA NICs	addy@quadrics.com	Tue Apr 26 17:49:01 CEST 2005	4270AB1E.3090000@quadrics.com
[PATCH -mm] Tell users to install xmlto, not stylesheets	adobriyan@mail.ru	Sat Apr 30 13:35:19 CEST 2005	20050430200407.GC8402@mars.ravnborg.org
[PATCH] 2.6.11.7 X.25 : x25tap	ahendry@tusc.com.au	Wed Apr 27 06:39:24 CEST 2005	1114733031.3725.5.camel@localhost.localdomain
New debugging patch was Re: x86-64 bad pmds in 2.6.11.6 II	ak@suse.de	Wed Apr 27 16:23:44 CEST 2005	20050427173704.GC19011@redhat.com
2.6.12-rc3-mm1	akpm@osdl.org	Sat Apr 30 08:16:53 CEST 2005	20050430033818.06b94e26.akpm@osdl.org
2.6.12-rc3-mm2	akpm@osdl.org	Sun May 01 01:43:03 CEST 2005	20050430173724.3189c50f.akpm@osdl.org
IRQ Disabling	Al.Niessner@jpl.nasa.gov	Mon Apr 25 20:16:41 CEST 2005	1114517101.18330.73.camel@localhost.localdomain
Multiple functionality breakages in 2.6.12rc3 IDE layer	alan@lxorguk.ukuu.org.uk	Thu Apr 28 17:48:05 CEST 2005	1114727182.24687.235.camel@localhost.localdomain
IDE problems with rmmod ide-cd	alan@lxorguk.ukuu.org.uk	Thu Apr 28 18:44:14 CEST 2005	1114725062.18809.226.camel@localhost.localdomain

Tabelle 4.15: Antwortzeit auf initiale E-Mails, Teil 3.

subject	first_response _developer_id	first_response_sender	first_response_email	first_response_timestamp
Primary ide disk inaccessible. 2.6.11.7, x86_64	21	A.E.Lawrence	A.E.Lawrence@lboro.ac.uk	Sun Apr 24 18:09:45 CEST 2005
[PATCH][RFC] Linux VM hooks for advanced RDMA NICs	53	David Addison	addy@quadrics.com	Thu Apr 28 11:21:34 CEST 2005
[PATCH -mm] Tell users to install xmlto, not stylesheets	6397	Sam Ravnborg	sam@ravnborg.org	Sat Apr 30 22:04:07 CEST 2005
[PATCH] 2.6.11.7 X.25 : x25tap	88	Andrew Hendry	ahendry@tusc.com.au	Fri Apr 29 02:03:51 CEST 2005
New debugging patch was Re: x86-64 bad pmds in 2.6.11.6 II	1786	Dave Jones	davej@redhat.com	Wed Apr 27 19:37:04 CEST 2005
2.6.12-rc3-mm1	202	Andrew Morton	akpm@osdl.org	Sat Apr 30 12:38:18 CEST 2005
2.6.12-rc3-mm2	202	Andrew Morton	akpm@osdl.org	Sun May 01 02:37:24 CEST 2005
IRQ Disabling	600	Alan Cox	alan@lxorguk.ukuu.org.uk	Tue Apr 26 14:05:03 CEST 2005
Multiple functionality breakages in 2.6.12rc3 IDE layer	600	Alan Cox	alan@lxorguk.ukuu.org.uk	Fri Apr 29 00:26:27 CEST 2005
IDE problems with rmmod ide-cd	600	Alan Cox	alan@lxorguk.ukuu.org.uk	Thu Apr 28 23:51:09 CEST 2005

Tabelle 4.16: Antwortzeit auf initiale E-Mails, Teil 4.

subject	minutes	hours	days	period
Primary ide disk inaccessible. 2.6.11.7, x86_64	237	3	0	PT3H57M41S
[PATCH][RFC] Linux VM hooks for advanced RDMA NICs	2492	41	1	P1DT17H32M33S
[PATCH -mm] Tell users to install xmlto, not stylesheets	508	8	0	PT8H28M48S
[PATCH] 2.6.11.7 X.25 : x25tap	2604	43	1	P1DT19H24M27S
New debugging patch was Re: x86-64 bad pmds in 2.6.11.6 II	193	3	0	PT3H13M20S
2.6.12-rc3-mm1	261	4	0	PT4H21M25S
2.6.12-rc3-mm2	54	0	0	PT54M21S
IRQ Disabling	1068	17	0	PT17H48M22S
Multiple functionality breakages in 2.6.12rc3 IDE layer	398	6	0	PT6H38M22S
IDE problems with rmmod ide-cd	306	5	0	PT5H6M55S

Die Spalte `subject` liefert, wie bereits oben schon erklärt, den Betreff einer initialen E-Mail. Die ID der initialen E-Mail wird in der Spalte `initial_message_id` aufgelistet. Die ID des Verfassers der E-Mail befindet sich in der Spalte `initial_developer_id`. `initial_sender` gibt den Namen des Absenders der E-Mail an. Die E-Mail Adresse des Absenders ist in der Spalte `initial_email` vorhanden. Der Zeitpunkt, an dem die E-Mail gesendet wurde, wird in `initial_timestamp` aufgelistet. Die gleichen Informationen werden zu der ersten Antwort E-Mail ebenfalls angegeben. Dabei wir das „initial“ durch „first_response“ ersetzt (`first_response_message_id`, `first_response_developer_id`, `first_response_sender`, `first_response_email` und `first_response_timestamp`). Die Information wie lange es dauert bis die erste Antwort gesendet wurde, wird in den Spalten `minutes` in Minuten, `hours` in Stunden und `days` in Tage angegeben. Dabei ist zu beachten, dass diese drei Spalten unabhängig zu betrachten sind. Die Spalte `period` liefert einen genaue Dauer in Tagen, Stunden, Minuten und Sekunden.
So ist zum Beispiel die E-Mail mit dem Betreff `New debugging patch was Re: x86-64 bad pmds in 2.6.11.6 II` mit der ID `20050427142343.GN13305@wotan.suse.de` vom Entwickler `Andi Kleen` mit der ID `145` mit der E-Mail Adresse `ak@suse.de` am 27.04.2005 um 16:23:44 CEST an die Mailingliste gesendet worden. Die Antwort kam am 27.04.2005 um 19:37:04 CEST an. Der Absender der ersten Antwort ist `Dave Jones` mit der ID `1786`. Seine E-Mail Adresse mit der er antwortete ist `davej@redhat.com`. Die Dauer bis diese Antwort gesendet wurde entspricht dabei `3` Stunden, `13` Minuten und `20` Sekunden.

4.4 Commit-Liste

Da bei der Analyse nicht nur die Kommunikation (E-Mails der Mailingliste), sondern auch der Beitrag an Commits und Veränderungen am Code des Linux-Kernels betrachtetet werden soll, wird die Commit-Liste benötigt. Diese listet alle Commits vom Git-Repository des Linux-Kernels im Zeitraum von 2004 bis 2016 auf. Dabei ist zu beachten, dass die Daten der Commit-Liste heuristisch sind. Die folgende Tabelle zeigt einen kleinen Ausschnitt der Datei.

Tabelle 4.17: Auszug der ursprünglichen Liste von Commits am Linux-Kernel, Teil 1.

commit.id	timestamp	author	author.email	commit.hash	commit .changedFiles	commit .addedLines	commit .deletedLines
921473	Mon Oct 19 15:19:20 CEST 2015	Russell King	rmk@dyn-67 .arm.linux.org.uk	cb0f2538039c65f2 bb64a9d427db...	1	14	0
921426	Mon Oct 19 15:23:48 CEST 2015	Namhyung Kim	namhyung @gmail.com	8719138318316656 988dbd422461...	3	3	3
921426	Mon Oct 19 15:23:48 CEST 2015	Namhyung Kim	namhyung @gmail.com	8719138318316656 988dbd422461...	3	3	3
921426	Mon Oct 19 15:23:48 CEST 2015	Namhyung Kim	namhyung @gmail.com	8719138318316656 988dbd422461...	3	3	3

Tabelle 4.18: Auszug der ursprünglichen Liste von Commits am Linux-Kernel, Teil 2.

commit.id	commit.diffSize	file	function.name	function.type	changeOnFunction
921 473	14	drivers/clocksource /time-armada-370-xp.c	File_Level	Function	6
921 426	6	tools/perf/tests /openat-syscall-all-cpus.c	test__openat_syscall _event_on_all_cpus	Function	1
921 426	6	tools/perf/tests /openat-syscall-tp-fields.c	test__syscall _openat_tp_fields	Function	1
921 426	6	tools/perf/tests /openat-syscall.c	test__openat _syscall_event	Function	1

Jeder Commit, der auf dem Linux-Kernel-Repository gemacht wird, hat eine eindeutige ID. Diese wird vom Versionsverwaltungssystem selbst generiert und im Feld `commit.id` gespeichert. Der genaue Zeitpunkt des Commits wird in der Spalte `timestamp` abgelegt. Die Formatierung ist dabei wie bei allen anderen Dateien die selbe. Die Entwicklerinformationen zum jeweiligen Commit sind der Name (`author`) und die E-Mail Adresse (`author.email`) der Person, die diesen getätigt hat. Das Feld `commit.hash` ist ein berechneter Hash-Wert, der ebenfalls eindeutig für einen Commit ist.
Die nächsten vier Spalten sind Counter. `commit.changedFiles` ist dabei die Anzahl an veränderten Dateien beim jeweiligen Commit. Die hinzugefügten Zeilen bei einem Commit wird im Feld `commit.addedLines` gezählt. Das Gegenstück dazu ist die Anzahl an gelöschten Zeilen (`commit.deletedLines`). Die Spalte `commit.diffSize` summiert die Counter von commit.addedLines und commit.deletedLines einer Zeile in der Commit Liste.
In der nächsten Spalte wird der genaue Pfad zur Datei (`file`), die von dem Commit verändert wurde, angegeben. Um die Veränderung an der Datei noch genauer spezifizieren zu können, wird in einer weiteren Spalte der Name der betroffene Funktion (`function.name`) in der Datei angegeben. Der Typ der betroffenen Funktion (`function.type`) ist ebenfalls in der Commit-Liste gespeichert. In den meisten Fällen ist der Typ `Function`. Die letzte Spalte nennt sich `commit.changeOnFunction` und bezeichnet die Anzahl an Änderungen an der jeweiligen Datei beziehungsweise Funktion. Da jede Datei, die ein Commit verändert, separat einen Eintrag in der Commit-Liste erhält, können mehrere Zeilen zu einem Commit gehören.
Dies ist zum Beispiel beim Commit mit der ID `921426` der Fall. Er wurde am Montag, den 19. Oktober vom Entwickler `Namhyung Kim` getätigt. Seine E-Mail Adresse lautet `namhyung@gmail.com` und der generierte Hash des Commits ist `2690c730935873065175de33f59cce5bb221b9dc`. Die Anzahl an veränderten Dateien ist `3`, es wurden `3` Zeilen hinzugefügt, `3` Zeilen gelöscht und die `diff_Size` beträgt `6` (Summe von hinzugefügten und gelöschten Zeilen). Die Spalten `file`, `function.name`, `function.type` und `changeOnFunction` können sich in jeder Zeile des jeweiligen Commits unterscheiden. So wurde im ersten Eintrag des Commits die Datei `openat-syscall-all-cpus.c` im Ordner `tools/perf/tests/` die Methode `test openat syscall event on all cpus` verändert. Der zweite Eintrag zu diesem Commit beschreibt die Änderungen der Datei `tools/perf/tests/openat-syscall-tp-fields.c`. Im letzten Eintrag werden die Veränderungen zur Datei `tools/perf/tests/openat-syscall.c` beschrieben. In diesem Beispiel sind die Felder `function.type` und `changeOnFunction` gleich. Dies muss allerdings nicht immer der Fall sein.

