Institut für Softwaretechnologie
Abteilung Software Engineering II
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 14

Analyse des Wartungsaufwandes von aktiven Produkten

Rick Krüger

Studiengang:	Informatik
Prüfer:	Prof. Dr. rer. nat. Stefan Wagner
Betreuer:	Dipl.-Inf. Jan-Peter Ostberg
begonnen am:	04. Juni 2012
beendet am:	04. Dezember 2012
CR-Klassifikation:	D.2.7, D.2.8

Kurzfassung

In dieser Arbeit werden Metriken zur Analyse von Software-Wartungsaufwänden vorgestellt und mithilfe von Daten aus der aktiven Wartung angewandt. Des Weiteren werden Metriken über die Datenqualität von Systemen zur Wartungsaufwandserfassung aufgezeigt und ebenfalls bei den vorhandenen Realdaten eingesetzt. Dabei wird ein realistisches Bild der Wartungsaufwände fern der statischen Codeanalyse gezeichnet.

INHALTSVERZEICHNIS

Abbildungsverzeichnis

KAPITEL 1

EINLEITUNG

Jedes Softwareprojekt, das eine gewisse Größe überschreitet und sich über längere Zeit in einem aktiven Nutzungs- und Wartungsprozess befindet, wird früher oder später Wartungsaufwände erzeugen, da es weder Code gibt, der zu 100 Prozent frei von Fehlern ist, noch davon auszugehen ist, dass sich die Anforderungen an eine Software nicht verändern.

Den Wartungsaufwand einer Software abschätzen zu können, ist dabei nicht nur von theoretischem Interesse. So kann ein hoher Wartungsaufwand (gerade im korrektiven Bereich) die Entwicklung anderer, neuer Produkte oder auch die Weiterentwicklung, also die adaptive Wartung, des Produktes zum Erliegen bringen. Oftmals fällt es schwer, den genauen Ort und die Menge der anfallenden Wartungsaufwände abzuschätzen. Gerade in einer laufenden Softwareentwicklung, die nicht von Beginn an mit den Grundsätze des Software Engineerings durchgeführt wurde, ist dies ein Problem, da es häufig keine konsequente Analyse der Softwarequalität gibt. Das Ziel dieser Arbeit soll nun sein, den Wartungsaufwand eines aktiven Produktes zu analysieren und somit die Möglichkeit der präzisen Optimierung des Softwareentwicklungsprozesses zu geben.

Im Gegensatz zu Analysen mithilfe von geänderten Zeilen (LOC) oder vergleichbaren Metriken, die die Analyse über den Programmquelltext durchführen und dementsprechend Analysen über die Wartbarkeit, ergo einer theoretische Größe, durchführen, ist in dieser Arbeit die Verwertung der Angaben aus dem aktiven Wartungsprozess vorgesehen und es wird somit eine Analyse der expliziten Wartungszeiten durchgeführt. Dabei werden Realdaten verwendet, welche aus jeweils einer Installation des Issue-System JIRA und dem Ticket-System OTRS entnommen werden, um Metriken zu entwickeln, die sich für die Analyse dieser Daten eignen. Dies stellt somit eine Ergänzung zu den bisherigen statischen Code-Analysen dar und soll so detailliertere Ergebnisse über die Wartung innerhalb eines Unternehmens geben.

Es soll im Folgenden auch gezeigt werden, inwieweit Aussagen über den Wartungsaufwand eines Produktes, welches sich im aktiven Nutzungsprozess befindet, mit diesen gemacht werden können, also inwiefern sich der Wartungsaufwand anhand dieser Metriken möglich ist. Da die Analyse anhand von real angefallenen Wartungsdaten durchgeführt wird, sind auch Vorschläge zur Optimierung der Datenqualität Teil der Arbeit.

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen: Hier werden werden die Grundlagen dieser Arbeit beschrieben und definiert.

Kapitel 3 – Über die zu analysierenden Daten stellt den Industriepartner und dessen Wartungsprozesse vor.

Kapitel 4 – Metriken: Hier werden die verwendeten Metriken und die dahinterliegenden Fragestellungen erläutert.

Kapitel 5 – Schaffung der technischen Grundlagen beschreibt, wie auf die Daten zugegriffen wurde und dieser verarbeitet wurden.

Kapitel 6 – Analyse der Wartungsaufwände enthält die Analysen anhand der vorher erarbeiteten Metriken

Kapitel 7 – Analyse und Verbesserungsvorschläge der Datenqualität zeigt Möglichkeiten auf, wie die Qualität der analysierten Daten zu bewerten ist

Kapitel 8 – Zusammenfassung und Ausblick fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

KAPITEL 2

GRUNDLAGEN

Für das Verständnis der Arbeit sind einige Begriffe vonnöten, die hier im Folgenden erläutert werden sollen. Im Allgemeinen werden die von Ludewig und Lichter in Software-Engineering [Lud10] verwendeten Definitionen übernommen. Für weitere Informationen zu einzelnen Stichpunkten ist auf dieses Werk verwiesen. Dennoch sollen hier die Definitionen der Vollständigkeit halber hier noch einmal dargelegt werden.

2.1. Software

Das IEEE hat folgende Definition von Software:

> **Software** - Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
> (IEEE Std 610.12 (1990) [IEE90])

Hervorzuheben ist an dieser Definition, dass explizit auch die Dokumentation erwähnt wird, womit sich Wartungsaufwände auch auf eine Verbesserung der Dokumentation beziehen können. Explizit soll der Begriff Dokumentation erläutert werden

2.2. Dokumentation

Die Dokumentation ist ein fester Bestandteil einer Software und kann nicht von dieser getrennt gedacht sein. Dabei ist im Allgemeinen zwischen zwei Arten der Dokumentation zu unterscheiden:

- **Integrierte Dokumentation** – Dies bezeichnet alle im Quelltext vorhandenen Kommentare bzw. Hinweise auf die Funktionsweise des Programms. Dabei seien auch Variablenbezeichner gemeint, die ein besseres Verständnis erlauben.
- **Separate Dokumentation** – Die separate Dokumentation sind die nicht im Quelltext enthaltenen Dokumente, die ein Verständnis der Software erlauben sollen.

Dabei kann die separate Dokumentation in Einzelfällen, wie beispielsweise der Dokumentation einer Schnittstelle direkt aus den im Quelltext enthaltenen Kommentaren generiert sein. Während bei der integrierten Dokumentation der Fokus auf der Wartbarkeit der Software liegt, ist dieser bei der separaten Dokumentation mehr auf ein Verständnis der Bedienung gelegt. Daher wird im weiteren Verlauf dieser Arbeit keine Analyse über die integrierte Dokumentation stattfinden.

2.3. Fehler

Ein Fehler in einer Software ist im Allgemeinen als eine nicht der Spezifikation der Software entsprechende Funktionsweise eines Programms. Das IEEE führt dies in [IEE10] wie folgt aus:

> **defect:** An imperfection or deficiency in a work product where that work product does not meet its requirements or specifications and needs to be either repaired or replaced. (adapted from the Project Management Institute)
>
> NOTE—Examples include such things as 1) omissions and imperfections found during early life cycle phases and 2) faults contained in software sufficiently mature for test or operation.
>
> **error:** A human action that produces an incorrect result. (adapted from ISO/IEC 24765:2009)
>
> **failure:** (A) Termination of the ability of a product to perform a required function or its inability to perform within previously specified limits. (adapted from ISO/IEC 25000:2005) (B) An event in which a system or system component does not perform a required function within specified limits. (adapted from ISO/IEC 24765:2009)
>
> NOTE—A failure may be produced when a fault is encountered.
>
> **fault:** A manifestation of an error in software. (adapted from ISO/IEC 24765:2009) problem: (A) Difficulty or uncertainty experienced by one or more persons, resulting from an unsatisfactory encounter with a system in use. (B) A negative situation to overcome. (adapted from ISO/IEC 24765:2009)
>
> (IEEE Std 1044-2009)

Dabei ist explizit festgelegt, dass ein Fehler (error) ein falsches Ergebnis aufgrund einer Eingabe durch den Benutzer bezeichnet, während der klassische Bug ein Defekt (failure) ist. Diese Definition soll auch im Folgenden gelten, das heißt, dass ein Fehler im Allgemeinen zu einem Wartungsfall führt, allerdings ein Fehler nicht unbedingt zu der Behebung eines Defekts führen muss, da für das Auftreten eines Fehlers kein Defekt vorliegen muss.

2.4. Software-Wartung

Software kann, wie von Ludewig und Lichter angemerkt (Seite 566), nicht altern, daher ist unter der Wartung von Software im Allgemeinen eher von einer Optimierung zu sprechen, sei es, um Fehler zu entfernen oder die Software an veränderte Gegebenheiten anzupassen. Bei der Definition für Software-Wartung schließen wir uns der Definition durch Ludewig und Lichter an, die wie folgt lautet:

> **Software-Wartung** ist jede Arbeit an einem bestehenden Softwaresystem, die nicht von Beginn der Entwicklung geplant war oder hätte geplant werden können und die unmittelbare Auswirkungen auf den Benutzer der Software hat.

Dabei gehen wir im Folgenden aufgrund der Lebensdauer der betrachteten Produkte davon aus, dass die Arbeit, die an den Softwaresystemen durchgeführt wird, nicht von Anfang an geplant wurde und es sich demnach bei allen Daten die betrachtet werden sollen um Wartungsdaten handelt.

Das IEEE unterscheidet die folgenden Wartungstypen bei der Software-Wartung:

> **adaptive maintenance** – Software maintenance performed to make a computer program usable in a changed environment.
>
> **corrective maintenance** – Maintenance performed to correct faults in Software.
>
> **perfective maintenance** – Software maintenance performed to improve the performace, maintainability, or other attributes of a computer program.
>
> **preventive maintenance** – Maintenance performed for the purpose of preventing problems before they occur. (IEEE Std 610.12 (1990) [IEE10])

Über die letzten beiden Punkte und ihren Sinn wurde bereits in dem Buch von Ludewig und Lichter[Lud10, Seite 567 ff.]

Im Folgenden sollen auch die vom Industriepartner verwendeten Begriffe verwendet werden. Dabei gibt es folgende zwei Definitionen für Fehlerbehebung:

- **Bug** – Ein Defekt, der reproduzierbar ein nicht erwünschtes Verhalten produziert, wird behoben
- **Incomplete Specification Implementation** – Das Nachrüsten von Funktionalitäten, die aus Kundensicht unzureichend implementiert waren. Ein Beispiel dafür ist die Erkennung von Barcodes. Die vorliegende Software kann einen Großteil der weltweit verwendeten Barcodes lesen, es kann aber sein, dass ein Kunde einen nicht implementierten Barcode verwenden will. In diesem Fall ist die Fähigkeit in Grundsätzen vorhanden, aber unzureichend implementiert
- **Improvement** – Die Erweiterung oder Verbesserung einer bestehenden Funktionalität
- **New Feature** – Die Einführung einer neuen Funktionalität

- **Wish** – Die Einführung einer neuen Funktionalität auf Kunden- oder internen Wunsch
- **Documentation** – Arbeiten an der Dokumentation, in diesem Fall der separaten Dokumentation
- **Task / Sub-Task** – Dienen zur Strukturierung von Funktionalitätseinführungen

Während die erste Variante für Fehler direkt mit der corrective maintenance korreliert, sind die zweite und dritte Variante problematisch, da sie Ähnlichkeiten mit der perfective maintenance, die von Ludewig und Lichter zu Recht kritisiert wurde, besitzen. Denn entweder wurde ein ungenügendes Produkt an den Kunden ausgeliefert, wodurch es sich um einen Defekt handelt, oder, falls die Spezifikation ein Feature nicht vorausgesetzt hat, es sich um ein neues Feature handelt, was dementsprechend der adaptiven Wartung entspräche. Da es diese Klassifizierungen allerdings innerhalb der Daten gibt und eine Einteilung der betroffenen Wartungsfälle den Rahmen der Analyse übersteigt, müssen diese Klassen als solche akzeptiert werden. Es ist allerdings angeregt worden, die Definition für Wartungsfälle den von Ludewig und Lichter als sinnvoll erachteten Kategorien anzupassen, um so zukünftig nach Maßstäben des Software-Engineerings korrekte Analysen zu ermöglichen.