Um den Zusammenhang zwischen der mbox-Datei und der Maintainer-Liste mit der Commit-Liste herstellen zu können, wird ebenfalls die Spalte mit der `developer_id` benötigt, die bei der mbox-Aufbereitung vergeben wird. Ein kleines Java-Programm verwendet als Input die Commit-Liste und die von der mbox-Aufbereitung erstellte Entwickler-Liste. Letztere wird wegen den generierten Entwickler-IDs benötigt. Durch das iterieren über alle Commiteinträge und vergleichen, ob der Name oder die E-Mail Adresse eines Entwicklers übereinstimmt [WSS+16] wird so zu jedem Autor eines Commits eine ID hinzugefügt. Ist eine E-Mail Adresse oder Name in der Commit-Liste vorhanden, jedoch nicht in der Entwickler-Liste, so erstellt das

Programm für diesen Entwickler eine neue ID.
Als Output der Aufbereitung erhält man zwei Dateien. Zum einen eine neue Commit-Liste mit einer neuen Spalte `developer_id` und veränderten Spaltennamen und zum Anderen eine Auflistung an Entwicklern, die sich in der Commit-Liste, aber nicht in der mbox-Datei befinden.
Bei der neuen Version der Commit-Liste wurde als Trennzeichen bei den Spaltennamen ein Unterstrich _ anstatt eines Punktes verwendet und alle Zeichen klein geschrieben. Die nachfolgende Tabelle zeigt einen kleinen Ausschnitt dieser Datei.

Tabelle 4.19: Auszug der aufbereiteten Liste von Commits am Linux-Kernel, Teil 1.

commit_id	**timestamp**	**developer_id**	**developer_name**	**developer_email**	**commit_hash**	**changed_files**	**added_lines**
921 473	Mon Oct 19 15:19:20 CEST 2015	877d0a20-aa1b -43eb-aa51-90f58f89e6d0	Russell King	rmk@dyn-67 .arm.linux.org.uk	cb0f2538039c65f2 bb64a9d427db...	1	14
921 426	Mon Oct 19 15:23:48 CEST 2015	e7598b07-90f8 -4f3a-b98b-d488243903c6	Namhyung Kim	namhyung @gmail.com	8719138318316656 988dbd422461...	3	3
921 426	Mon Oct 19 15:23:48 CEST 2015	e7598b07-90f8 -4f3a-b98b-d488243903c6	Namhyung Kim	namhyung @gmail.com	8719138318316656 988dbd422461...	3	3
921 426	Mon Oct 19 15:23:48 CEST 2015	e7598b07-90f8 -4f3a-b98b-d488243903c6	Namhyung Kim	namhyung @gmail.com	8719138318316656 988dbd422461...	3	3
...	...	...	...	...	...	...	...

Tabelle 4.20: Auszug der aufbereiteten Liste von Commits am Linux-Kernel, Teil 2.

commit_id	**deleted_lines**	**diff_size**	**file**	**function_name**	**function_type**	**change_on_function**
921 473	0	14	drivers/clocksource /time-armada-370-xp.c	File_Level	Function	6
921 426	3	6	tools/perf/tests /openat-syscall-all-cpus.c	test__openat_syscall _event_on_all_cpus	Function	1
921 426	3	6	tools/perf/tests /openat-syscall-tp-fields.c	test__syscall _openat_tp_fields	Function	1
921 426	3	6	tools/perf/tests /openat-syscall.c	test__openat _syscall_event	Function	1
...	...	...	...	...	...	...

Die Auflistung der Entwickler, die in der Commit-Liste, aber nicht in der mbox-Datei zu finden sind, wird in der Datei `noMboxDev.csv` abgespeichert. Diese Datei enthält pro Entwickler eine ID, die während der Commit-Listen-Aufbereitung erstellt wird, den Namen und die E-Mail Adresse. `noMboxDev.csv` wird benötigt, um abschätzen zu können, welcher Anteil an Entwicklern nur committet und keine E-Mail Kommunikation über den vorgesehenen Kommunikationsweg betreibt. Als kleines Beispiel dieser Datei kann Tabelle 4.21 betrachtet werden.

Tabelle 4.21: Auszug der generierten Liste von Entwicklern, die keine E-Mail an die Mailingliste geschrieben, aber committet haben. Gesamtanzahl 178

id	name	email
1	felix@derklecks.de	felix@derklecks.de
2	Liu Tao	liutao1980@gmail.com
3	Loic Le Loarer	loic.le-loarer+lk@polytechnique.org
4	Tim	tedon@rogers.com
5	Gergely Tamas	dice@mfa.kfki.hu
6	Gareth Howlett	ghowlett@connecttech.com
7	Deti Fliegl	deti@fliegl.de
8	Albrecht+Dre%DF	albrecht.dress@lios-tech.com
9	Ami Perlmutter	amip@mellanox.co.il
10	Amit Arora	aarora@in.ibm.com
11	Axel (XL) Dyks	xl@xlsigned.net
12	Suresh R	sureshr@serverengines.com
13	Curtis Wald	cwald@watchguardvideo.com
14	Luis Maia	lmaia@royalhat.org

Es gibt immer wieder Abweichungen bei Einträgen, wie zum Beispiel beim Entwickler mit der ID `cab2a1d2-9ea4-4abd-ba6d-39c2a37b42af`. Der Name entspricht in diesem Fall seiner E-Mail Adresse `felix@derklecks.de`. Diese Abweichungen kommen zustande, wenn ein Entwickler beim committen oder im E-Mail-Header den Namen falsch setzt.

4.5 Patch-Acceptance-Rate

Wenn ein Entwickler eine Änderung am Code vornimmt, kann es sein, dass der zuständige Maintainer diesen nicht akzeptiert und der Entwickler nachbessern muss. Dazu müssen die betroffenen Personen über eine der Mailinglisten kommunizieren. Die Patch-Acceptance-Rate berechnet sich als Kehrwert der Anzahl an Anläufen, die ein Entwickler benötigt, damit seine Änderungen von einem Maintainer akzeptiert werden. Die Formel sieht wie folgt aus:

$$\text{Patch-Acceptance-Rate} = \frac{1}{\text{Anzahl an Anläufen damit der Patch übernommen wird}} \tag{4.1}$$

Dafür ist eine Zuordnung zwischen den Commits aus dem Repository und den Nachrichten aus der mbox-Datei, die die Patches enthalten, notwendig.
Um diese Zuordnung herzustellen, wird die mbox-Analyse vom Tool `The Patch Stack Analysis (PaStA)` verwendet. PaStA analysiert dabei quantitativ die Evolution von Patch-Stacks eines Repositories. Dabei werden verschiedene Releases von Stacks verglichen und ähnliche Patches gruppiert. Dabei ist zu beachten, dass das Vergleichen der Patches heuristisch ist. Somit erhält man eine Gruppierung von ähnlichen Patches und durch die Reihenfolge in den Stacks eine Übersicht über die Evolution der jeweiligen Patches. PaStA vergleicht im nächsten Schritt die Repräsentaten der jeweiligen Patch-Stacks mit den Patches der Mailingliste, um die Zuordnung vorzunehmen [RR16].
Das Ergebnis ist eine Liste von Commit-Hashes mit den jeweiligen, dazugehörigen Message-IDs der E-Mails aus der mbox-Datei. Durch das Absender-Feld in der E-Mail lässt sich auf den jeweiligen Entwickler rückschließen.
Für das Zuordnen und Gruppieren von Patches benötigt PaStA eine Konfigurationsdatei. Ein Beispiel wird in Listing 4.7 gezeigt.

```
1  [PaStA]
2  ### Project name. Should not contain whitespaces ###
3  PROJECT_NAME = Linux
4
5  ### necessary for upstream comparison. Patches within this range will
       be considered ###
6  UPSTREAM = v2.6.12..v4.8
7
8  ### Blacklist of upstream commit hashes to ignore for upstream
       comparison ###
9  UPSTREAM_BLACKLIST = linux-blacklist
10
11 ### Project specific autothresholds for interactive rating ###
12 INTERACTIVE_THRESHOLD = 0.90
13 AUTOACCEPT_THRESHOLD = 0.90
14 DIFF_LINES_RATIO = 0.70
15
16 ### Used for parsing mailinglists ###
17 MBOX_MINDATE = 2004-06-17
18 MBOX_MAXDATE = 2017-10-02
```

Listing 4.7: PaStA-Konfigurationsdatei für Release v4.4 des Linux-Kernels.

Die Variable `PROJECT_ NAME` legt den Projekt Namen fest. `UPSTREAM` beschränkt die Analyse auf bestimmte Releases. In diesem Fall werden alle Versionen von v2.6.12 bis v4.8 betrachtet. Um bestimmte Commits auszuschließen, kann man in der Variable `UPSTREAM_BLACKLIST` eine Blacklist festlegen. Da PaStA nur einen halbautomatischen Algorithmus verwendet, kann es sein, dass der Benutzer einige Patches von Hand vergleichen muss. Aus diesem Grund werden die Variablen `INTERACTIVE_THRESHOLD`, `AUTOACCEPT_THRESHOLD` und `DIFF_LINES_RATIO` benötigt. Bei dieser Konfiguration wurden die Schwellwerte so gewählt, dass der Abgleich von Hand wegfällt, da die beiden Werte `INTERACTIVE_THRESHOLD` und `AUTOACCEPT_THRESHOLD` identisch sind. Der Schwellenwert, bei dem das Tool automatisch einen Patch zuordnet, ist der `AUTOACCEPT_THRESHOLD`. Eine halbautomatische Zuordnung der Pat-

ches wird beim Schwellwert `INTERACTIVE_THRESHOLD` festgelegt. Die Variablen zum zeitlichen Eingrenzen der E-Mail Nachrichten werden in den letzten beiden Zeilen festgelegt (`MBOX_ MINDATE` und `MBOX_ MAXDATE`).

```
<20080119150737.GB7231@cvg> => 102f4a02de5c7217a04ccbbc24f35224b98bb183
<20080119154453.1afddbfb@lxorguk.ukuu.org.uk>
<20080119155126.180a010d@lxorguk.ukuu.org.uk>
<20080119160345.4160d8a5@lxorguk.ukuu.org.uk> => 9e491e54f0589cc26e2c096664e9d95493b1af29
<20080119160506.5aab8000@lxorguk.ukuu.org.uk> => 5c5a26fa9cd27ed4642f5a4ec76d9b487959e8dc
<20080119160743.GA8352@csn.ul.ie>
<20080119160758.1d2c908a@lxorguk.ukuu.org.uk>
<200801192045.17291.yinghai.lu@sun.com>
<20080119212019.GB29070@uranus.ravnborg.org> <20080123113308.GA20679@uranus.ravnborg.org> =>
08b6d290f977d8145804fd2b9bc2c331f2484f8e
<200801200017.24445.yinghai.lu@sun.com>
<20080120004439.GA3715@spacedout.fries.net> => 941ed3b53086697eac7449f3ab5d2c5ab3259de2
<20080120004440.GA3738@spacedout.fries.net> => 80c002ddd2e732062e4371314d40515d0b5d8415
<20080120123815.GA20082@gollum.tnic>
<20080120172151.GA22524@uranus.ravnborg.org>
<20080120192019.GH8669@does.not.exist> <20080213213120.GY3383@cs181133002.pp.htv.fi> =>
3202e1811fd312f3f32ddc8f526aa2691b64ec55
```

Listing 4.8: Ausschnitt der generierten PaStA-Ausgabe

Nach der Konfiguration von PaStA wird zunächst die mbox-Datei eingelesen (`./pasta mbox_prepare mbox ./mbox`). Im Anschluss wird der benötigte Cache erstellt (`./pasta cache -create mbox`). Danach erfolgt die initiale Analyse der mbox (`./pasta analyse -mbox init`). Dadurch, dass auch die Patches, die nicht in den master-Branch übernommen wurden, betrachtet werden sollen, wird der Befehl `./pasta analyse -mbox upstream` ausgeführt. Dieser Befehl vergleicht alle E-Mails mit allen Commits vom master-Branch. Im letzten Schritt werden die Patches bewertet und es wird eine Ausgabedatei erstellt, welche die Hashes der Commits mit den Message-IDs der dazugehörigen E-Mails beinhaltet (`./pasta rate`).
Einen Auszug dieser generierten Datei kann im Listing 4.8 eingesehen werden. Jede Zeile in der generierten Datei beinhaltet dabei die Informationen für genau einen Commit. Dabei werden links von dem Pfeil (=>) die Message-IDs aufgelistet und rechts des Pfeils der jeweilige Hash-Wert eines Commits. Die Anzahl an Message-IDs in einer Zeile steht dabei für die Anzahl an Anläufen, die ein Enwtickler benötigt, damit sein Patch in den master-Branch übernommen wird. Wurde eine Änderung nicht in den master-Branch übernommen, sind in den jeweiligen Zeilen nur die Message-IDs der Patches und kein Commit-Hash enthalten.
Betrachtet man die erste Zeile vom Listing 4.8, kann man daraus ablesen, dass der Patch, der in der Nachricht mit der Message-ID `<20080119150737.GB7231@cvg>` enthalten ist, in den master-Branch mit dem Commit `102f4a02de5c7217a04ccbbc24f35224b98bb183` übernommen wurde. Da nur eine Message-ID in dieser Zeile vorhanden ist, bedeutet dies, dass der Entwickler den Patch nicht nachbessern musste und der Patch sofort in den master-Branch übernommen wurde.
Die Patches in den Zeilen zwei und drei vom gleichen Listing haben keinen Commit-Hash. Dies bedeutet, dass die Patches von den jeweiligen Maintainern nicht in den master-Branch übernommen wurden.

Ähnlich wie bei den anderen zuvor generierten Daten, muss auch die von PaStA generierte Datei zuerst aufbereitet werden. Hierbei wird ebenfalls das CSV-Format gewählt. Da das Abändern der erhaltenen Daten nicht mit sonderlich viel Aufwand verbunden ist, wird dies durch simples „Suchen und Ersetzen“ im Texteditor durchgeführt. Als Spaltennamen werden dabei `message_ ids` und `commit_ hash` eingefügt. Der Pfeil (=>), welcher die Zuordnung zwischen Message-IDs und Commit-Hash darstellt, wird dabei durch ein Komma ersetzt. Somit erhält man eine auswertbare CSV-Datei.
Um die Anzahl an Anläufe, bis ein Patch in den master-Branch übernommen wird und die Patch-Acceptance-Rate, zu bekommen, wurde ein kleines Java-Programm implementiert. Als Input verwendet es die von Hand abgeänderte Datei mit der Zuordnung von Message-IDs zu Commit-Hash. Die Anzahl an Anläufe wurde durch das Zählen der Message-IDs berechnet. Die Patch-Acceptance-Rate wird ebenfalls in der Aufbereitung generiert. Sie wird als Kehrwert der `patch_message_count` angegeben. Einen kurzen Auszug der Ausgabe des Programms kann in der Tabelle 4.22 eingesehen werden.

Tabelle 4.22: Aufbereitete PaStA-Output-Datei (vorliegend als CSV-Datei)

message_ids	**commit_hash**	**patch_message_count**	**acceptance_rate**
<20080119150737.GB7231@cvg>	102f4a02de5c7217a04ccbbc24f35224b98bb183	1	1
<20080119154453.1afddbfb@lxorguk.ukuu.org.uk>		1	
<20080119155126.180a010d@lxorguk.ukuu.org.uk>		1	
<20080119160345.4160d8a5@lxorguk.ukuu.org.uk>	9e491e54f0589cc26e2c096664e9d95493b1af29	1	1
<20080119160506.5aab8000@lxorguk.ukuu.org.uk>	5c5a26fa9cd27ed4642f5a4ec76d9b487959e8dc	1	1
<20080119160743.GA8352@csn.ul.ie>		1	
<20080119160758.1d2c908a@lxorguk.ukuu.org.uk>		1	
<200801192045.17291.yinghai.lu@sun.com>		1	
<20080119212019.GB29070@uranus.ravnborg.org> <20080123113308.GA20679@uranus.ravnborg.org>	08b6d290f977d8145804fd2b9bc2c331f2484f8e	2	0.5
<200801200017.24445.yinghai.lu@sun.com>		1	
<20080120004439.GA3715@spacedout.fries.net>	941ed3b53086697eac7449f3ab5d2c5ab3259de2	1	1
<20080120004440.GA3738@spacedout.fries.net>	80c002ddd2e732062e4371314d40515d0b5d8415	1	1
<20080120123815.GA20082@gollum.tnic>		1	
<20080120172151.GA22524@uranus.ravnborg.org>		1	
<20080120192019.GH8669@does.not.exist> <20080213213120.GY3383@cs181133002.pp.htv.fi>	3202e1811fd312f3f32ddc8f526aa2691b64ec55	2	0.5

In Tabelle 4.22 Zeile 1 musste der Entwickler kein einziges Mal bei seinem Patch nachbessern. Dadurch ist der Eintrag in der Spalte `patch_message_count` gleich 1. Der Kehrwert ist ebenfalls 1 ()Patch-Acceptance-Rate).
Der Commit mit dem Hash `3202e1811fd312f3f32ddc8f526aa2691b64ec55` benötigte eine Nachbesserung. Dadurch ergibt sich `patch_message_count` = 2. Die dazugehörige Patch-Acceptance-Rate ist dabei 0,5. Wurde ein Patch nicht in den master-Branch übernommen, gibt es auch keinen dazugehörigen Commit-Hash dazu. Dies ist der Fall in den Zeilen, in denen der Wert in der Spalte `acceptance_rate` fehlt.

5. Statistische Analysen

Nach der Aufbereitung der generierten Daten, müssen diese statistisch Ausgewertet werden. Zu diesem Zweck wird das Statistikprogramm STATA verwendet.

5.1 Einlesen und Zusammenfügen der Daten

Damit STATA eine statistische Cox-Analyse auf den Daten ausführen kann, müssen diese zu einer großen Datei zusammengefügt werden. Dabei werden zunächst die einzelnen Dateien eingelesen und kleinere Änderungen an den Variablennamen durchgeführt. Im nächsten Schritt werden die Datumsinformationen so abgeändert, damit STATA diese verarbeiten kann. Anschließend werden die einzelnen Dateien anhand von den IDs zusammengefügt.
Die fertige Datei, welche zur statistische Analyse verwendet wird, kann in Tabelle 5.1, Tabelle 5.2 und Tabelle 5.3 eingesehen werden. Das für die Generierung verwendet STATA-Skript kann im Anhang im Abschnitt A.3 eingesehen werden.
Eine Zeile liefert dabei Informationen für einen Entwickler in einem bestimmten Monat in einem Jahr. So sind zum Beispiel in Zeile 1 sämtliche Informationen zu Entwickler mit der developer_id 5528 vom März 2016 enthalten. In der darauffolgenden Zeile sind die Informationen zum gleichen Entwickler vom April 2016 verfügbar. Die Spalte `developer_id` ist dabei eine generierte fortlaufende Nummer, die für jeden Entwickler einzigartig ist. Die Spalten `month` und `year` geben an, auf welchen Monat und Jahr sich die Daten beziehen. `month_year` ist eine von STATA generierte Spalte, die für die richtige Datumsinterpretation benötigt wird.
Die Informationen, wie viele direkte E-Mails an Linus Torvalds in einem bestimmten Monat gesendet wurden, ist in der Spalte `mail_to_torvalds` enthalten. Die ähnlichen Spalten mit den direkten E-Mails an andere wichtige Maintainer (Morton, Kroah Hartman, Viro, Molnar, Miller, Iwai und Zolnierkiewicz) wurde aus Platzgründen in die Tabelle nicht mit aufgenommen (vgl. Tabelle 4.7). Die nächste Variable `email_count` liefert die Anzahl an gesendeten E-Mails von einem Entwickler in dem jeweiligen Monat. `initial_mail` is die Anzahl an E-Mails, die ein Entwickler in einem Monat initial, also keine Antworten oder Weiterleitungen, gesendet hat.