New Feature, wobei Task und Sub-Task hinzugezählt werden, und Wish sind in die Klasse der adaptive maintenance einzuordnen, da bei beiden das Anforderungsprofil der Software geändert wurde.

Bei der Wartungsarbeit Documentation handelt es sich, wie bereits angemerkt um die Arbeiten an der separaten Dokumentation und ist daher als Wartungsfall nicht in die Definitionen einzuordnen, sondern ist den jeweiligen Wartungsfällen zuzuordnen, die eine Veränderung der Dokumentation geführt haben.

Die Einsortierung von Wartungstypen nach dem IEEE ist in der Realität des Öfteren ein Problem, in den vorliegenden Daten sind diese nahe den Standards, allerdings wurde in [Opfo4] darauf hingewiesen, dass Wartungsfälle oft nur nach geschätzter Länge und nicht nach Typ einsortiert werden.

2.5. Wartbarkeit von Software

Auch wenn auf die Wartbarkeit der betrachteten Produkte in dieser Arbeit nicht weiter eingegangen werden soll, so soll sie doch zur Abgrenzung zum Terminus des Wartungsaufwandes definiert sein.

Die Wartbarkeit einer Software ist eine theoretische Größe die Aussagen darüber treffen soll, wie einfach und zeitintensiv die Wartung einer Software vonstatten geht, falls eine Wartung notwendig wird. Dabei werden verschiedene Metriken zur Komplexitäts- und Übersichtlichkeitsanalyse (Lesbarkeit) des Quelltextes verwendet, wie beispielsweise Lines-of-Code (LOC) oder der Halstead-Metrik.

Die Wartbarkeit ist erst in dem Moment für die Wartung relevant, in dem ein Wartungsfall eintritt. In diesem Fall kann eine schlechte Wartbarkeit den Prozess der Wartung verlangsamen. Durch die Tatsache, dass es sich um eine theoretische Größe handelt, kann

im Vergleich zur Analyse über die real anfallenden Wartungsvorgänge nur teilweise eine Aussage über die Software-Wartung getroffen werden, während natürlich eine Analyse der Wartungsaufwände nur teilweise Rückschlüsse auf die Wartbarkeit zulässt.

2.6. Wartungsaufwand

Mit Wartungsaufwand soll im Folgenden die real anfallende Zeit bezeichnet werden, die für Wartung verwendet wird. Dabei soll eine Unterscheidung in die zwei folgenden Kategorien für Wartungsaufwände getroffen werden (nicht mit den Kategorien für Wartungsfälle zu verwechseln):

- **Firmenrelevanter Wartungsaufwand** – dies bezeichnet sämtliche Aufwände, die in Support, Entwicklung und Dokumentation anfallen
- **Softwarerelevanter Wartungsaufwand** – umfasst nur den Teil des firmenrelevanten Wartungsaufwandes, der zu einer konkreten Änderung an der Software nach der Definition von Software-Wartung führt

Diese Unterteilung erlaubt die Klassifizierung der anfallenden Zeiten, die für eine Bearbeitung von Support-Fällen aufgewendet wird und soll im Folgenden auch Hinweise auf eventuell notwendige Überarbeitungen der Dokumentation liefern.

2.7. Wartungsfall

Ein Wartungsfall ist definiert als ein in sich abgeschlossener Vorgang, der mit der Kontaktierung des Supports durch einen Kunden begonnen wird, danach im Support bearbeitet wird, bei Bedarf die Korrektur eines potentiellen Fehlers umfasst und schlussendlich mit der Nachbearbeitung durch den Support endet.

2.8. Ticket / Issue

Im Rahmen dieser Arbeit ist häufig von den zwei Begriffen Ticket und Issue die Rede. Dabei bezeichnet ein Ticket eine durch den Support abgelegte Kundenanfrage im Trouble-Ticket-System, während ein Issue einen Eintrag im Issue-Tracking-System einer Firma bezeichnet. Dieser kann, muss aber keine Verbindung zu einem Ticket besitzen und kann in verschiedene Unterarten aufgeteilt werden beispielsweise einem Bug, aber auch einem neuen Feature und erlaubt so eine bessere Unterscheidung als im klassischen Bug-Tracking.

2.9. Aktive Produkte

Ein aktives Produkt ist eine Software oder eine Softwareversion, welche sich im firmeninternen oder -externen Gebrauch befindet und zu dem es im letzteren Fall bestehende Wartungsverträge gibt. Ein aktives Produkt kann mehrere aktive Versionen beinhalten, wobei auch in diesem Fall nur in Benutzung befindliche Versionen mit Wartungsverträgen über die jeweilige Versionen als aktiv gelten.

2.10. Anmerkungen

Es werden im Folgenden nur abgeschlossene Wartungsfälle zur Analyse herangezogen, da bei noch nicht abgeschlossenen Vorgängen keine Aussage über endgültige Zeiten getroffen werden kann. Allerdings ist anzumerken, dass auch bereits geschlossene Wartungsfälle in Einzelfällen durch Wiederauftauchen eines Fehlers erneut geöffnet werden können und somit die erfassten Zeiten dennoch nicht den endgültigen auf den gesamten Prozess angefallenen Zeiten entsprechen müssen.

Alle Aussagen, die im Laufe der Arbeit getroffen werden, sind in Relation zu der Größe der Software zu sehen, das bedeutet, alle absoluten Zahlen in Verhältnis zu der Anzahl der Codezeilen zu sehen sind. Dies erlaubt einen Vergleich verschiedener Wartungsaufwände verschiedener Firmen und verschiedener Produkte. Für die interne Analyse, wie sie in diesem Fall durchgeführt wurde, ist eine Darstellung der absoluten Zahlen sinnvoller.

KAPITEL 3

ÜBER DIE ZU ANALYSIERENDEN DATEN

Die Analyse des Wartungsaufwände findet an Realdaten statt, die bei der Wartung einer Software entstanden sind, die sich bereits mehr als 10 Jahre in einem aktiven Entwicklungs- und Verwendungsprozess befindet. Die Software wird dabei für verschiedene Plattformen entwickelt und enthält ca. 5 Mio. Zeilen C-Code.

Die Software als solche ist auch durch die mit der Zeit gewachsenen Strukturen innerhalb der Firma interessant, da diese im Laufe des Entwicklungsprozesses regelmäßig geändert wurden und somit auch auf die Wartungsprozesse als solche Einfluss genommen haben.

Durch diese Faktoren ist die Software prädestiniert für eine Analyse.

Von derselben Firma werden auch Produkte ausgeliefert, die in Java entwickelt werden, allerdings befinden sich diese noch nicht lange in einem aktiven Wartungsprozess, sprich kürzer als ein Jahr, wodurch eine Analyse von dieser Software keinen großen Nutzen hätte, da die Ergebnisse nicht einfach einzuordnen wären und noch in einem zu großen Maße schwanken.

Bevor mit der eigentlichen Analyse begonnen werden kann, muss der Wartungsprozess innerhalb der Firma als solches dargelegt werden. Dafür dient das Diagramm 6.7.

Wie ersichtlich ist, existiert ein zweigliedriger Support, bei dem der 1st-Level-Support versucht, simple Probleme direkt zu lösen und in komplexeren Fällen das Problem möglichst gut zu reproduzieren, während der 2nd-Level-Support Rücksprache mit der Entwicklungsabteilung halten kann und so auch komplexere Probleme zu einer Lösung bringen kann. Dabei versucht der 2nd-Level-Support auch, das Problem möglichst zu reduzieren, das bedeutet, dass bei einer größeren Datei, die durch die Software verarbeitet werden soll, die betreffende Stelle wenn möglich isoliert werden soll, um die Übersichtlichkeit des Problems für die Entwicklungsabteilung zu steigern.

Der Support verwendet für die Verwaltung der Supportanfragen das Ticket-System OTRS [AG], während in der Entwicklungs- und der Dokumentationsabteilung hingegen wird JIRA [Atlb], ein Issue-System, verwendet, um die anfallenden Zeiten zu erfassen. Diese beiden Systeme werden im Kapitel 5 – Schaffung der technischen Grundlagen näher erläutert und der Zugriff auf diese geschildert.

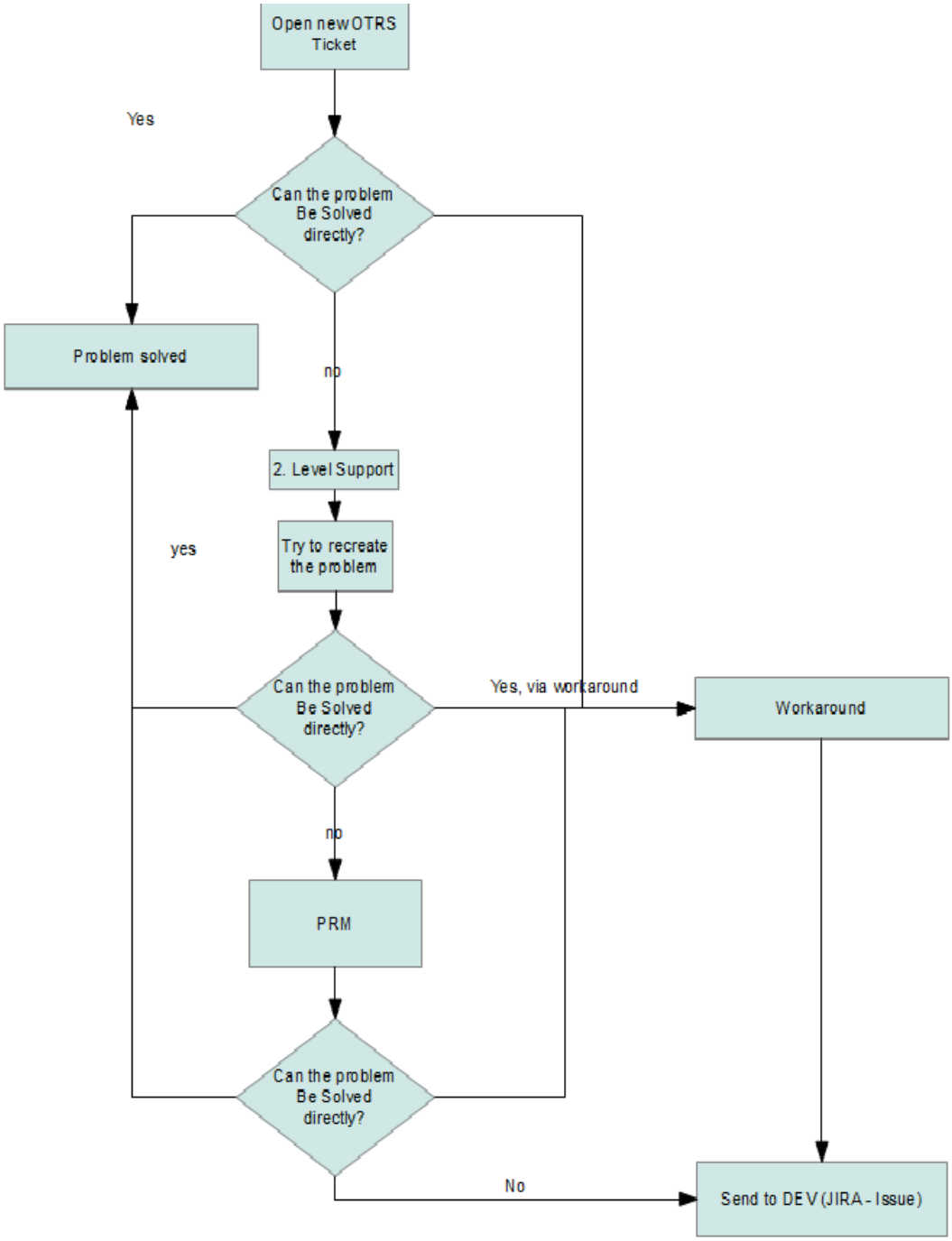

Abbildung 3.1.: Workflow Wartungsprozess

KAPITEL 4

METRIKEN

Das Ziel dieser Arbeit soll nicht nur die Analyse von Wartungsdaten einer Firma sein, sondern auch auf die Wartungsprozesse anderer Firmen verallgemeinert werden können. Zu diesem Zwecke ist es wichtig, die gemessenen Daten mithilfe von Metriken zu standardisieren. Eine Metrik definiert sich dabei nach dem IEEE wie folgt:

> **metric** – A quantitative measure of the degree to which a system, component, or proccess possesses a given attribute
>
> See also: quality metric
>
> **quality metric** – (1) A quantitative measure of the degree to which an item possesses a given quality attribute.
>
> (2) A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.
>
> (IEEE Std 610.12 (1990) [IEE10])

Damit ist festgelegt, dass bei einer Metrik quanitativ die Qualität eines Merkmals eingestuft werden soll.