Tabelle 5.1: In STATA aufbereiteter Datensatz, Teil 1

developer_id	month	year	month_year	mail_to_torvalds	email_count	initial_mail	count_cc	count_to	count_from	count_reply_to	count_attachments
5528	3	2016	2016m3	0	38	4	396	72	38	0	0
5528	4	2016	2016m4	0	92	8	1083	154	92	3	0
5528	5	2016	2016m5	0	44	8	435	99	44	0	0
5528	6	2016	2016m6	0	62	6	628	93	62	0	0
5528	7	2016	2016m7	1	35	3	325	51	35	0	0
5528	8	2016	2016m8	1	21	3	179	28	21	1	0
5675	4	2005	2005m4	0	2	1	3	0	2	0	0
5675	5	2005	2005m5	0	4	4	0	0	4	0	0
5675	6	2005	2005m6	0	6	1	6	0	6	0	0
5675	7	2005	2005m7	0	3	0	6	0	3	0	0
19 848	5	2007	2007m5	0	1	0	3	1	1	0	0
19 848	6	2007	2007m6	0	24	0	127	24	24	0	0
19 848	7	2007	2007m7	0	19	2	69	21	19	0	0
19 848	8	2007	2007m8	0	20	2	101	24	20	0	0
19 848	9	2007	2007m9	0	7	2	25	7	7	0	0
19 848	10	2007	2007m10	0	9	0	40	9	9	0	0
19 848	11	2007	2007m11	0	9	0	34	10	9	0	0

Tabelle 5.2: In STATA aufbereiteter Datensatz, Teil 2

developer_id	maintainer	private_mail_provider	top_company	is_edu	minutes	changed_files	commit_count	added_lines	deleted_lines	diff_size
5528	1	0	0	0	3951	0	0	0	0	0
5528	1	0	0	0	696.2	0	0	0	0	0
5528	1	0	0	0	2608.142 857	0	0	0	0	0
5528	1	0	0	0	5264	0	0	0	0	0
5528	1	0	0	0	65.5	0	0	0	0	0
5528	1	0	0	0	4183	0	0	0	0	0
5675	0	0	0	0	0	0	0	0	0	0
5675	0	0	0	0	0	0	0	0	0	0
5675	1	0	0	0	0	23	13	967	996	1963
5675	1	0	0	0	0	962	31	1246	5370	6616
19 848	1	0	1	0	0	0	0	0	0	0
19 848	1	0	1	0	0	0	0	0	0	0
19 848	1	0	1	0	153.5	235	82	5648	1827	7475
19 848	1	0	1	0	241	72	9	414	126	540
19 848	1	0	1	0	0	0	0	0	0	0
19 848	1	0	1	0	0	9	9	114	81	195
19 848	1	0	0.444 444	0	0	0	0	0	0	0

Tabelle 5.3: In STATA aufbereiteter Datensatz, Teil 3

developer_id	change_on_function	patch_message_count	acceptance_rate	patch_message_count_min	patch_message_count_max	not_accepted_count	accepted_count	total_mail_to_maintainer
5528	0	0	0	0	0	38	0	0
5528	0	0	0	0	0	92	0	0
5528	0	0	0	0	0	44	0	1
5528	0	0	0	0	0	62	0	0
5528	0	0	0	0	0	35	0	1
5528	0	0	0	0	0	21	0	3
5675	0	0	0	0	0	2	0	0
5675	0	0	0	0	0	4	0	0
5675	213	0	0	0	0	6	0	0
5675	44	0	0	0	0	3	0	0
19 848	0	0	0	0	0	1	0	1
19 848	0	0	0	0	0	24	0	0
19 848	476	1	1	1	1	18	1	1
19 848	32	2	.5	2	2	19	1	2
19 848	0	0	0	0	0	6	1	0
19 848	26	1	1	1	1	9	0	0
19 848	0	0	0	0	0	9	0	0

Die Variablen `count_cc`, `count_to`, `count_from` und `count_reply_to` enthalten die Summen der jeweiligen E-Mail-Header für alle gesendeten E-Mails eines Entwicklers pro Monat. `count_attachments` zählt alle Anhänge, die bei den gesendeten E-Mails eines Entwicklers in einem Monat angehängt waren. (vgl. Abschnitt 4.3)
Die Splate `maintainer` gibt an, ob der Entwickler in diesem Monat den Maintainer Status hatte (1) oder nicht (0) (vgl. Abschnitt 4.2). Wurden E-Mails von einer Firmen E-Mail Adresse (vgl. Tabelle 4.6) verschickt, wird dies in der Spalte `top_company` angezeigt. Dabei werden Nachrichten, die von Firmen stammen mit 1 und die restlichen als 0 gewertet. Die Werte werden addiert und durch die Anzahl geteilt. Das gleiche Prinzip geschieht in der Spalte `is_edu`. Dabei werden die Nachrichten, deren Adresse auf .edu endet, als 1 und andere als 0 gewertet. Nach dem Aufaddieren und Teilen durch die Anzahl erhält man den jeweiligen Wert in der Tabelle (vgl. Abschnitt 4.3). Die Information `minutes` stellt die durchschnittliche Antwortzeit in Minuten der ersten Antwort auf eine initiale E-Mail dar. Sendet ein Entwickler zwei E-Mails an die Mailingliste, auf die erste wird in 60 und auf die zweite in 30 Minuten geantwortet, so beträgt der Mittelwert 45 Minuten. Die Variablen `changed_files`, `commit_count`, `added_lines`, `deleted_lines`, `diff_size`, `change_on_function` sind die Summen der jeweiligen Variable von der Commit-Liste pro Monat und wurden bereits im Abschnitt 4.4 ausführlich erklärt.
Die Spalte `patch_message_count` bildet den Durchschnitt von der Anzahl an Nachbesserungen, die ein Entwickler benötigt, um einen Patch in den master-Branch zu bekommen. Die Information `patch_acceptance_rate` ist dabei der Kehrwert von `patch_message_count`. Damit man einen Überblick über die Abweichung der Anzahl von Nachbesserungen bekommt, wurden die Spalten `patch_message_count_min`, mit dem minimalen Wert, und `patch_message_count_max`, mit dem maximalen Wert, hinzugefügt. Informationen über Patches, die von Maintainer nicht akzeptiert wurden liefert die Spalte `not_accepted_count`. Die Anzahl an akzeptierten Patches liefert die Spalte `accepted_count` (vgl. Abschnitt 4.5). Die letzte Spalte `total_mail_to_maintainer` summiert sämtliche direkten E-Mails an wichtige Maintainer in einem Monat auf.

5.2 Auswertung und Modelle

Um die Hypothesen zu überprüfen, wurden insgesamt fünf Modelle gerechnet. Modell 1 betrachtet lediglich die Kontrollvariablen. In Modell 2 fließen die direkten- und die Kontrollvariablen mit ein. In die Analyse bei Modell 3 fließen die direkten- und indirekten technischen Variablen und die Kontrollvariablen mit ein. Ähnlich wie in Modell 3 werden in Modell 4 die direkten- und indirekten Variablen der kommunikativen Variablen und die Kontrollvariablen beachtet. Im letzten, Modell 5, werden alle Variablen mit in die Berechnung einbezogen. Die Werte sind die Koeffizienten der jeweiligen Variable zu einem bestimmten Modell. Die Sternchen geben die Signifikanz an. Die Werte in den Klammern unter den Koeffizienten sind die robusten Standardfehler. Um einen Überblick über die Anzahl an Entwicklern und Beobachtungen zu bekommen, werde diese ebenfalls in der Auswertung mit angegeben. Die Werte `Wald chi2` und `Prob > chi2` liefern Informationen darüber, wie gut ein Modell die verwendeten Daten erklärt und ob die Null-Hypothese abgelehnt werde kann, oder nicht. AIC und BIC sind Gütemaße, welche die Varianz in der anhängigen Variable durch eine unabhängige Variable erklären. Die berechneten Modelle können in

Abbildung 5.2 eingesehen werden. In der folgenden Erklärung wird in runden Klammern hinter dem angepassten Namen die Art der Zusammenfassung der Werte pro Monat angezeigt. In den eckigen Klammern dahinter steht der ursprüngliche Name, der auch in der zusammengefügten Datei zu finden ist. Damit die einzelnen Koeffizienten bei der Auswertung vergleichbarer werden, wurden die folgenden Variablen standardisiert. Dies geschieht ebenfalls im STATA-Skript und kann im Abschnitt A.3 nachgelesen werden.
Für Hypothese 1 werden dabei die technischen Beiträge durch die `Anzahl an Commits (Summe)` [`commit_count`] dargestellt. Desto mehr Commits ein Entwickler tätigt, umso mehr technische Beiträge leistet er am Projekt. Betrachtet man dabei den Koeffizienten der Variable in Modell 2, ist dieser positiv und signifikant ($p<0.05$). Der Betrag des Koeffizienten liegt dabei bei 0.15. Daraus lässt sich folgern, dass Hypothese 1 bestätigt werden kann und eine hohe Anzahl an Commits die Aufstiegschancen eines Entwicklers begünstigt.
Die Variable `Patch-Acceptance-Rate (Durchschnitt)` [`acceptance_rate`] stellt in Hypothese 2 die Akzeptanz der technischen Beiträge durch Maintainer dar. Dieser Wert variiert zwischen 0 und 1. Dabei bedeutet ein Wert von 1 die bestmögliche Akzeptanz. Diese Variable wird bei der Cox-Analyse als direkte Variable verwendet. Die Signifikanz ($p<0.05$) liegt auf der gleichen Ebene wie die der Anzahl an Commits. Der positive Einfluss durch den Koeffizienten von 0.31 ist deutlich größer als der von der Anzahl an Commits. Dies bedeutet, dass eine hohe Patch-Acceptance-Rate die Wahrscheinlichkeit, dass ein Entwickler irgendwann zum Maintainer wird, stärker begünstigt. Der Interaktionseffekt, den die berechnete Variable `Anzahl Commits x Patch-Acceptance-Rate` darstellt, ist bei der Cox-Analyse nicht signifikant. Dies bedeutet, dass ein verstärkender Effekt der Patch-Acceptance-Rate auf die Anzahl an Commits nicht bestätigt werden kann. Für Hypothese 2 folgt daraus, dass nur der direkte Effekt der Patch-Acceptance-Rate positiv auf die Aufstiegschancen von Entwicklern wirkt.
Die Variable `Anzahl an E-Mails (Summe)` [`email_count`] stellt in Hypothese 3 die Kommunikationsfreudigkeit eines Entwicklers dar. Desto höher der Wert, umso mehr kommuniziert ein Entwickler über die Mailingliste. In die Cox-Analyse fließt diese als direkte Variable mit ein. Der Koeffizient der Variable in Modell 2 ist sehr signifikant ($p<0.01$) und positiv. Der Betrag von 0.47 des Koeffizienten ist mit einer der höchsten in der Auswertung und beeinflusst die Aufstiegschancen von Entwicklern zum Maintainer positiv. Daraus lässt sich folgern, dass Hypothese 3 erfüllt ist und ein Entwickler, der viel über die Mailingliste kommuniziert höhere Aufstiegschancen hat.
Um herauszufinden, ob Hypothese 4 bestätigt werden kann, wird die Variable `relative Anzahl E-Mails an Maintainer (Summe)` verwendet. Diese liefert den Anteil an E-Mails, die direkt und nur an einen wichtigen Maintainer gesendet wurden, um eine Multikollinearität zu vermeiden. Je höher der Wert ist, umso häufiger kommunizierte der jeweilige Entwickler mit einem wichtigen Maintainer. Dieser Wert wird ebenfalls als direkte Variable in die Cox-Analyse mit übernommen. Dadurch, dass die Koeffizienten nicht signifikant sind, kann kein positiver Effekt auf die Aufstiegswahrscheinlichkeit von Entwicklern am LINUX-Kernel bestätigt werden.

Abbildung 5.1: Korrelationstabelle zur Analyse

	1	2	3	4	5	6	7	8	9	10
1 Anzahl an Commits (Summe)	1.00									
2 Patch-Acceptance-Rate (Durchschnitt)	0.06	1.00								
	(0.00)									
3 Anzahl an Commits x Patch-Acceptance-Rate	0.49	-0.09	1.00							
	(0.00)	(0.00)								
4 Anzahl E-Mails (Summe)	0.25	-0.01	0.07	1.00						
	(0.00)	(0.55)	(0.00)							
5 relative Anzahl E-Mails an Maintainer (Summe)	0.04	0.02	-0.01	-0.00	1.00					
	(0.00)	(0.16)	(0.48)	(0.81)						
6 Anzahl E-Mails x relative Anzahl E-Mails an Maintainer	0.02	-0.05	-0.02	-0.03	-0.24	1.00				
	(0.11)	(0.00)	(0.05)	(0.00)	(0.00)					
7 Antwortzeit auf erste E-Mail (Durchschnitt)	0.02	0.02	-0.02	0.03	0.02	0.01	1.00			
	(0.04)	(0.03)	(0.10)	(0.02)	(0.13)	(0.41)				
8 Anzahl geänderter Zeilen (Summe)	0.50	0.02	0.37	0.02	-0.01	0.00	-0.00	1.00		
	(0.00)	(0.04)	(0.00)	(0.06)	(0.37)	(0.82)	(0.95)			
9 Anzahl an Nachbesserungen (Durchschnitt)	-0.04	-0.85	0.08	0.01	-0.02	0.02	-0.02	-0.01	1.00	
	(0.00)	(0.00)	(0.00)	(0.22)	(0.03)	(0.09)	(0.05)	(0.20)		
10 relative Anzahl Nachrichten berufliche E-Mail (Durchschnitt)	-0.04	-0.01	0.01	-0.16	-0.00	0.00	0.06	0.01	0.01	1.00
	(0.00)	(0.41)	(0.60)	(0.00)	(0.88)	(0.93)	(0.00)	(0.56)	(0.59)	

Einen negativen Koeffizienten besitzt die Variable `Anzahl E-Mails x relative Anzahl E-Mails an Maintainer`. Da dieser nicht signifikant sind, lässt sich kein moderierender Effekt dieser Variable Schlussfolgern. Für Hypothese 4 bedeutet dies, dass die direkte Kommunikation mit wichtigen Maintainern die Aufstiegschancen von Entwicklern nicht verstärkt und somit nicht bestätigt werden kann.
Die Kontrollvariablen `Antwortzeit auf erste E-Mail (Durchschnitt)` [`minutes`] liefert die durchschnittliche Antwortzeit auf eine initiale E-Mail eines Entwicklers in Minuten. Diese ist bei allen Modellen in der Auswertung nicht signifikant. Das gleiche gilt für die Kontrollvariable `Anzahl geänderter Zeilen (Summe)` [`diff_size`], welche die geänderten Zeilen in den Commits eines Entwicklers aufsummiert. Eine hohe Signifikanz haben die negativen Koeffizienten der Variable `Anzahl an Nachbesserungen` [`patch_message_count`] in den Modellen 1 und 4, welche den Durchschnitt der Nachbesserungen, die ein Entwickler machen musste, damit seine Änderungen in den master-Branch übernommen worden sind beschreibt. In den restlichen Modellen sind die Koeffizienten ebenfalls negative, jedoch nicht signifikant. Dies bedeutet, dass der negative Einfluss einer hohen Anzahl an Nachbesserungen auf die Aufstiegschancen eines Entwicklers nur gegeben ist, wenn die technischen Variablen nicht im Modell betrachtet werden. Die letzte Kontrollvariable `relative Anzahl Nachrichten berufliche E-Mail` ist der Durchschnitt des Anteils an Nachrichten, die von einer beruflichen E-Mail Adresse gesendet wurden. In Modell 1 und 3 sind diese Koeffizienten signifikant ($p<0.05$). Durch den Koeffizienten von -0.07 lässt sich ein negativer Effekt auf die Aufstiegschancen von Entwicklern schließen. Dieser Effekt ist nur zu beobachten, wenn keine technischen Variablen mit in das Modell einfließen. Im Allgemeinen zeigen die Kontrollvariablen eine nur sehr begrenzte Wirkung bei den gewählten Modellen.
Bei der Analyse wurden 855 unterschiedliche `Entwickler` in 3805 `Beobachtungen` analysiert. Laut den Variablen `Wald chi2` und `Prob > chi2` sind die gewählten Modelle robust und erklären die Daten gut. Die Null-Hypothese kann abgelehnt werden. `AIC` und `BIC` haben bei allen Modellen die gleichen Werte und stellen ein Gütemaß für die Modelle dar. Dies bedeutet, dass alle Modelle die gleiche Genauigkeit für eine Vorhersage haben.

Abbildung 5.2: Ergebnisse der statistischen Cox-Analyse

Variablen	Model 1	Model 2	Model 3	Model 4	Model 5
Anzahl Commits (Summe)		0.15**	0.35***		0.24**
		(0.07)	(0.14)		(0.12)
Patch-Acceptance-Rate (Durchschnitt)		0.31**	0.25		0.26
		(0.14)	(0.16)		(0.16)
Anzahl Commits x Patch-Acceptance-Rate			-0.14		-0.12
			(0.16)		(0.14)
Anzahl E-Mails (Summe)		0.47***		0.52***	0.47***
		(0.06)		(0.06)	(0.06)
relative Anzahl E-Mails an Maintainer (Summe)		-0.04		-0.02	-0.03
		(0.04)		(0.06)	(0.06)
Anzahl E-Mails x relative Anzahl E-Mails an Maintainer				0.03	0.01
				(0.07)	(0.07)
Antwortzeit auf erste E-Mail (Durchschnitt)	-0.03	-0.08	-0.29	-0.02	-0.21
	(0.08)	(0.12)	(0.24)	(0.08)	(0.23)
Anzahl geänderter Zeilen (Summe)	0.02	-0.07	-0.15	0.02	-0.06
	(0.03)	(0.10)	(0.24)	(0.03)	(0.08)
Anzahl an Nachbesserungen (Durchschnitt)	-0.47***	-0.09	-0.13	-0.47***	-0.13
	(0.11)	(0.17)	(0.18)	(0.11)	(0.18)
relative Anzahl Nachrichten berufliche E-Mail (Durchschnitt)	-0.07**	-0.05	-0.07**	-0.05	-0.05
	(0.04)	(0.03)	(0.03)	(0.03)	(0.03)
Entwickler	855	855	855	855	855
Beobachtungen	3,805	3,805	3,805	3,805	3,805
Wald chi2	25.48	139.6	69.68	119.1	150.6
Prob > chi2	0.00	0.00	0.00	0.00	0.00
AIC	3623	3623	3623	3623	3623
BIC	3685	3685	3685	3685	3685

Robuste Standardfehler in Klammern
*** p<0.01, ** p<0.05, * p<0.1

Tabelle liefert die Koeffizienten der unterschiedlichen Modelle zu den jeweiligen Variablen. Hinter dem Variablennamen wird in Klammern die Art der Zusammenfassung der Daten pro Monat angegeben. Im ersten Bereich der Auswertung werden die technischen Variablen betrachtet. Der zweite Abschnitt listet die Ergebnisse der kommunikativen Variablen auf. Anschließend werden die Kontrollvariablen aufgelistet. Naben der Anzahl an Entwicklern und Beobachtungen folgen zum Schluss ein paar allgemeine statistische Werte wie Wald chi2, Prob > chi2, AIC (Akaike information criterion) und BIC (Bayesian information criterion).

6. Ergebnisse und Diskussion

Durch die Auswertung der aufbereiteten Daten komme ich zum dem Ergebnis, dass technische Beiträge einen Einfluss auf die Aufstiegschancen von Entwickler haben. Dies wird bei der Anzahl an Commits deutlich. Die Cox-Analyse zeigte, wenn ein Entwickler viele Commits tätig, steigt die Wahrscheinlichkeit, dass er irgendwann Maintainer wird. Dies könnte dadurch erklärt werden, dass sich der Entwickler viel mit der Entwicklung des Linux-Kernels beschäftigt und dadurch viel Wissen aufbaut. Für andere Entwickler erscheint dieser glaubwürdiger und vertrauenswürdiger [DO11]. Das führt zum schnelleren Aufstieg in der Entwicklerhierarchie.
Ein weiterer technischer Aspekt, der analysiert wurde, ist die Akzeptanz von technischen Beiträgen durch Maintainer. Die Auswertung der Analyse zeigt, dass der Einfluss einer hohen Akzeptanz ebenfalls den Aufstieg zum Maintainer positiv beeinflusst. Diese wird durch die Patch-Acceptance-Rate beschrieben. Eine hohe Patch-Acceptance-Rate steht dabei für wenig Nachbesserungsbedarf bei den Patches. Dadurch sind die Patches qualitativ hochwertig und zeigen, dass der jeweilige Entwickler kompetent ist. Dies begünstigt die Wahrscheinlichkeit zum Maintainer aufzusteigen.
Neben den technischen- wurden auch die kommunikativen Beiträge analysiert. Diese wirken sich ebenfalls auf die Aufstiegschancen von Entwicklern aus. Kommuniziert ein Entwickler viel über die Mailingliste wirkt sich dies positiv auf die Wahrscheinlichkeit später irgendwann Maintainer zu werden aus. Als Maß für die Kommunikation über die Mailingliste wurde dabei die Anzahl an gesendeten E-Mails gewählt. Der positive Effekt lässt sich dadurch erklären, dass Entwickler die viele E-Mails schreiben großes Interesse an der Entwicklung des Linux-Kernels haben. Durch das beantworten und stellen von Fragen teilt der Entwickler sein Wissen und hilft anderen weiter. Dadurch erreicht er höheres Ansehen [DO11]. Entwickler mit hohem Ansehen haben eine höhere Aufstiegschance in der Entwickler-Community.
Eine weiterer kommunikativer Beitrag der analysiert wurde, ist die Sichtbarkeit bei wichtigen Maintainern. Dies wurde durch den Anteil an gesendeten E-Mails an wichtige Maintainer beschrieben. Wie die statistische Cox-Analyse zeigt, kann diese Hypothese nicht bestätigt werden. Daraus lässt sich folgern, dass der Einfluss durch direkte Kommunikation mit wichtigen Maintainern keinen positiven Einfluss auf die

Aufstiegschancen von Entwicklern am Linux-Kernel hat.
Zusammenfassend kann man sagen, dass die analysierten technischen Einflussfaktoren, wie die Anzahl und die Akzeptanz von technischen Beiträgen sich positiv auf die Aufstiegswahrscheinlichkeit von Entwicklern am Linux-Kernel auswirken. Ein weiterer positiver Faktor ist die Kommunikation über die Mailingliste. Dass die direkte Kommunikation mit wichtigen Maintainern die Aufstiegschancen positiv beeinflusst, kann dabei nicht bestätigt werden.