Im Folgenden sollen nun Metriken vorgestellt werden, die zur Analyse des Wartungsaufwandes bei aktiven Produkten anhand vorhandener Daten aus Ticket-Systemen verwendet werden können.

Dabei werden einzelne Metriken zur Darstellung von verschiedenen qualitativen und quantitativen Faktoren der Software-Wartung definiert, die zusammen ein Gesamtbild der Wartung ergeben. Es bleibt abzuwägen, ob eine Gesamtmetrik, welche die einzelnen Faktoren bündelt, sinnvoll ist, oder ob zur Gesamtanalyse nicht die Bewertung der einzelnen Punkte auf jeweils getrennten Skalen sinnvoll ist, wie sie auch in dieser Arbeit durchgeführt wird.

Die Metriken dienen einerseits zur Erfassung aktueller Wartungsdaten, können aber auch dazu dienen, den Aufwand von zukünftigen Versionen abzuschätzen, falls am Entwicklungsprozess keine grundsätzlichen Änderungen vorgenommen wurden, also von einer vergleichbaren Softwarequalität mit ähnlichem Wartungsaufwand auszugehen

ist oder es kann auch teilweise der Erfolg von neuen Maßnahmen abgeschätzt werden, nachdem diese für eine längere Zeit im Entwicklungsprozess eingesetzt wurden.

4.1. Goal-Question-Metric-Ansatz

Um die zu verwendenden Metriken auszuwählen, wurde nach dem Goal-Question-Metric-Ansatz (GQM) vorgegangen. Bei diesem werden im ersten Schritt die elementaren, abstrakten Fragen definiert, im nächsten Schritt die daraus resultierenden, konkreten Fragestellungen erarbeitet, die sich schlussendlich im letzten Schritt in Metriken ausdrücken lassen. Weiterführende Informationen zu GQM sind in [Bas92] zu finden.

Vorabbemerkungen:

- Falls ein Issue mehrere Versionen anbetrifft, wird die Zeit grundsätzlich der ersten betroffenen Version angerechnet. Im vereinfachten Modell nehmen wir an, dass ein Bugfix für eine Version sich auch in geringer Zeit auf eine spätere Version anwenden lässt. Da eben dieser Aufwand bisher nicht erfassbar ist, rechnen wir den Aufwand grundsätzlich der ersten betroffenen Version an.
- Fragestellungen, die den Code der untersuchten Software und dessen Qualität anbetreffen, sind im Folgenden ausgeklammert, da dies in den Bereich der statischen Codeanalyse fällt, welche allerdings, wie bereits angemerkt, nicht Teil dieser Arbeit sein soll.

Unser **Ziel** ist, den Wartungsaufwand eines Produktes zu erfassen, welches sich im Produktiveinsatz befindet.

Aus diesem Ziel lassen sich die folgenden Fragen ableiten:

1. Wie hoch ist der aktuelle Wartungsaufwand?
2. Warum fällt dieser Wartungsaufwand an?

Aus der 1. abstrakten Frage lassen sich mehrere konkrete Fragestellungen ableiten:

1. Gesamtzeit, die für eine Version an Wartungszeit aufgewendet wird
2. Durchschnittliche Zeit pro Wartungsvorgang (pro Version)
3. Anzahl an Tickets zu einer Version (zeitliche Verteilung)
4. Anzahl an Nachrichten zu einem Ticket

Die 2. Fragestellung ist ebenfalls abstrakt und lässt folgende detaillierteren Fragestellungen zu:

1. Wie ist die Verteilung auf die einzelnen Wartungsarten?
2. Gibt es Tickets im Support, die durch eine bessere Dokumentation verhindert werden können? (diese Fragestellung soll indirekt benötigten Wartungsaufwand aufzeigen, näheres weiter unten)

4.2. Metriken basierend auf JIRA

1. **Time per Version**: Zeit, die für alle Issues zu einer Version aufgewendet wurde.
2. **Time per Issue per Version**: Diese Metrik gibt eine Aussage über den Median oder Durchschnittswert der Bearbeitungszeit für ein Issue innerhalb einer Version. Dabei kann in die einzelnen Wartungstypen unterschieden werden
3. **Issues per Version**: Diese Metrik gibt eine statistische Zusammenfassung über die zu einer Version erstellten Tickets dar. Sie gibt noch keine genauen Daten über den Wartungsaufwand, da keine qualitative Aussagen über die Tickets getroffen werden und soll nur als Hilfsmetrik dienen. Eine Spezifizierung dieser Metrik erlaubt die Angabe der Issue-Art, sei es ein Bugfix oder eine Neuentwicklung, um detailliertere Statistiken über die Wartungsart zu geben.

Außerdem wird versucht, Aussagen über die Dokumentationsqualität / Handbuchqualität und dem nötigen Aufwand, diese zu verbessern, zu treffen. Dies ist relevant, da der zeitliche Aufwand, der auf den Support von mangelhaft dokumentierter Features abfällt, den Gesamtprozess lähmt. Einem hohen Supportaufwand in diesen Fällen kann und sollte durch eine bessere Dokumentation entgegengewirkt werden. Um Aussagen über diese Qualität und schlussendlich auch auf den nötigen Wartungsaufwand treffen zu können werden die Tickets betrachtet, die keine Verknüpfung zu einem JIRA-Issue besitzen und als „resolved“, sprich gelöst, gelistet sind. Dies bietet nur eine erste Annäherung, da wie bei der Analyse der Daten aus JIRA keine Analyse der eigentlichen Software, bzw. in diesem Fall der Dokumentation, vorgenommen wird. Des Weiteren ist natürlich anzumerken, dass es ebenso wie es nicht den perfekten Code einer Software geben kann, auch keine perfekte Dokumentation geben kann.

4.3. Metriken basierend auf OTRS

1. **Percentage of Tickets without Issues (per month)**: Bei dieser Metrik wird die Anzahl der Tickets, die keine Verknüpfung zu einem JIRA – Issue besitzen durch die gesamt auf ein Produkt innerhalb entfallenen Tickets geteilt. Dies kann zur statistischen Erfassung auch monatsweise geschehen.
2. **Messages per Ticket**: Bei dieser Metrik wird die durchschnittliche Anzahl an Nachrichten (bzw. der Median) dargestellt, die bei einem Ticket hin- und hergesendet werden. Dabei soll unterschieden werden in Tickets, die eine Verknüpfung zu einem Issue besitzen und, falls dies der Fall ist, in die einzelnen Phasen der Ticketbearbeitung aufgeteilt (Erstbearbeitung durch Support, Bearbeitung in Entwicklung, Nachbearbeitung im Support)

Des Weiteren soll im weiteren Verlauf versucht werden, den Gesamtprozess zu veranschaulichen.

Dabei soll der Gesamtfluss eines Tickets, das softwarerelevanten Wartungsaufwand erzeugt, dargelegt werden und mit den durchschnittlichen Verweildauern in den jeweiligen Abteilungen / Systemen verknüpft werden.

KAPITEL 5

SCHAFFUNG DER TECHNISCHEN GRUNDLAGEN

Nachdem nun im vorherigen Kapitel die benötigten Metriken entwickelt wurden, soll nun der erste Teil der praktischen Arbeit vorgenommen werden. Um für die Analyse Daten zu erlangen, müssen diese aus den verwendeten Systemen abgerufen werden. Um dies zu tun werden im Folgenden zwei verschiedene Varianten verwendet, für die eigentliche Auswertung und die Aufbereitung der Daten Eclipse-BIRT, welches beide Varianten unterstützt.

Die beiden Zugriffsvarianten sind auch für eine Verallgemeinerung der in dieser Arbeit vorgestellten Prozesse zur Analyse interessant, da sie bereits zwei der Möglichkeiten abdecken, wie auf benötigte Daten zugegriffen werden kann.

Abschließend soll in diesem Kapitel auch auf weitere Möglichkeiten eingegangen werden, wie benötigte Daten in BIRT zur Analyse eingepflegt werden können.

5.1. BIRT

Für die Analysen die im Rahmen dieser Arbeit durchgeführt wurden, wurde Eclipse BIRT (Business Intelligence and Reporting Tools) eingesetzt. BIRT erlaubt, unter anderem, die Einbindung verschiedener Datenquellen wie XML – Dateien, verschiedene Datenbanken per JDBC, Web Services und anderer Quellen via eigener JAVA – Klassen, um diese zu Reports zu verbinden.

BIRT ist ein Plugin für Eclipse und wurde von der Firma Actuate initiiert. Diese bietet auch Erweiterungen für BIRT an, die die Arbeit an Reports erleichtern sollen. BIRT bietet viele vorgefertigte Diagrammtypen, die durch simples Drag&Drop mit den zu verwenden Daten befüllt werden können und so den Arbeitsaufwand zur Erstellung eines Reports stark vereinfacht.

Die Arbeit mit den Daten ist dabei vielfältig möglich. Falls beispielsweise mehrere Datenquellen in einem SQL-ähnlichen, zeilenbasierten Rückgabeformat vorliegen können diese, wie für Datenbankabfragen üblich, per Join verbunden werden, etc. Auf der anderen Seite bietet BIRT, wie auch später verwendet, die Möglichkeiten, den gesamten

Zugriff in Eigenarbeit zu lösen und damit eine Vielzahl an komplexen Funktionalitäten bereitzustellen. Dieser Zugriff basiert auf Javascript zusammen mit Mozilla Rhino, das in diesem Fall erlaubt, JAVA-Objekte in Javascript anzusprechen.

5.2. JIRA

Nachdem JIRA bereits als das Issue-System vorgestellt wurde, soll hier näher auf die technischen Eigenschaften und die Installation innerhalb der Firma eingegangen werden.

JIRA ist eine JAVA-Applikation[Atlc] und wird zum Zeitpunkt dieser Arbeit in der Version 5 mit einer PostgreSQL-Datenbank eingesetzt. Der Zugriff auf die Daten des Issue-Systems soll über die bereitgestellte REST-API hergestellt werden, damit eine zukünftige Weiterverwendung der Reports bei einem Versionswechsel erleichtert wird.

Um die Daten aus JIRA zu verwerten, müssen Anfragen über die API in der JIRA Query Language (JQL)[Atla] gestellt werden. Für die API existiert eine Beispiel - JAVA-Client – Implementierung (JRJC[Atld]), die auch im Folgenden erweitert und verwendet wurde. JQL ist eine Sprache, die eng mit SQL verwandt ist. Allerdings wird auf Konstrukte für die Auswahl von Tabellen, Spalten und damit zusammenhängenden Funktionen wie JOINS verzichtet. Die Konzentration liegt auf den WHERE und ORDER BY – Klauseln.