7. Verwandte Arbeiten

Die vorliegende Arbeit baut auf die Bachelorarbeit „The Usual Suspects? Developers' Progression from Peri- phery to Core in the Linux Kernel Development Project.“ von Stefan Berreiter auf, welche ebenfalls im Umfeld dieser Forschungsgruppe angefertigt wurde [Ber16]. Der wesentliche Unterschied zu dieser Ausarbeitung ist die Menge an betrachteten Daten bei der statistischen Auswertung. Stefan Berreiter verwendete für seine Analysen zwei unterschiedliche Entwicklergruppen. Zum einen die Entwickler, welche nach einer gewissen Zeit zum Maintainer wurden und die Entwickler, welche ihren Status nie verändert haben. Dabei wurden „nur“ 108 Entwickler in jeder Gruppe betrachtet. Bei der vorliegenden Arbeit wurden dabei alle verfügbaren Informationen verwendet, also die komplette E-Mail Kommunikation und alle Commits von allen beteiligten Entwicklern seit April 2005.
Stefans Analyse zeigte, dass sowohl technische- und soziale Faktoren Einfluss auf die Aufstiegschancen eines Entwicklers haben. Besonders große positive Faktoren waren dabei die Anzahl und Größe von Commits und ein breites technisches Wissen beim Projekt (Beteiligung an vielen unterschiedlichen Bereichen/Subsystemen). Faktoren, wie die unterstützende Kommunikation und die soziale Beziehung zu wichtigen Entwicklern, speziell zu Linus Torvalds, sind positive soziale Faktoren. Einen Weniger großen Einfluss auf die Wahrscheinlichkeit ein Maintainer zu werden hat dabei die Anzahl an gesendeten E-Mails oder die Beliebtheit an den mitwirkenden Subsystemen.

8. Zukünftige Arbeiten

8.1 E-Mail-Inhalt analysieren

Bei der bisherigen Analyse werden lediglich die Header der gesendeten E-Mails betrachtet. Dabei handelt es sich um die standard From-, To-, Cc-, Date-, Message-ID- und Subject-Header. Für weitere, genauere Auswertungen könnte zum Beispiel der Inhalt der einzelnen Nachrichten betrachtet werden. So könnten wichtige, technische Inhalte die Code enthalten von eher simplen Einzeilern oder „Smalltalk“ getrennt werden. Eine Klassifizierung in technisch relevante und eher unwichtige Nachrichten könnte ebenfalls stattfinden.

Als Beispiel für eine eher unwichtige E-Mail, bei der sich der Entwickler für das reviewen von Code bedankt wird, kann in den Listings 8.1 und 8.2 eingesehen werden. Dabei handelt es sich um einen Zweizeiler, bei dem sich Akira Iguchi für das Warten seines Codes bedankt.
Es gibt auch Nachrichten, die Informationen zu Commits enthalten. Dabei werden hinzugefügte Zeilen durch ein + und entfernte Zeilen durch ein - verdeutlicht. Als Beispiel kann dabei die E-Mail aus dem Listing 8.3, 8.4 und 8.5 betrachtet werden. Eine Nachricht, die in technischer Hinsicht sehr relevant ist, muss nicht immer Code enthalten. Diskussionen und Informationen über Veränderungen können im Fließtext einer E-Mail vorkommen. Als Beispiel dient dabei die E-Mail im Listing 8.6.

```
1  From MAILER-DAEMON Wed Jun  8 16:02:26 2016
2  Path: news.gmane.org!not-for-mail
3  From: <akira2.iguchi@toshiba.co.jp>
4  Newsgroups: gmane.linux.ide,gmane.linux.kernel
5  Subject: RE: [PATCH] pata_scc: add proper cable detection method
6  Date: Thu, 13 Oct 2011 07:23:19 +0000
7  Lines: 93
8  ...
9  Thank you for maintaining our code.
10 This patch makes no difference in functionality.
```

Listing 8.1: Auszug einer kurzen Dank E-Mail, Teil 1.

```
1  Acked-by: Kou Ishizaki <kou.ishizaki@toshiba.co.jp>
2  Acked-by: Akira Iguchi <akira2.iguchi@toshiba.co.jp>
3
4  > ------Original Message------
5  > From: Bartlomiej Zolnierkiewicz [mailto:bzolnier@gmail.com]
6  > Sent: Thursday, October 13, 2011 12:17 AM
7  > To: Alan Cox
8  > Cc: Jeff Garzik; linux-ide@vger.kernel.org;
9  > linux-kernel@vger.kernel.org; ishizaki ko
10 ...
11 --
12 To unsubscribe from this list: send the line "unsubscribe linux-ide"
       in
13 the body of a message to majordomo@vger.kernel.org
14 More majordomo info at  http://vger.kernel.org/majordomo-info.html
```

Listing 8.2: Auszug einer kurzen Dank E-Mail, Teil 2.

```
1  From MAILER-DAEMON Wed Jun  8 11:07:19 2016
2  Path: news.gmane.org!not-for-mail
3  From: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
4  Newsgroups: gmane.linux.kernel.mm,gmane.linux.kernel
5  Subject: [BUGFIX][PATCH 1/3] memory hotplug: fix next block
       calculation in
6   is_removable
7  Date: Mon, 6 Sep 2010 14:42:28 +0900
8  Organization: FUJITSU Co. LTD.
9  Lines: 56
10 ...
11 From: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
```

Listing 8.3: Auszug einer E-Mail mit Commit Informationen, Teil 1.

Um also eine bessere Analyse der Kommunikation zwischen den einzelnen Entwicklern durchführen zu können, sollte in zukünftigen Arbeiten eine Inhaltliche Betrachtung und Klassifizierung von Nachrichten erfolgen.

```
1  next_active_pageblock() is for finding next _used_ freeblock. It
       skips
2  several blocks when it finds there are a chunk of free pages lager
       than
3  pageblock. But it has 2 bugs.
4
5    1. We have no lock. page_order(page) - pageblock_order can be minus
         .
6    2. pageblocks_stride += is wrong. it should skip page_order(p) of
         pages.
7
8  CC: stable@kernel.org
9  Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
10 ---
11  mm/memory_hotplug.c |   16 ++++++++--------
12  1 file changed, 8 insertions(+), 8 deletions(-)
13
14 Index: kametest/mm/memory_hotplug.c
15 ===================================================================
16 --- kametest.orig/mm/memory_hotplug.c
17 +++ kametest/mm/memory_hotplug.c
18 @@ -584,19 +584,19 @@ static inline int pageblock_free(struct
19  /* Return the start of the next active pageblock after a given page
       */
20  static struct page *next_active_pageblock(struct page *page)
21  {
22 -   int pageblocks_stride;
23 -
24     /* Ensure the starting page is pageblock-aligned */
25     BUG_ON(page_to_pfn(page) & (pageblock_nr_pages - 1));
26     -   /* Move forward by at least 1 * pageblock_nr_pages */
27 -   pageblocks_stride = 1;
28 -
29     /* If the entire pageblock is free, move to the end of free page
         */
30 -   if (pageblock_free(page))
31 -       pageblocks_stride += page_order(page) - pageblock_order;
```

Listing 8.4: Auszug einer E-Mail mit Commit Informationen, Teil 2.

```
+   if (pageblock_free(page)) {
+       int order;
+       /* be careful. we don't have locks, page_order can be changed
    .*/
+       order = page_order(page);
+       if (order > pageblock_order)
+           return page + (1 << order);
+   }

-   return page + (pageblocks_stride * pageblock_nr_pages);
+   return page + pageblock_nr_pages;
 }

 /* Checks if this range of memory is likely to be hot-removable. */

--
To unsubscribe, send a message with 'unsubscribe linux-mm' in
the body to majordomo@kvack.org.  For more info on Linux MM,
see: http://www.linux-mm.org/ .
Don't email: <a href=mailto:"dont@kvack.org"> email@kvack.org </a>
```

Listing 8.5: Auszug einer E-Mail mit Commit Informationen, Teil 3.

8.2 Inhalt der Commits analysieren

Um herauszufinden, welche Änderungen ein Entwickler mit einem Commit getätigt hat, wäre es sinnvoll diesen genau zu analysieren. Commits müssen nicht immer direkt den Code am Linux-Kernel verändern. Es können auch Änderungen oder das Hinzufügen von Kommentaren betroffen sein. Eine andere Möglichkeit ist das Refactoring, welches ebenfalls nicht die Funktionalität beeinträchtigt, sondern lediglich die Struktur des Codes verändert.
Eine Einteilung in unterschiedliche Commit-Klassen wäre dabei ein weiterer Ansatz die Analyse zu verfeinern.

8.3 Umfrage zur Subsystemwichtigkeit

Um die Wichtigkeit der einzelnen Subsysteme und Bereiche des Linux-Kernels besser Klassifizieren zu können, wäre eine Umfrage an die beteiligten Entwickler eine mögliche Option. Durch senden einer E-Mail an die Verteilerliste oder eine Umfrage in beliebten Foren könnte dies umgesetzt werden. Somit könnte nicht nur die Wichtigkeit anhand der Anzahl an Commits beziehungsweise an der Anzahl gesendeter E-Mails zu einem Subsystem, sondern auch die Meinung der einzelnen Entwickler berücksichtigt werden. Dadurch könnten sich bestimmte Verhalten und Zusammenhänge bei der Evolution des Linux Kernels und der Aufstiegschancen einzelner Entwickler zum Maintainer besser erklären lassen.