Um die Verbindung zu dem System herzustellen, musste als erstes die gegebene Implementierung erweitert werden, um beispielsweise bei einer Anfrage sämtliche Datensätze und nicht nur die durch das System standardmäßig auf 1000 begrenzte Anzahl zu verwenden. Dafür wurden eigene abgeleitete Klassen angelegt, die im Falle, dass weniger Datensätze als die geforderte Anzahl zurückgeliefert wurden, eine weitere Anfrage, beginnend nach dem letzten Element der ersten, beginnt und diese Datensätze mit den vorher erhaltenen zu einer Menge verbindet.

Als nächstes wurde ein DAO (Data-Access-Object) geschaffen, das von BIRT zur Abfrage der Daten verwendet werden soll. Dieses stellt einzelne Methoden bereit, die zur Gewinnung der benötigten Daten dienen und bereitet diese auch schon nach den gegebenen Vorgaben auf, so dass bereits Gruppierungen nach Versionsnummer oder Wartungsart vorgenommen werden können.

Bei der Abfrage der einzelnen Issues aus dem System ist ein Nachteil des API-Zugriffs sichtbar geworden, und zwar erlaubt diese zwar, wie vorher genannt, die Abfrage von mehreren Datensätzen über eine JQL-Anfrage, allerdings sind die Issues, die zurückgegeben werden, unvollständig. Gerade Daten wie die eingetragenen Zeiten sind in diesen Antworten unvollständig und können leider nicht detaillierter angefragt werden. Daher ist es für diese Zwecke notwendig, jedes einzelne Issue anzufragen, um die benötigten Daten zu ermitteln. Dies führt zu einem großen zeitlichen Aufwand, der sich verringern ließe, wenn das System die direkte Angabe der benötigten Felder schon in einer JQL - Anfrage erlauben würde.

5.3. OTRS

Um die Daten aus dem OTRS zu verwerten, wurde der direkte Weg über die Datenbank des Systems gewählt. Dies ist zwar aus Gründen der einfachen Weiterverwendung nicht optimal, da aber die API der aktuell verwendeten Version (3.0.9) schlecht dokumentiert ist und das System in Zukunft durch ein CRM ersetzt werden soll, wurde aus Gründen der kürzeren Einarbeitungszeit der Weg über die Datenbank gewählt.

Als erstes wurde für die Verwertung ein Extrakt der Datenbank durch den Industriepartner bereitgestellt, welcher in einer virtuellen Maschine eingepflegt wurde, um somit eine Gefährdung der Daten des aktiven Systems zu verhindern.

Daraufhin wurden für die Extraktion der Daten Sichten angelegt, die eine Abstraktion der Tabellendaten erlauben und somit eine Vorverarbeitung der analysierten Daten mithilfe der Datenbanksoftware erlauben. Dies ist teilweise notwendig, da nicht die gesamte Komplexität der SQL - Abfragen durch BIRT unterstützt werden.

Abschließend wurde die Verbindung zu der in der virtuellen Maschine befindlichen Datenbank hergestellt und die Daten konnten in BIRT verwendet werden.

5.4. Weitere Möglichkeiten des Datenimports

Neben den bereits vorgestellten Varianten, Daten zu importieren, erlaubt BIRT auch beispielsweise XML-Dateien, Tabellendokumente und verschiedene andere Dateiformate als Datenquelle, wodurch auch eine Analyse von Daten die weder über eine bereitgestellte API, noch durch den Direktzugriff auf eine Datenbank in das System gelangen können, möglich wird. Diese können in der Praxis gleichberechtigt neben den vorgestellten Möglichkeiten verwendet werden.

Als weitere Möglichkeit erlaubt BIRT auch die direkte Ansprache einer SOAP-API, vorausgesetzt, es gibt zu dieser die benötigten Beschreibungsgdaten. Dies wäre auch für das OTRS denkbar gewesen, aufgrund der nicht vorhandenen Beschreibungsdaten und der im Allgemeinen geringen Dokumentation der API der verwendeten Version wurde allerdings von dieser Möglichkeit abgesehen.

KAPITEL 6

ANALYSE DER WARTUNGSAUFWÄNDE

Im Folgenden soll zur Analyse der Wartungsaufwände eine allgemeine Übersicht über den Wartungsprozess gegeben werden und danach anhand der vorher definierten Metriken die Prozesse analysiert werden. Die Analysen sind Top-down, also beginnend im 1st-Level-Support und endend in der Entwicklung.

Im Allgemeinen ist zwischen zwei Ticket-Arten zu unterscheiden:

1. **Tickets ohne eine Verknüpfung zu einem Issue**: Diese Tickets erzeugen keine softwarerelevanten Wartungsaufwände, sind allerdings für die firmenrelevanten Wartungsaufwände relevant, da sie zum einen dafür sorgen, dass Kosten für die Firma entstehen und zum anderen die Bearbeitung von Tickets der ersten Klasse verzögern können. Daher ist zu analysieren, aus welchen Gründen es Tickets dieser Art gibt.
2. **Tickets mit einer Verknüpfung zu einem Issue**: Diese Tickets erzeugen softwarerelevante Wartungsaufwände, da sie einen Prozess in der Entwicklung führen.

Wie bereits vorher angemerkt, werden nur geschlossene Tickets betrachtet.

6.1. Gesamtübersicht

Die Gesamtübersicht kann nur getrennt für beide Ticketarten gegeben werden, da eine gemeinsame Statistik aufgrund der unterschiedlichen Abläufe keine schlüssige Darstellung erlaubt.

Dementsprechend folgt nun eine Darlegung des Prozesses anhand der beiden Ticketarten.

Zur Analyse dieser Tickets werden die Metriken Percentage of Tickets without Issues (per month) und Messages per Ticket implizit verwendet.

6.1.1. Tickets ohne eine Verknüpfung zu einem Issue

Als erstes betrachten wir die Tickets, die keine Verbindung zu einem Issue haben. Diese Analyse ist unterteilt in den 1st- und den 2nd-Level - Support, da im 1st-Level-Support aufgrund des strukturellen Aufbaus der Wartung grundsätzlich keine Verknüpfung zu der Entwicklung herrschen sollte und diese Tickets daher gesondert betrachtet werden sollen.

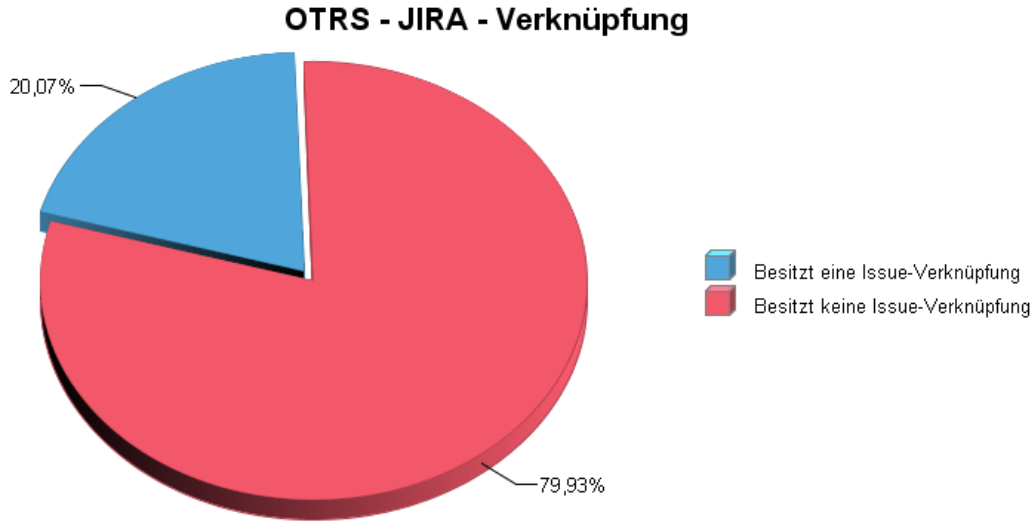

Abbildung 6.1.: Anteil der Tickets mit einer Verknüpfung im 1st - Level - Support

Beginnend mit dem 1st-Level-Support ergibt sich, dass 80% der Tickets in diesem keine Verknüpfung zu einem Issue besitzen. Allerdings hat sich dieser Anteil auf bis zu 95% erhöht (siehe auch Anteilsverlaufsgrafik im Anhang A.7), das heißt es widersprechen nur noch ein geringer Teil der Tickets der firmentinternen Prozesse. Es ist daher davon auszugehen, dass es sich, auch durch die gewachsenen Strukturen der Firma, die keine vollständige Personaltrennung zwischen den beiden Supportschichten umschließt, inzwischen nur noch um nicht vorgenommene Einsortierungen handelt, die zwar optimiert werden sollten, um die statistische Analyse zu verbessern, aber im weiteren Verlauf keinen Einfluss auf die Wartungsqualität und auf die entsprechenende Wartungsaufwände haben. Diese Tickets können allerdings nicht in weitere Analysen einfließen, da sie nicht produktspezifisch sind, also nicht unterschieden werden kann, zu welchem Produkt aus der Firma die entsprechenden Anfragen gehören.

Eine weiterführende Analyse dieser Tickets zeigt, dass diese durchschnittlich 415 Stunden im Support verweilen, was einer ungefähren Verweildauer von 17 Tagen und 7 Stunden entspricht. In dieser Zeit werden im Durchschnitt 1,965 Nachrichten durch einen Supportmitarbeiter und 2,466 Nachrichten durch den Kunden versandt.

Als nächstes sollen die Tickets des 2nd-Level-Supports auf die gleichen Merkmale überprüft werden.

In der produktspezifischen Support-Queue des 2nd-Levels besitzen 44,46% der Tickets keine Verknüpfung mit einem entsprechenden Issue im Issue-Tracker. Dies ist nicht in

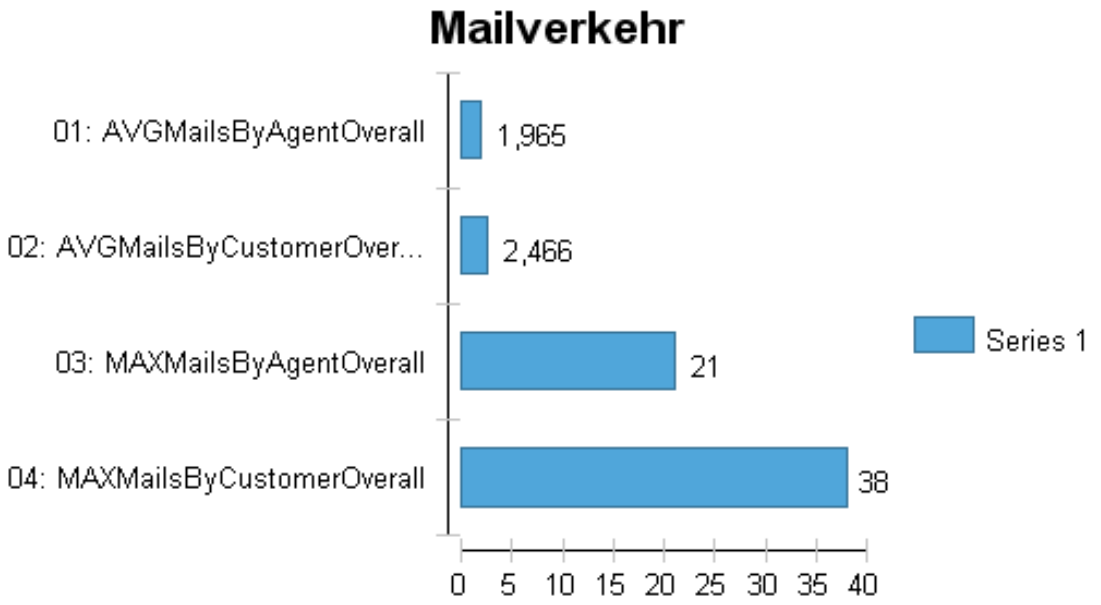

Abbildung 6.2.: Nachrichtenversand im 1st-Level (keine Verknüpfung)

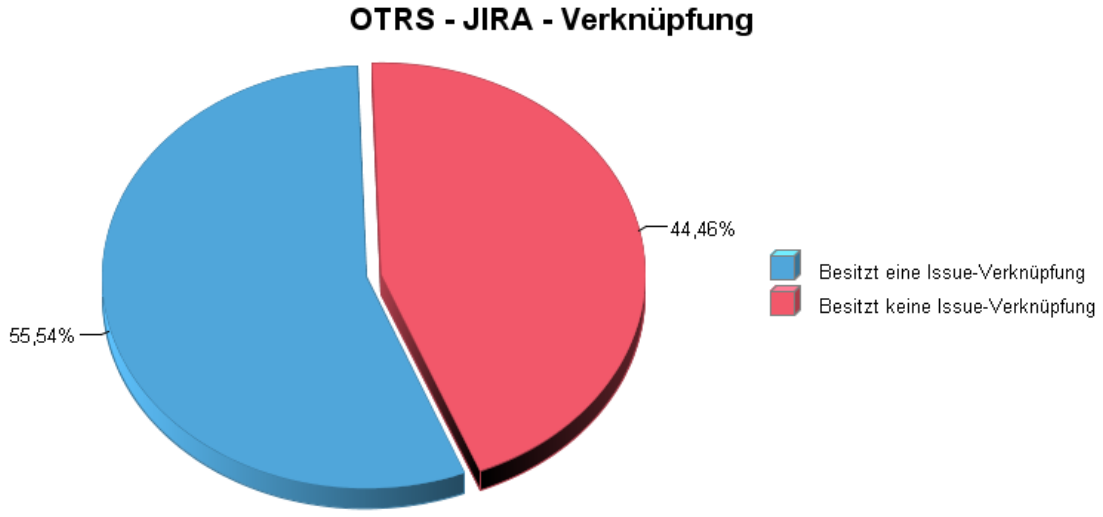

Abbildung 6.3.: Anteil der Tickets mit einer Verknüpfung im 2nd - Level - Support

erster Linie problematisch, da es auch Anfragen durch Kunden geben kann, die nicht durch das Wissen des 1st-Level-Supports gelöst werden können. Diese Tickets sind durchschnittlich 956 Stunden im Support, dies entspricht einer ungefähren Verweildauer von 39 Tagen und 20 Stunden entspricht. Während dieser Zeit werden durch den Support 3,552 Mails und durch den Kunden 4,102 Mails im Durchschnitt versandt.

Zu betrachten ist, warum diese Tickets keine Verknüpfung zu einem entsprechenden Issue besitzen, also ob ein etwaiges Problem bereits im 1st-Level-Support hätte gelöst werden können. Dazu soll eine Stichprobe genommen werden und auf die jeweiligen Lösungen hin analysiert werden. Eine entsprechende Volltext-Analyse wird unter zukünftige Erweiterungen näher erläutert werden.

Stichprobenbeschreibung

Zeitraum: 2011

Gesamtanzahl: 406 Tickets

Stichprobengröße / Abdeckung: 20 Tickets / 5%

Diese Tickets wurden mithilfe der zufälligen Sortierung, die von der MySQL-Datenbank bereitgestellt wird (ORDER BY RAND()), ermittelt.

Folgende Informationen lassen sich aus diesen Tickets ermitteln:

- 16 Tickets entsprechen den Einsortierungsrichtlinien, das heißt es sind Probleme, die nur mit entsprechenden Fachwissen oder Entwicklerrückfragen (die wiederum kein Issue mit sich bringen) lösbar sind. Innerhalb dieser Gruppe gibt es allerdings 2 Tickets, die auf Konfigurationsfehler zurückzuführen sind. Dies kann bedeuten, dass eine etwaige Optimierung der Dokumentation dem Support die Last abnehmen könnte. Ein weiterer Fall wurde in einer neueren Version gelöst, es könnte dementsprechend eine Verknüpfung zu dem entsprechenden Issue vorgenommen werden, allerdings ist diese Nichtverknüpfung kein gravierendes Problem.
- 2 Tickets sind als erfolgreich geschlossen deklariert worden, nachdem lange keine Antwort durch den Kunden erhalten wurde. Eventuell ist über einen weiteren Status für Tickets, die aufgrund einer lang ausgebliebenen Antwort nach Problemlösung geschlossen wurden, nachzudenken.
- 1 Ticket wurde aufgrund eines Fehlers mit Hardware-Relation erstellt (mit Konfiguration lösbar), eventuell durch 1st-Level-Support lösbar.
- 1 Ticket stellt eine Frage zu der Einstellung, die die Lösung im vorher genannten Ticket war. Es konnte kein Zusammenhang zwischen den Tickets. Dadurch sollte über eine Verbesserung der Dokumentation überlegt werden, damit eine solche Fragestellung in der Zukunft verhindert werden kann.

Die Stichprobenuntersuchung zeigt, dass 3 / 4 der Tickets richtig behandelt wurden, also nicht durch den 1st - Level - Support gelöst werden konnten, aber keinen Defekt als solchen in der Software beschreiben.

6.1.2. Tickets mit einer Verknüpfung zu einem Issue

Das folgende Diagramm gibt die durchschnittlichen Stunden an, die ein Ticket in den einzelnen Stadien des Prozesses verbringt.

Wie vorher angemerkt sind hierbei Tickets aus dem 1st-Level-Support nicht mit einbezogen, da diese keinem spezifischen Produkt zugeordnet werden können.

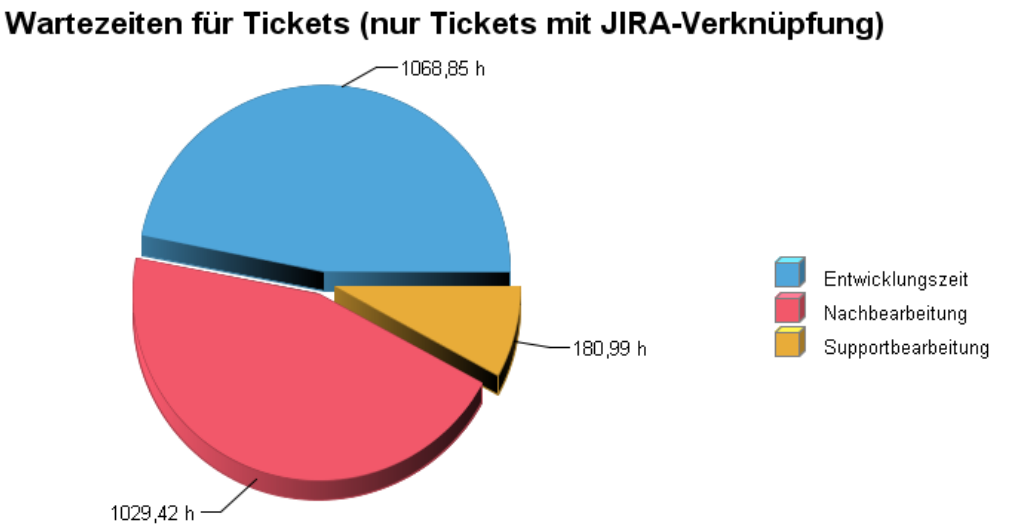

Abbildung 6.4.: Wartezeiten für Tickets im 2nd - Level - Support mit einer Verknüpfung zu einem Issue

Als erstes zeigt die Gesamtübersicht, dass ein Ticket mit Verknüpfung in die Entwicklung durchschnittlich 7 1/2 Tage im Support bearbeitet werden, bevor daraufhin 44 1/2 Tage auf die Lösung des Problems durch die Entwicklung gewartet wird und schlussendlich noch weitere 43 Tage in der Nachbearbeitung sind.

Allerdings ist anzumerken, dass die Nachbearbeitungszeit primär von der Zeit abhängt, die auf Kundenseite vorhanden ist, um die Änderungen einzupflegen, des Weiteren kommt es vor, dass Kunden nach erfolgreicher Änderung nicht in dem entsprechenden Ticket antworten, dass der Fehler behoben wurde, wodurch teilweise Tickets nicht direkt geschlossen werden können. Näheres zu diesem Problem ist in dem Kapitel über die Datenqualität zu finden.

Während dieser Zeiten wurden im Durchschnitt die in 6.5 angegebenen Anzahlen an Nachrichten versandt, dabei sind die einzelnen Phasen, also die Erstbearbeitung durch den Support, Entwicklung, Nachbearbeitung durch Support einzeln aufgezeigt.

Wie hier erkennbar ist, werden innerhalb der Erstbearbeitung, welche den geringsten zeitlichen Anteil hat, die meisten Nachrichten versandt, während in der erheblich längeren Nachbearbeitungszeit im Durchschnitt eine Nachricht weniger versendet wird, was wiederum auf die Tatsache hinweist, dass diese Zeit nur aufgrund von Nicht-Beantwortung durch den Kunden in dieser Länge vorhanden ist.

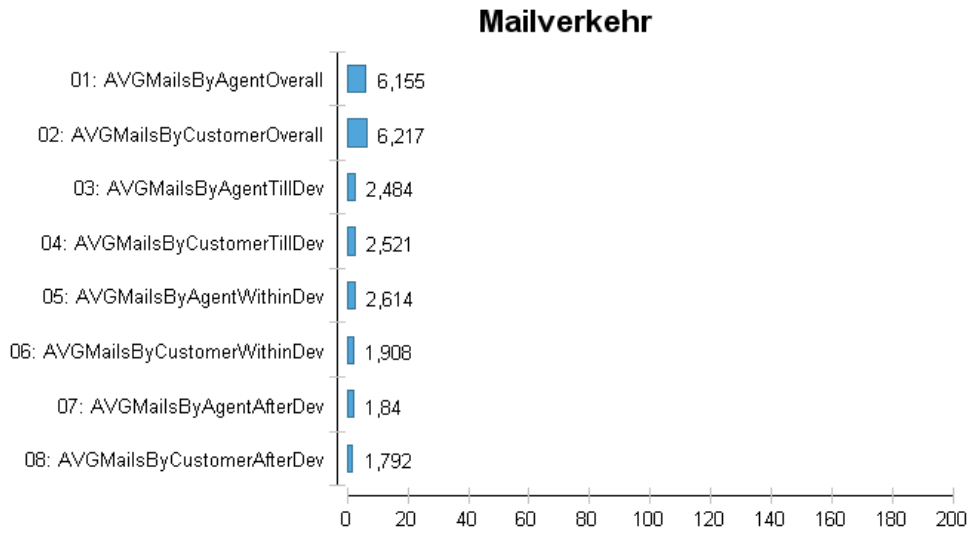

Abbildung 6.5.: Nachrichtenversand im 2nd-Level

6.2. Analysen der Wartungsaufwände mithilfe der Metriken

Als nächstes sollen die Wartungsdaten mithilfe der Metriken analysiert werden.

Dabei sind die Metriken, die im vorherigen Abschnitt bereits angewendet wurden nicht noch ein weiteres Mal explizit aufgeführt.

Vorab sind folgende Dinge festzustellen:

- Es wurde keine Version 201108 veröffentlicht, sondern diese mit der Version 201109 zusammen veröffentlicht. Dabei ist ein erhöhter Anteil an neuen Features zu sehen, welcher bei näherer Analyse auf eine einzelne Erweiterung zurückzuführen ist. Dabei ist anzumerken, dass diese Version dennoch über dem statistischen Durchschnitt liegt, selbst wenn man die Zeiten für diese Version auf zwei Versionen aufteilen würde. Auf Nachfrage ergab sich, dass diese Versionszusammenführung aufgrund eines veränderten Releasezyklus' durchgeführt wurde, bei dem der Zeitpunkt der monatlichen Veröffentlichung geändert wurde.
- alle Versionen nach 201204 sind in der Statistik nicht als endgültig zu betrachten, da der Stand der Daten dem von Anfang Juni 2012 entspricht. Des Weiteren werden, wie vorher angemerkt, Wartungszeiten der ersten betroffenen Version angerechnet, was in diesem Fall auch das Ergebnis von früheren Versionen noch nach Veröffentlichung dieser Statistik verändern kann. Auch bei vorigen Versionen können sich, aufgrund der aktiven Wartung, noch Veränderungen ergeben.
- die Versionen vor 200903 weisen eine hohe Unstetigkeit auf, was durch die Einführung von JIRA zu diesem Zeitpunkt zurückzuführen ist, dementsprechend sind auch diese nicht weiter zu betrachten, da sie etwaige Schlüsse verfälschen würden. Auch der folgende Jahreszeitraum ist von vergleichsweise geringen Zeiten geprägt, was auf die kontinuierliche Einführung des Systems hinweist.