8.4 Aktuelle Daten verwenden

Bei der vorliegenden Arbeit werden Daten, die bis zum August 2016 erhoben wurden, verwendet. Diese wurden jeweils durch Shell-Scripte vom jeweiligen Server heruntergeladen und generiert. Um die Daten immer aktuell bei der Aufbereitung und

```
From MAILER-DAEMON Thu Jun  9 14:20:14 2016
Path: news.gmane.org!not-for-mail
From: Alexei Starovoitov <ast@plumgrid.com>
Newsgroups: gmane.linux.kernel,gmane.linux.kernel.api,gmane.linux.network
Subject: Re: [PATCH RFC v4 net-next 01/26] net: filter: add "load 64-bit
 immediate" eBPF instruction
Date: Wed, 13 Aug 2014 14:56:44 -0700
Lines: 26
...
On Wed, Aug 13, 2014 at 2:38 PM, Alexei Starovoitov <ast@plumgrid.com> wrote:
> On Wed, Aug 13, 2014 at 2:27 PM, H. Peter Anvin <hpa@zytor.com> wrote:
>> On 08/13/2014 02:23 PM, Andy Lutomirski wrote:
>>> On Wed, Aug 13, 2014 at 2:21 PM, H. Peter Anvin <hpa@zytor.com> wrote:
>>>> One thing about this that may be a serious concern: allowing the user to
>>>> control 8 contiguous bytes of kernel memory may be a security hazard.
>>>
>>> I'm confused.  What kind of memory?  I can control a lot more than 8
>>> bytes of stack very easily.
>>>
>>> Or are you concerned about 8 contiguous bytes of *executable* memory?
>>>
>>
>> Yes.  Useful for some kinds of ROP custom gadgets.
>
> I don't get it. What is ROP ?
> What is the concern about 8 bytes ?

looked it up. too many abbreviations now days.
x64 jit spraying was fixed by Eric some time ago, so JIT emitting
movabsq doesn't increase attack surface. various movs of 32-bit
immediates can be used for 'custom gadget' just as well.
Worst case JIT won't be enabled.
In classic BPF we allow junk to be stored in used fields of
'struct sock_filter' and so far that wasn't a problem.
eBPF is more paranoid regarding verification.
```

Listing 8.6: Auszug einer E-Mail mit technischen Diskussionen.

Analyse zur Verfügung zu haben, wäre es denkbar, die Datenbeschaffung in das Programm zur Aufbereitung zu integrieren.

9. Zusammenfassung

Zusammenfassend lässt sich sagen, dass es Einflussfaktoren gibt, die die Aufstiegschancen von Entwicklern am Linux-Kernel positiv beeinflussen. Dazu gehören die technischen Faktoren, wie die Anzahl an Commits, aber auch eine hohe Akzeptanz der Patches durch Maintainer. Der kommunikative Faktor Anzahl an E-Mails an die Mailingliste erhöht ebenfalls die Wahrscheinlichkeit, dass der Entwickler irgendwann zu einem Maintainer wird. Dass die direkte Kommunikation mit wichtigen Maintainern einen positiven Einfluss hat, konnte nicht bestätigt werden. Bei dieser Ausarbeitung könnte noch viel verbessert werden. Das Analysieren der E-Mail-Inhalte, oder die einzelnen Commits nach Wichtigkeit kategorisieren sind nur zwei Beispiele. Die anfänglichen Probleme mit PaStA konnten durch die Hilfe des Entwicklers schnell behoben und so gute Ergebnisse geliefert werden.

A. Anhang

A.1 Verwendete Programmiersprachen und Programme

- Da viele der Skripte zur Generierung der Daten in Python programmiert sind, wird diese Sprache ebenfalls benötigt. Als Entwicklungsumgebung wurde Pycharm verwendet. Dabei wurde folgende Version installiert:

  PyCharm Community Edition 2017.1.2
  Build # PC-171.4249.47, built on April 26, 2017
  JRE: 1.8.0_ 112-release-736-b21 x86_ 64
  JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
  Mac OS X 10.12.4
  Python *3.5.2*

  Link zur Webseite von Python: https://www.python.org
  Link zur Webseite von Pycharm: https://www.jetbrains.com/pycharm/

- Die Aufbereutng der Rohdaten erfolgte in der Programmiersprache Java. Dabei wurde Entwicklungsumgebung IntelliJ in der folgenden Version verwendet:

  IntelliJ IDEA 2017.1.2
  Build # IC-171.4249.39, built on April 25, 2017
  JRE: 1.8.0_ 112-release-736-b16 x86_ 64
  JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
  Mac OS X 10.12.4
  Java*(TM) SE Runtime Environment (build 1.8.0_ 102-b14)*

Link zur Webseite von Java: https://www.java.com/de/
Link zur Webseite von IntelliJ: https://www.jetbrains.com/idea/

- Um die Patch Acceptance Rate zu erhalten wurde PaStA-mbox verwendet. Da dieses Programm sehr viele Ressourcen benötigt, wurde mir freundlicherweise eine Workstation an der OTH Regensburg von Ralf Ramsauer zur Verfügung gestellt:

  Patch Stack Analysis v0.2

  Link zur Webseite: https://github.com/lfd/PaStA

- Um die statistische Analyse durchzuführen wurde Stata verwendet. Es wurde mir freundlicherweise vom Lehrstuhl für Betriebswirtschaftslehre mit Schwerpunkt Organisation, Technologiemanagement und Entrepreneurship auf der Workstation zur Verfügung gestellt. Folgende Version wurde verwendet:

  Stata/MP 14.2 for Windows (64-bit)

  Link zur Webseite: http://www.stata.com

- Um die Ausarbeitung anzufertigen wurde der LaTeX Editor Texmaker in der folgenden Version verwendet:

  Texmaker 4.5

  Link zur Webseite: http://www.xm1math.net/texmaker/

A.2 Verwendete Bibliotheken

- Nachdem die erforderlichen Daten generiert wurden, müssen sie in ein vernünftiges Format konvertiert werden. Dazu wurde Apaches Java CSV-Bibiothek verwendet:

  apache-commons-csv-1.4

  Link zur Webseite: https://commons.apache.org/proper/commons-csv/

- Um die mbox-Datei mit den E-Mail Nachrichten auslesen und darüber iterieren zu können, wurde Apaches mime4j Java-Bibliothek verwendet. Folgende

Version wurde dazu eingebunden:

apache-mime4j-0.8.1-SNAPSHOT

Link zur Webseite: https://james.apache.org/mime4j/index.html

A.3 Stata-Skript

```
1  cls
2  clear all
3
4  cd "F:\mbox_analyse\StataInput"
5
6  // save csv files as dta files
7  import delimited "emailRecords.csv", delim(",") varnames(1)
8
9  rename count_replyto count_reply_to
10 rename replyto reply_to
11 rename messageid message_id
12
13 save emailRecords.dta, replace
14
15 clear all
16
17 import delimited "commitRecords.csv", delim(",") varnames(1)
18 save commitRecords.dta, replace
19
20 clear all
21
22 import delimited "maintainerMerged.csv", delim(",") varnames(1)
23
24 rename maintainer_id developer_id
25 rename maintainer_name developer_name
26 rename maintainer_email developer_email
27 rename firsttimemaintainerdate first_time_maintainer_date
28 rename firsttimemaintainerrelease first_time_maintainer_release
29 rename allreleasesasmaintainer all_releases_as_maintainer
30
31 save maintainerMerged_removedDuplicates.dta, replace
32
33 clear all
```

Listing A.1: STATA-Skript, Teil 1

```
import delimited "similar-mailbox-merged.csv", delim(",") varnames(1)

gen message_id = first_message_id
drop if missing(commit_hash)
rename commit_hash commit_hash_old
gen str40 commit_hash = commit_hash_old

save similar-mailbox-merged-only-accepted.dta, replace

clear all

import delimited "similar-mailbox-merged.csv", delim(",") varnames(1)

gen message_id = first_message_id
rename commit_hash commit_hash_old
gen str40 commit_hash = commit_hash_old

save similar-mailbox-merged-not-accepted.dta, replace

clear all

import delimited "developer_id.csv", delim(",") varnames(1)

save developer_id.dta, replace

clear all

import delimited "response.csv", delim(",") varnames(1)

gen message_id = initial_message_id

save response.dta, replace

// ##############################################################
// ##                    generate date fields                  ##
// ##############################################################

clear all

cd "F:\mbox_analyse\StataInput"

// generate year, month and day from timestamp for emailRecords.dta
use emailRecords.dta

gen str timestampTemp = substr(timestamp, 5, 15) + " "
    + trim(substr(timestamp, 25, 5))
gen double timestampDouble = clock(timestampTemp, "MD hms Y")
format timestampDouble %tc

gen month = month(dofc(timestampDouble))
gen year = year(dofc(timestampDouble))
gen day = day(dofc(timestampDouble))

gen month_year = ym(year, month)
format month_year %tm

save emailRecords.dta, replace
```

Listing A.2: STATA-Skript, Teil 2

```
clear all

cd "F:\mbox_analyse\StataInput"

// generate year, month and day from timestamp for commitRecords.dta
use commitRecords.dta

gen str timestampTemp = substr(timestamp, 5, 15) + "␣"
    + trim(substr(timestamp, 25, 5))
gen double timestampDouble = clock(timestampTemp, "MD␣hms␣Y")
format timestampDouble %tc

gen month = month(dofc(timestampDouble))
gen year = year(dofc(timestampDouble))
gen day = day(dofc(timestampDouble))

gen month_year = ym(year, month)
format month_year %tm

save commitRecords.dta, replace

// ##################################################################
// ##                merge data sets                               ##
// ##################################################################

clear all

cd "F:\mbox_analyse\StataInput"

// merge maintainer to developer
use developer_id.dta

mmerge developer_id using maintainerMerged_removedDuplicates,
    type(n:1)
save developerMaintainer.dta, replace

// merge all developer and maintainer to email records
use emailRecords.dta

mmerge developer_id using developerMaintainer, type(n:1)
save emailRecordsDeveloper.dta, replace

// merge response time to emailRecordsDeveloper
use emailRecordsDeveloper.dta

mmerge message_id using response.dta, type(n:1)

save emailRecordsDeveloper.dta, replace

// merge similar-mailbox to email-records
use emailRecordsDeveloper.dta

drop if missing(message_id)
duplicates drop message_id, force
```

Listing A.3: STATA-Skript, Teil 3

```
save emailRecordsDeveloperDrop.dta, replace

use similar_mailbox_merged_not_accepted.dta

mmerge message_id using emailRecordsDeveloperDrop.dta, type(n:1)

save similar_mailbox_merged_not_accepted_emails, replace

// merge similar_mailbox to commit records
use commitRecords.dta

mmerge commit_hash using similar_mailbox_merged_only_accepted,
    type(n:1)

// generate min / max values of patch_message_count
gen patch_message_count_min = patch_message_count
gen patch_message_count_max = patch_message_count

save commitRecordsSimilar.dta, replace

// ################################################################
// ##                  collapse data                             ##
// ################################################################

clear all

cd "F:\mbox_analyse\StataInput"

// collapse similar_mailbox emailrecords
use similar_mailbox_merged_not_accepted_emails.dta

gen not_accepted_count = 1 if commit_hash == ""
gen accepted_count = 1 if commit_hash != ""

collapse (sum) not_accepted_count accepted_count, by
    (month_year developer_id month year)

save collapsedSimilarMonthNotAccepted.dta, replace

// collapse commitSimilar data by month and developer_id
use commitRecordsSimilar.dta

gen commit_count = 1

collapse (sum) changed_files commit_count added_lines deleted_lines
    diff_size change_on_function (mean) patch_message_count
    acceptance_rate (min) patch_message_count_min (max)
    patch_message_count_max, by (month_year developer_id month year)
sort developer_id month_year

save colpasedCommitRecordsSimilarMonth.dta, replace
```