- nicht die gesamte Zeit, die für eine Version aufgewendet wird, muss in dem der Version vorangehenden Monat geleistet werden. So sind gerade Zeiten für Bugs nach dem Erscheinen einer Version und neue Features vor dem Erscheinen einer Version anzusiedeln.

6.2.1. Time per Version

Als erste Metrik soll Time per Version betrachtet werden, die Statistik ist in der Abbildung 7.1 zu finden.

Die Analyse der Zeit pro Version ergibt eine durchschnittliche Zeit von ca. 586 Stunden , wobei im Durchschnitt 190 Stunden auf Bugfixing und weitere 51 Stunden für Incomplete Specification Implementation aufgewendet werden. Laut Entwicklungsabteilung sind 40% der Arbeitszeit eines Mitarbeiters auf Bugfixing eingeplant, allerdings wird dies nicht strikt kontrolliert. Dennoch ist eine gewisse Korrelation von den realen und geplanten Zeiten vorzufinden.

Die Zeit, die für die Issue-Art Wish aufgewendet wird, ist sehr niedrig und taucht nur in zwei Version auf, nämlich 201009 und 201105. Da sich bei beiden die Zeit in einem Bereich von 5 Stunden befindet, ist anzunehmen, dass viele Kundenwünsche nicht als solche klassifiziert werden, sondern als neues Feature, wobei auch diese Einsortierung keineswegs falsch ist, allerdings ist abzuwägen, ob eine Unterteilung in die beiden Arten sinnvoll ist oder ob es im Allgemeinen reicht, beide Wartungsarten in einer zusammenzufassen, da beide nach den klassischen Wartungskategorien in dieselbe fallen.

Im Allgemeinen haben die Zeiten eine hohe Varianz, so wurden beispielsweise für die Versionen 201011, 201103, 201106, 201111 und 201201 jeweils ca. 350 Stunden aufgewendet, für die Version 201203 sogar weniger als 200 Stunden, während auf der anderen Seite für Versionen wie 201010, 201105, 201107 und 201204 mehr als 800 Stunden aufgewendet wurden.

6.2.2. Time per Issue per Version

Die Analyse über diese Metrik soll zur genaueren Darlegung in die einzelnen Wartungstypen aufgeteilt betrachtet werden; dabei wurden die innerhalb des Unternehmens verwendeten Issue-Klassifikationen anhand der in 2.4 definierten Zuordnung zusammengefasst.

Die jeweiligen Diagramme sind der Übersichtlichkeit halber im Anhang zu finden.

Des Weiteren sind alle Statistiken sowohl für den Median als auch für den Durchschnitt aufgestellt.

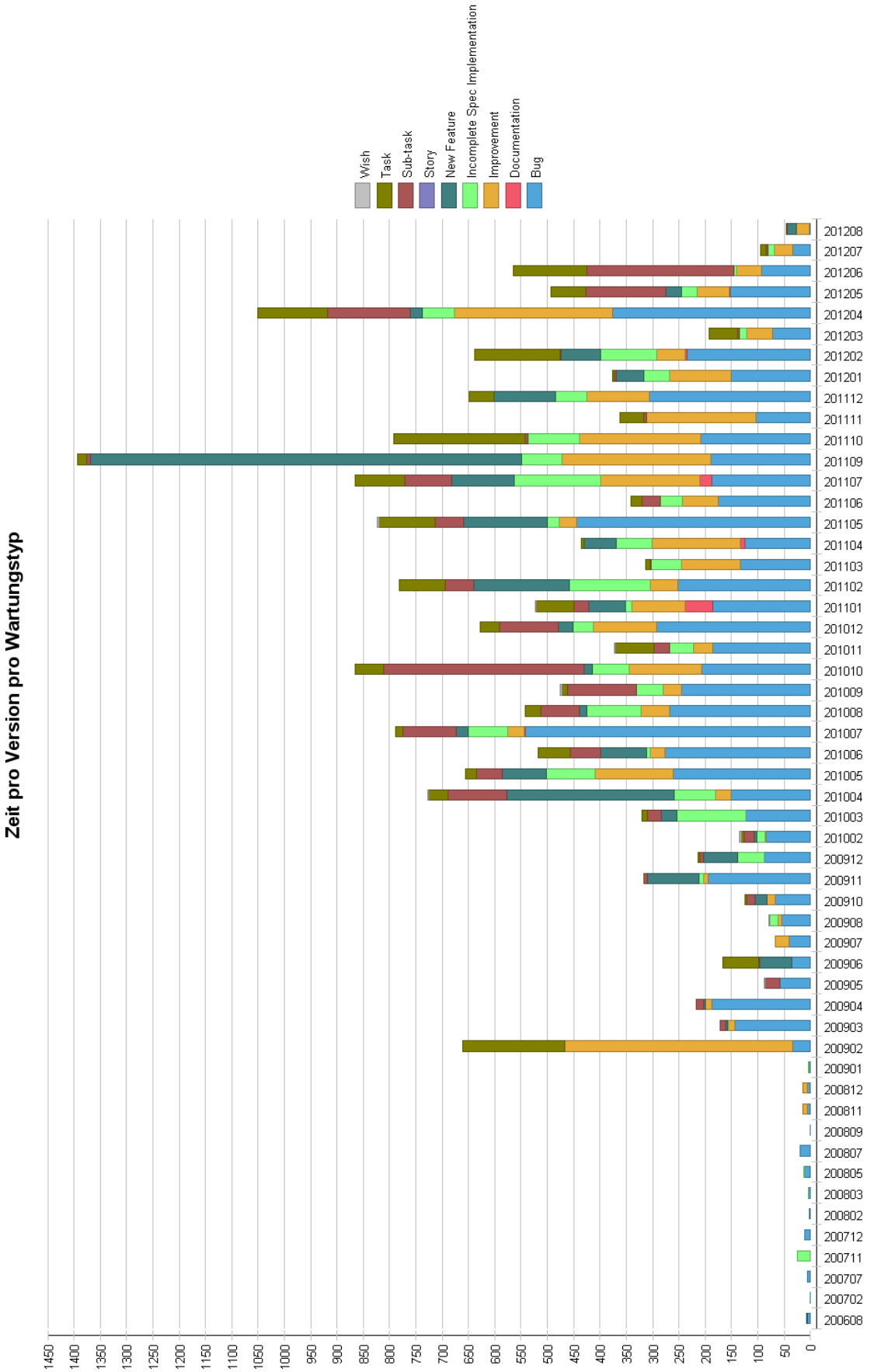

Abbildung 6.6.: Zeit pro Version aufgeteilt in einzelne Wartungstypen

Adaptive Wartung

Bei der adaptiven Wartung ergibt sich eine durchschnittliche Zeit von 10,472 Stunden, um in Issue zu bearbeiten.

Der Median (berechnet durch den Median der Mediane) für die Zeit, um ein Issue zu bearbeiten beträgt 4 Stunden.

Korrektive Wartung

Bei der korrektiven Wartung ergibt sich eine durchschnittliche Zeit von 5,6146 Stunden, um in Issue zu bearbeiten.

Der Median (berechnet durch den Median der Mediane) für die Zeit, um ein Issue zu bearbeiten beträgt 2 Stunden.

Perfektionierende Wartung

Bei der perfektionierenden Wartung ergibt sich eine durchschnittliche Zeit von 13,453 Stunden, um in Issue zu bearbeiten.

Der Median (berechnet durch den Median der Mediane) für die Zeit, um ein Issue zu bearbeiten beträgt 7.792 Stunden.

Zusammenfassung

Laut Statistik sind Issues, die einen Defekt behandeln die Issues, die am schnellste geschlossen werden können, wobei der Median 2 Stunden beträgt, während für die perfektionierende Wartung im Median 7,792 Stunden aufgewendet werden müssen, also nahezu einen gesamten Entwicklertag. Die durchschnittlichen Zahlen sind jeweils höher, was sich durch einzelne, sehr zeitintensive Issues erklären lässt.

Da die perfektionierende Wartung im Verhältnis zu anderen Wartungsarten teils erheblich länger dauert ist im Einzelfall abzuwägen, ob eine Implementierung der vollen Funktionalität von vorneherein sinnvoller ist, da auch grundsätzlich aus der Theorie und Praxis bekannt ist, dass das nachträgliche Einfügen oder die Optimierung von Funktionalitäten zeitintensiver ist als eine Entwicklung, die dies von Anfang an mit einschließt.

6.2.3. Issues per Version

In der Abbildung 6.7 ist der zeitliche Verlauf der Ticketanzahlen pro Wartungstyp ersichtlich.

Auch für diese Statistik gelten die bei der Metrik Time per Version genannten Einschränkungen, was die zu betrachtenden Version betrifft. Allerdings fällt auf, dass die Version 201109 nicht wie bei der zeitlichen Ansicht aus der statistischen Varianz herausfällt, sondern

Wie in der Abbildung erkennbar ist, beträgt die durchschnittliche Gesamtanzahl der Tickets beträgt 63. Dabei sind pro Version eine durchschnittliche Anzahl von 36 Bug-Issues bearbeitet worden, während ca. 19 Issues zu neuen Funktionen (einschließlich Task und Sub-Task) bearbeitet werden. Auf die Wartungsart Incomplete Specification Implementation entfallen weitere 4 Issues. 9 Issues beschäftigen sich mit der Verbesserung der Funktionen der Software.

Auch bei der Anzahl an Issues, die für eine Version anfallen gibt es eine gewisse Varianz, die bei der Analyse der Zeit pro Version angemerkten Version bilden auch hier die Varianz, allerdings gibt es ein paar weitere Versionen, die statistisch markant sind, beispielsweise die Version 201101. Die Version 201203, die mit einer geringen Stundenanzahl aufgefallen war, ist statistisch nicht auffällig, was die Anzahl an Issues angeht.

6.3. Zusammenfassung der Ergebnisse

Zusammenfassend lässt sich aussagen, dass es einen großen Anteil an Arbeit zulasten der Behebung von Defekten gibt, wobei groß in diesem Zusammenhang ohne Wertung zu betrachten ist. Es zeigt sich, dass mehr als 40 % der Zeit für diese Arbeiten aufgebracht werden müssen, wobei der Anteil an Issues höher ist als die Zeit, die aufgewendet wird.

Des Weiteren ist sichtbar geworden, dass es eine große Anzahl an Tickets gibt, die sich lange in der Nachbearbeitung befinden, obwohl in dieser Zeit nur wenige Nachrichte versandt werden. Hier kann überprüft werden, ob eine schnellere Schließung eins Tickets oder ein weiterer Status für unter Vorbehalt geschlossene Tickets der Übersichtlichkeit zuträglich wären, da dann Tickets, die keinen akuten Eingriff benötigen, nicht mehr in den entsprechenden Listen auftauchen würden.

Ein weiterer Punkt, der sichtbar wird, ist, dass es sowohl im Durchschnitt als auch im Median halb solange dauert, einen Fehler zu beheben, wie es für die Schaffung einer neuen Funktion oder der Optimierung einer alten benötigt. Am höchsten fällt allerdings die Dauer für die perfektionierende Wartung aus, diese ist im Median beinahe doppelt so hoch wie die für die adaptive Wartung.

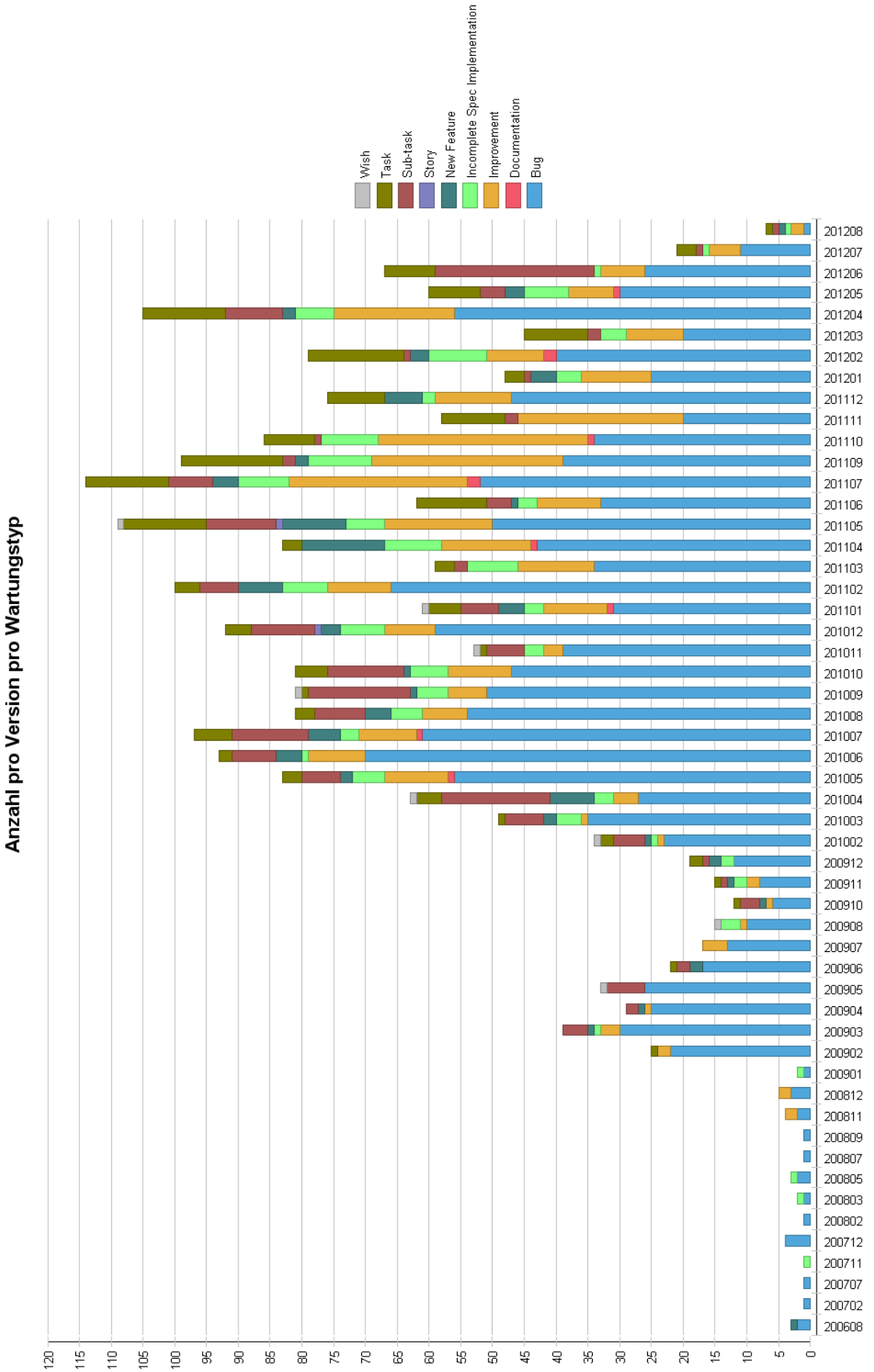

Abbildung 6.7.: Anzahl an Issues pro Version aufgeteilt in einzelne Wartungstypen

KAPITEL 7

ANALYSE UND VERBESSERUNGSVORSCHLÄGE DER DATENQUALITÄT

Um eine Analyse der Daten in ihrer Richtigkeit bewerten zu können, müssen Aussagen über die Qualität der analysierten Daten gegeben werden. Dabei gibt es verschiedene Faktoren, die die Qualität beeinflussen, darunter:

- **Genauigkeit der Daten**: Wie detailliert sind einzelne Zeiteinträge im Rahmen eines Issue-Trackers? Wie gut ist die Abdeckung der wöchentlichen Arbeitszeit?
- **Aufbereitung der Daten**: Wie leicht sind die Daten aufbereitbar / statistisch verwertbar? Welche Hindernisse gibt es?

Die Analyse der Datenqualität soll für jedes System individuell betrachtet werden, da für beide Systeme unterschiedliche Maßstäbe für Qualität anzulegen sind. Während beispielsweise die Genauigkeit der Daten für einen Issue-Tracker eine Rolle spielt, ist die genaue Zeit, die an einem Problem innerhalb des Supports gearbeitet wird, nicht sonderlich relevant.

7.1. Datenqualität im OTRS

Bei den aus dem OTRS bzw. im Allgemeinen aus einem Ticketsystem zu erlangenden Daten ist eine Erfassung der zeitlichen Dauer eines einzelnen Vorgangs nicht relevant, das heißt es ist für die Messung der Aufwände, die im Support entstehen, nebensächlich, wie lange an einer Antwort gearbeitet wurde. In diesem Fall sind Parameter wie die Anzahl der Wartungsfälle oder die Anzahl an versandten Nachrichten für eine Analyse relevant. Diese Faktoren können aber immer exakt aus einem solchen System entnommen werden und sind nicht manuell manipulierbar, wie es in einem Issue-Tracking-System, aufgrund der Eingabe der Daten durch einen Benutzer, möglich ist.

Diese Angaben wurden bereits mithilfe der Metriken analysiert und sind daher nicht Teil der Diskussion über die Datenqualität.

Ein anderer Punkt, der bereits bei der Analyse der Wartungsaufwände aufgetaucht ist, ist die Tatsache, dass eine Einordnung der Tickets in die richtige Queue nicht immer vorgenommen wird, da es beispielsweise Mitarbeiter gibt, die sowohl im 1st- als auch 2nd-Level-Support tätig sind. Dieser Punkt besitzt zwar aufgrund der bei den analysierten Daten inzwischen geringen Häufigkeit eine geringe Relevanz, darf aber nicht unterschlagen werden, wenn über die Qualität der Daten diskutiert wird.

Problematisch sind solche Fehler in der Einsortierung insbesondere dadurch, dass diese Tickets für Analysen der Wartungsaufwände nicht zur Verfügung stehen, außer es wird in Kleinstarbeit ein nachträgliche Sortierung vorgenommen und somit der Mangel behoben. Dies ist allerdings eine mühsame Arbeit, da für diese Zwecke, ähnlich der im vorherigen Kapitel angewandten Stichprobenuntersuchung, eine Analyse jedes Tickets auf dessen Inhalt durchgeführt werden muss. Es ist daher im Allgemeinen sinnvoller, die älteren Daten als solche zu akzeptieren und dafür zu sorgen, dass zukünftig Einsortierungsrichtlinien beachtet werden und dies mithilfe der vorgestellten Metrik zu analysieren und wenn notwendig zu verschieben.

Des Weiteren sollte angedacht werden die statistische Verwertung zu erleichtern, indem ein kurzer Abschlussfragebogen zu jedem Ticket abgegeben werden kann, in dem kurz erläutert wird, was der Fehler war, wie das eigene Gefühl zur Länge des Wartungsfalls ist und wie die Kommunikation mit dem Kunden sowie mit der Entwicklung, so sie denn involviert ist, zu bewerten ist. Dies wäre leicht in einem Formular mit drei Auswahlfeldern (mit der zusätzlichen Freitextmöglichkeit) lösbar und wäre somit auch in den Arbeitsfluss leicht zu integrieren, würde aber die Analyse der Wartungsaufwände vereinfachen, da detailliertere Informationen vorliegen würden.

7.2. Datenqualität in JIRA

JIRA hat, wie jedes System, das durch den Entwickler selbst gepflegt wird, den Nachteil, dass Arbeitsstunden direkt durch den Entwickler in das System eingetragen werden. Dies führt dazu, dass bei der Datenqualität im Allgemeinen eine gewisse Skepsis angebracht ist. Es ist in der Realität eine schwierige Gratwanderung wie notwendig eine exakte Protokollierung der Arbeitsleistung durch einen Mitarbeiter ist und inwieweit dies seine eigentliche Arbeit, der Erstellung der Software, behindert. Für die Analyse der Datenqualität innerhalb von JIRA soll betrachtet werden, wie viele Arbeitsstunden täglich durch Eintragungen im System gedeckt sind. Dafür betrachten wir den Zeitraum 2011.

Dabei sind folgende Punkte zu beachten:

- eine durchschnittliche Arbeitswoche umfasst 40 Stunden
- es findet keine explizite Arbeitszeitkontrolle statt
- die Zeit, die für Planungen, interne Meetings, u.ä. aufgewendet wird, wird nicht im System erfasst
- die monatlich verfügbare Arbeitskraft schwankt aus verschiedenen Gründen, bspw. zusätzliche Praktikanten, Neueinstellungen, Urlaub oder Entlassungen, dementsprechend soll zum Vergleich für die folgenden Diagramme noch jeweils das

Vollzeitäquivalent angegeben werden, also wie viele Arbeitskräfte theoretisch in den jeeiligen Monaten zur Verfügung standen.

- es werden alle Projekte erfasst, an denen in der C-Abteilung des Unternehmens gearbeitet wird, da ansonsten nicht ersichtlich ist, wie gut die monatliche Abdeckung wirklich ist.

Arbeitszeitäquivalenzen

Januar	Februar	März	April	Mai	Juni
16,7	16,7	15,7	15,7	15,7	15,7

Juli	August	September	Oktober	November	Dezember
15,7	16,2	14,9	15,0	15,7	16,0

Wie in den Abbildungen 7.1 und 7.2 erkennbar ist, sind 2011 insgesamt 12.668 Stunden eingetragen worden.

Anhand der Arbeitszeitäquivalenzen berechnen wir folgende maximale Stundenanzahl:

Summe der Arbeitszeitäquivalenzen: 189,7

Durchschnitt: 15,81

Gesamtarbeitstage: 251 (9 gesetzliche Feiertage)

Ergibt folgende Berechnung:

15,81*8*251 = 31743,13 Stunden, was einer Abdeckung von 39,9 % ergibt

Dies ist allerdings wie bereits angemerkt die maximal mögliche Anzahl an Stunden, die in das System eingepflegt werden könnte. Eine realistische (durch Befragung aufgenommene) Anzahl an Stunden, die für Meetings aufgewendet werden, ist 5 pro Woche, also eine Stunde pro Tag. Darüber hinaus sind verschiedene Pausen der Bildschirmarbeit gängig und auch sinnvoll, welche mit 5 Minuten pro Stunde veranschlagt werden. Mit einer nicht näher geregelten Mittagspause, die aufgrund der nicht explizit geregelten Arbeitszeit auch von dieser entnommen werden kann ist von einer maximalen Stundenanzahl von 6 Stunden pro Tag auszugehen (bei einer ca. 30 Minuten Mittagspause). Dies enstpricht der maximalen Zeit, die für das Schreiben von Quelltext veranschlagt wird, dabei sind Denkpausen, kurze Besprechungen mit Kollegen bezüglich einem Problem und verschiedene andere Faktoren nicht näher betrachtet.

Alle Zeiten, die angegeben wurden sind Mindestschätzungen und können in der Realität auch zu niedrig angesetzt sein. Mit diesen Zahlen ergibt sich allerdings eine maximale Stundenanzahl von 23809,86 mit einer Abdeckung von 53 % ergibt.