Listing A.4: STATA-Skript, Teil 4

```
1  // collapse commitSimilar data by month and developer_id
2  use emailRecordsDeveloper.dta
3
4  gen email_count = 1
5
6  collapse (sum) mail_to_torvalds email_count mail_to_morton
7      mail_to_kroah_hartman mail_to_viro mail_to_molnar mail_to_miller
8      mail_to_iwai mail_to_zolnierkiewicz initial_mail count_cc count_to
9      count_from count_reply_to count_attachments (max) maintainer
10     (mean) private_mail_provider top_company is_edu minutes, by
11     (month_year month year developer_id)
12 sort developer_id month_year
13
14 save colpasedEmailRecordsDeveloper.dta, replace
15
16 // ##############################################################
17 // ##               merge collapsed data                       ##
18 // ##############################################################
19
20 cd "F:\mbox_analyse\StataInput"
21
22 use colpasedEmailRecordsDeveloper.dta
23
24 mmerge developer_id month_year month year using
25     colpasedCommitRecordsSimilarMonth
26
27 mmerge developer_id month_year month year using
28     collapsedSimilarMonthNotAccepted
29
30 egen total_mail_to_maintainer = rowtotal(mail_to_torvalds
31     mail_to_morton mail_to_kroah_hartman mail_to_viro mail_to_molnar
32     mail_to_miller mail_to_iwai mail_to_zolnierkiewicz)
33
34 // fill missings
35
36 replace mail_to_torvalds = 0 if missing(mail_to_torvalds)
37 replace mail_to_morton = 0 if missing(mail_to_morton)
38 replace mail_to_kroah_hartman = 0 if missing(mail_to_kroah_hartman)
39 replace mail_to_viro = 0 if missing(mail_to_viro)
40 replace mail_to_molnar = 0 if missing(mail_to_molnar)
41 replace mail_to_miller = 0 if missing(mail_to_miller)
42 replace mail_to_iwai = 0 if missing(mail_to_iwai)
43 replace mail_to_zolnierkiewicz = 0 if missing(mail_to_zolnierkiewicz)
44
45 replace email_count = 0 if missing(email_count)
46
47 save complete.dta, replace
48 outsheet using "complete.csv", replace comma
```

Listing A.5: STATA-Skript, Teil 5

```
// ###################################################################
// ##                  stcox                                        ##
// ###################################################################

cls
clear all

cd "F:\mbox_analyse\StataInput"

use complete.dta

drop if missing(acceptance_rate)
drop if missing(minutes)

gen rel_total_mail_to_maintainer = total_mail_to_maintainer
    / email_count
gen rel_top_company = top_company / email_count

// standardize vars

set varabbrev off
global varlist commit_count acceptance_rate email_count
    rel_total_mail_to_maintainer minutes diff_size patch_message_count
    rel_top_company
foreach e in $varlist {
egen std_`e' = std(`e')
local label : variable label `e'
label variable std_`e' `"`label'"'
}

gen commit_count_x_acceptance_rate = std_commit_count *
    std_acceptance_rate
gen email_count_x_total_mail_to = std_email_count *
    std_rel_total_mail_to_maintainer

sort developer_id month_year

egen uniqueDevelopers = group(developer_id)
sum uniqueDevelopers

stset month_year, id(developer_id) failure(maintainer==1)
```

Listing A.6: STATA-Skript, Teil 6

```
1  // Model 5 - all variables
2
3  stcox std_commit_count std_acceptance_rate
4      commit_count_x_acceptance_rate std_email_count
5      std_rel_total_mail_to_maintainer email_count_x_total_mail_to
6      std_minutes std_diff_size std_patch_message_count
7      std_rel_top_company, nohr vce(cluster developer_id)
8  mkcorr std_commit_count std_acceptance_rate
9      commit_count_x_acceptance_rate std_email_count
10     std_rel_total_mail_to_maintainer email_count_x_total_mail_to
11     std_minutes std_diff_size std_patch_message_count
12     std_rel_top_company, log(mkcorr) replace sig
13
14 estat ic
15 matrix out=r(S)
16 local AIC=out[1,5]
17 local BIC=out[1,6]
18
19 outreg2 using outreg.doc, replace adds("Wald␣chi2␣(`=e(df_m)')␣",
20     e(chi2), Prob > chi2, e(p), AIC, `AIC',␣BIC,␣`BIC') label
21     ctitle(Model 5) drop(i.id) dec(2)
22
23 // Model 2 - only direct variables
24
25 stcox std_commit_count std_acceptance_rate std_email_coun
26      std_rel_total_mail_to_maintainer std_minutes std_diff_size
27      std_patch_message_count std_rel_top_company,
28      nohr vce(cluster developer_id)
29
30 outreg2 using outreg.doc, append adds("Wald␣chi2␣(`=e(df_m)')␣",
31     e(chi2), Prob > chi2, e(p), AIC, `AIC',␣BIC,␣`BIC') label
32     ctitle(Model 2) drop(i.id) dec(2)
33
34 // Model 3 - technical direct and mod variables
35
36 stcox std_commit_count std_acceptance_rate
37     commit_count_x_acceptance_rate std_minutes std_diff_size
38     std_patch_message_count std_rel_top_company,
39     nohr vce(cluster developer_id)
40
41 outreg2 using outreg.doc, append adds("Wald␣chi2␣(`=e(df_m)')␣",
42     e(chi2), Prob > chi2, e(p), AIC, `AIC',␣BIC,␣`BIC') label
43     ctitle(Model 3) drop(i.id) dec(2)
```

Listing A.7: STATA-Skript, Teil 7

```
1  // Model 4 - communication direct and mod variables
2
3  stcox std_email_count std_rel_total_mail_to_maintainer
4  email_count_x_total_mail_to std_minutes std_diff_size
5  std_patch_message_count std_rel_top_company,
6  nohr vce(cluster developer_id)
7
8  outreg2 using outreg.doc, append adds("Wald chi2 (`=e(df_m)') ",
9  e(chi2), Prob > chi2, e(p), AIC, `AIC', BIC, `BIC') label
10 ctitle(Model 4) drop(i.id) dec(2)
11
12 // Model 1 - only control variables
13 stcox std_minutes std_diff_size std_patch_message_count
14     std_rel_top_company, nohr vce(cluster developer_id)
15
16 outreg2 using outreg.doc, append adds("Wald chi2 (`=e(df_m)') ",
17     e(chi2), Prob > chi2, e(p), AIC, `AIC', BIC, `BIC') label
18     ctitle(Model 1) drop(i.id) dec(2)
```

Listing A.8: STATA-Skript, Teil 8

Literaturverzeichnis

[Bec] Matthias Becker. Linux Top 10 mit Neueinsteiger: Doch neue Distro ist gar kein Linux. Webseite. Online verfügbar auf http://www.chip.de/news/Bestes-Linux-Das-ist-die-Top-10-der-beliebtesten-Distributionen_100413102.html; zuletzt besucht am 28.11.2017. (zitiert auf Seite 3)

[Ber16] Stefan Berreiter. The Usual Suspects? Developers' Progression from Periphery to Core in the Linux Kernel Development Project. Bachelorarbeit, Universität Passau, Deutschland, September 2016. (zitiert auf Seite 52)

[Bis17] Rahul Biswal. Top 10 Best Free Email Service Providers For You. Webseite, March 2017. Online verfügbar auf http://www.ecloudbuzz.com/best-free-email-service-providers/; zuletzt besucht am 13.05.2017. (zitiert auf Seite viii und 22)

[Coq] Comparison of webmail providers. Webseite. Online verfügbar auf https://en.wikipedia.org/wiki/Comparison_of_webmail_providers; zuletzt besucht am 13.05.2017. (zitiert auf Seite viii und 22)

[DO11] Linus Dahlander and Siobhan O'Mahony. Progressing to the Center: Coordinating Project Work. *Organization Science*, 22(4):961—-979, 2011. (zitiert auf Seite 8, 9 und 50)

[Fou] The Linux Foundation. What is Linux? Webseite. Online verfügbar auf https://www.linux.com/what-is-linux; zuletzt besucht am 03.05.2017. (zitiert auf Seite 3)

[Fou10] The Linux Foundation. Linux Kernel Development. https://www.linuxfoundation.org/docs/lf_linux_kernel_development_2010.pdf, 2010. (zitiert auf Seite viii, 4 und 23)

[JH08] J. Y. Moon & C. Zhang J. Hahn. Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties. *Information Systems Research*, 19(3):369—-391, 2008. (zitiert auf Seite 9)

[Jon] M. Jones. Anatomy of the Linux kernel. Webseite. Online verfügbar auf https://www.ibm.com/developerworks/library/l-linux-kernel/index.html; zuletzt besucht am 02.01.2018. (zitiert auf Seite 6)

[NWGPR08] Ed. Network Working Group P. Resnick. Internet Message Format. Webseite, October 2008. Online verfügbar auf https://tools.ietf.org/html/rfc5322; zuletzt besucht am 13.05.2017. (zitiert auf Seite viii und 18)

[proa] Advanced Linux Sound Architecture (ALSA) project. Advanced Linux Sound Architecture (ALSA) project homepage. Webseite. Online verfügbar auf https://www.alsa-project.org/main/index.php/Main_Page; zuletzt besucht am 02.01.2018. (zitiert auf Seite 6)

[Prob] The Linux Counter Project. Lines of code of the Linux Kernel Versions. Webseite. Online verfügbar auf https://www.linuxcounter.net/statistics/kernel; zuletzt besucht am 20.08.2017. (zitiert auf Seite viii und 4)

[ras] raspbian.org. Welcome to Raspbian. Webseite. Online verfügbar auf http://www.raspbian.org/FrontPage; zuletzt besucht am 28.11.2017. (zitiert auf Seite 3)

[RR16] Wolfgang Mauerer Ralf Ramsauer, Daniel Lohmann. Observing custom software modifications: A quantitative approach of tracking the evolution of patch stacks, 04 2016. (zitiert auf Seite 9 und 35)

[WSS+16] Igor Wiese, Jose Silva, Igor Steinmacher, Christoph Treude, and Marco Aurelio Gerosa. Who is who in the mailing list? comparing six disambiguation heuristics to identify multiple addresses of a participant, 10 2016. (zitiert auf Seite 13, 22 und 31)

”

Eidesstattliche Erklärung:

Hiermit versichere ich an Eides statt, dass ich diese Masterarbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie dass ich die Masterarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Name

Passau, den 9. Februar 2018