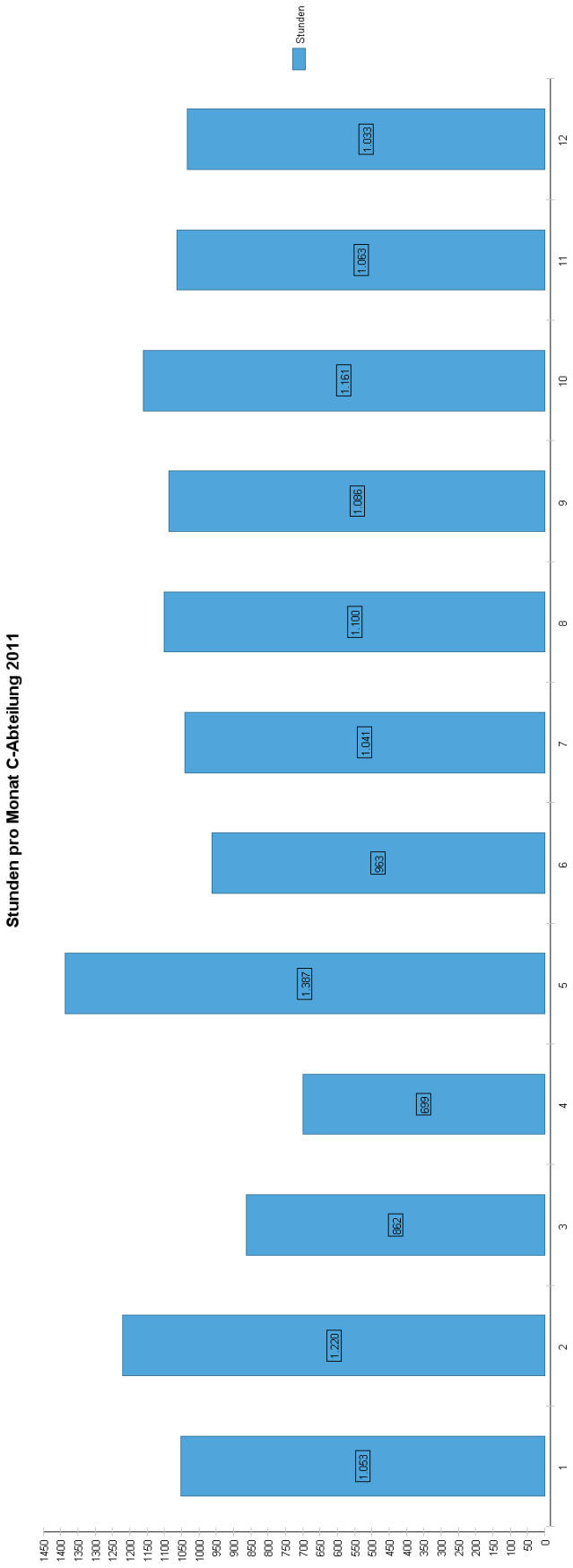

Abbildung 7.1.: Zeit pro Version aufgeteilt in einzelne Wartungstypen 2011

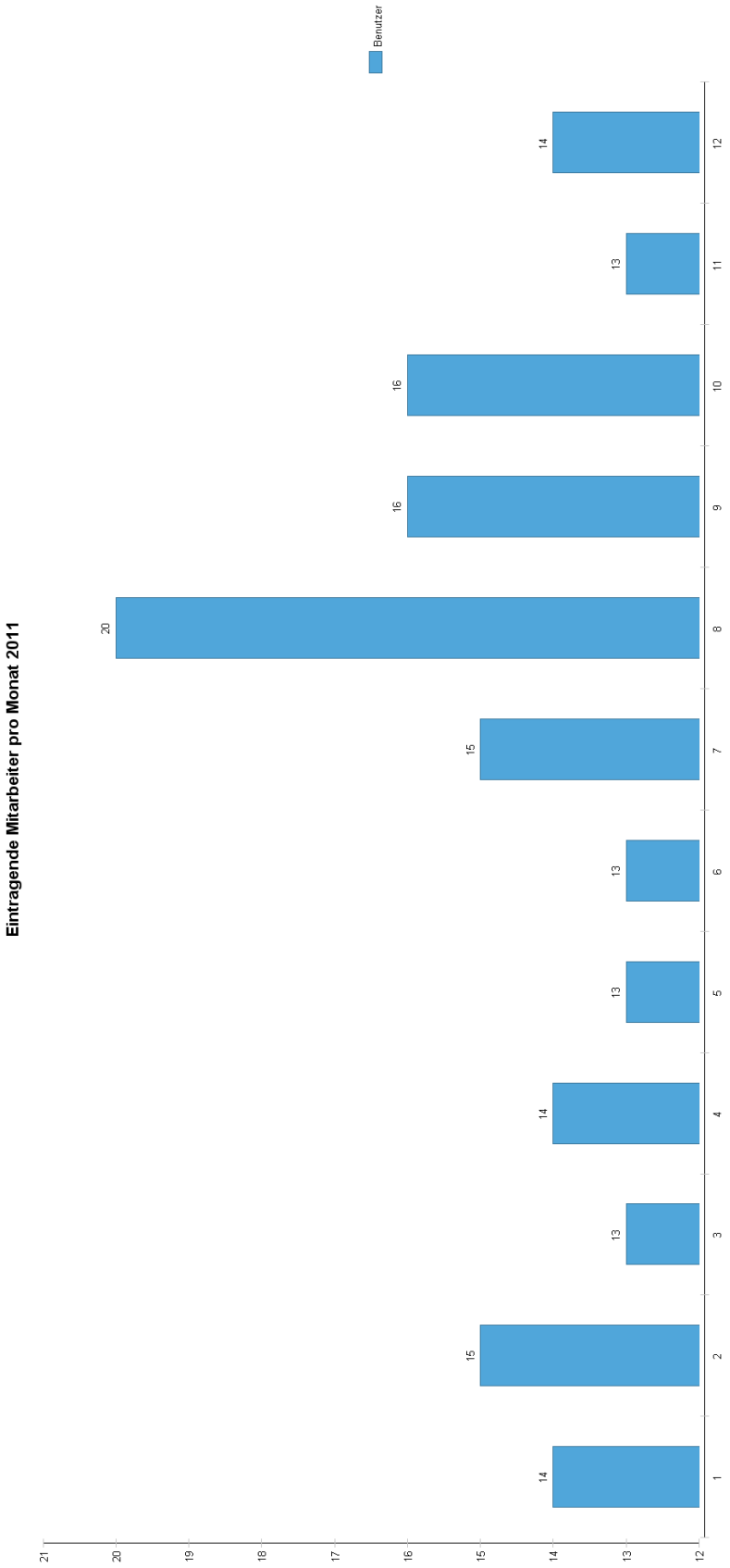

Abbildung 7.2.: Eintragende Benutzer pro Monat 2011

KAPITEL 8

ZUSAMMENFASSUNG UND AUSBLICK

In dieser Arbeit wurden verschiedene Metriken eingeführt, die zu einer besseren Erfassung und Bewertung von anfallenden Wartungsaufwänden verwendet werden können. Im weiteren Verlauf haben wir diese auf Realdaten anwenden können. Es liegt in der Natur der Sache, dass die Analyse von Wartungsdaten eines einzigen Produktes einer einzigen Firma nur ein Tatsachenbericht ohne Vergleichsmöglichkeiten ist. Dementsprechend knapp sind auch die Analysen in dieser Arbeit ausgefallen. So umfassen sie zwar die Details der einzelnen Statistiken, doch können sie kaum eine Bewertung darstellen, da kein endgültiger Schluss darüber gezogen wird, ob der Wartungsaufwand nun hoch oder niedrig ist. Auch gerade unter diesen Aspekten ist die Plausibilität der vorgestellten Metriken natürlich nicht nachweisbar, weshalb zu hoffen bleibt, dass sich weitere Firmen für bereit erklären, Wartungsdaten für Analysen bereitzustellen, um in diesem Forschungsfeld weitere Erkenntnisse zu ermöglichen.

Im Verlauf der Arbeit wurde auch gezeigt, welche Fallstricke sich in der Analyse der Wartungsvorgänge verbergen und welche Anforderungen an verwendete Systeme zur Erfassung von Wartungsvorgängen gestellt werden sollten, um eine gute Verwertbarkeit von gesammelten Daten zu gewährleisten. Auf der anderen Seite sind nicht zuletzt datenschutzrechtliche Aspekte im Auge zu behalten, da eine Vollerfassung sämtlicher Arbeitsschritte eines Mitarbeiters auch eine Überwachung der Tätigkeit bis hin zu einer vollen Tageserfassung bedeuten kann.

Ausblick und mögliche Erweiterungen

Wie bereits im Verlaufe der Arbeit angemerkt wurde, gibt es verschiedene Möglichkeiten, die hier beschriebenen Analysen weiter auszubauen, diese sollen im Folgenden kurz erläutert werden:

8.0.1. Volltext-Analysen

Die Analyse von einzelnen Tickets auf ihren Inhalt ist eine mühsame Tätigkeit, da häufig der gesamte Verlauf betrachtet werden muss, um Fehlerquelle und auch den Wartungsverlauf zu ermitteln. Abgesehen von der Möglichkeit der Klassifizierung der einzelnen Tickets durch den Support anhand von verschiedenen Parametern, wie es in unter der Analyse der Datenqualität angesprochen wurde, kann auch eine automatisierte Volltextanalyse zielführend werden. Diese hätte den Vorteil, dass der Support keine weitere Arbeit durchführen muss und somit keine Veränderung im Arbeitsfluss vonnöten sind, allerdings ist für die Analyse mehr Aufwand und neue Metriken vonnöten.

Vorrausetzungen für eine solche Analyse sind zum einen ein Wörterbuch für nicht aufzunehmende Wörter (Artikel, Konjunktionen, o.ä.), um bei der Analyse nicht von irrelevanten Wörtern abgelenkt zu werden.

Des Weiteren ist es vonnöten, dass eine genügend große Anzahl an Tickets in einer ersten Analyse statistisch in verschiedene Klassen einsortiert werden, die jeweils über charakteristische Wortkombinationen erkannt werden können.

8.0.2. Formalisierung des Wartungsaufwandes

Die in dieser Arbeit vorgestellten Metriken erlauben es, Wartungsaufwände in verschiedenen Zahlen zu messen. Allerdings sind all diese Metriken im Ergebnis Absolutzahlen, die auch in Relation zum Softwareumfang und der Entwicklungsteamgröße stehen.

Idealerweise sollte eine Formalisierung der Metriken in Hinblick auf diese Faktoren vorgenommen werden, allerdings ist es im Rahmen der Analyse von Wartungsaufwänden einer Firma nicht einfach abzuwägen, welche Faktoren in welchem Maße in eine Metrik einfließen sollen. Daher ist eine Formalisierung im aktuellen Stadium der Forschung nicht zielführend.

8.0.3. Analyse der Entwicklung neuer Funktionen zur Kundenakquise

Es kommt in der laufenden Softwareentwicklung des öfteren vor, dass für die Akquise eines neuen Kundens ein von diesem gewünschtes Feature in die Software eingebaut wird. Die Analyse dieser Entwicklungen kann eine Abschätzung liefern, inwiefern sich solche Funktionaitätserweiterungen finanziell rentieren.

Aufgrund der Tatsache, dass diese Erweiterungen bisher nicht speziell klassifiziert werden ist eine solche Analyse anhand der hier analysierten Daten nicht möglich, aber als zukünftige Erweiterung denkbar.

8.0.4. Einzelfallanalyse der perfektiven Wartung

Wie im Laufe dieser Arbeit festgestellt wurde macht in den vorliegenden Daten die perfektionierende Wartung einen konstanten Anteil an den Wartungsvorgängen aus. Es ist nun interessant, inwiefern sich der Gesamtwardungsaufwand, speziell im korrektiven Bereich, zu dem wir den perfektiven Wartungsaufwand mindestens teilweise hinzubeziehen müssen, verringern ließe, falls Funktionen grundsätzlich vollständig implementiert würden.

ANHANG **A**

ANHANG

Im Anhang befinden sich die Diagramme für die einzelnen Durchschnittszeiten und Mediane für die Metrik time per Issue. Des weiteren findet sich die Statistik über den Verlauf der Verknüpfungen von OTRS und JIRA.

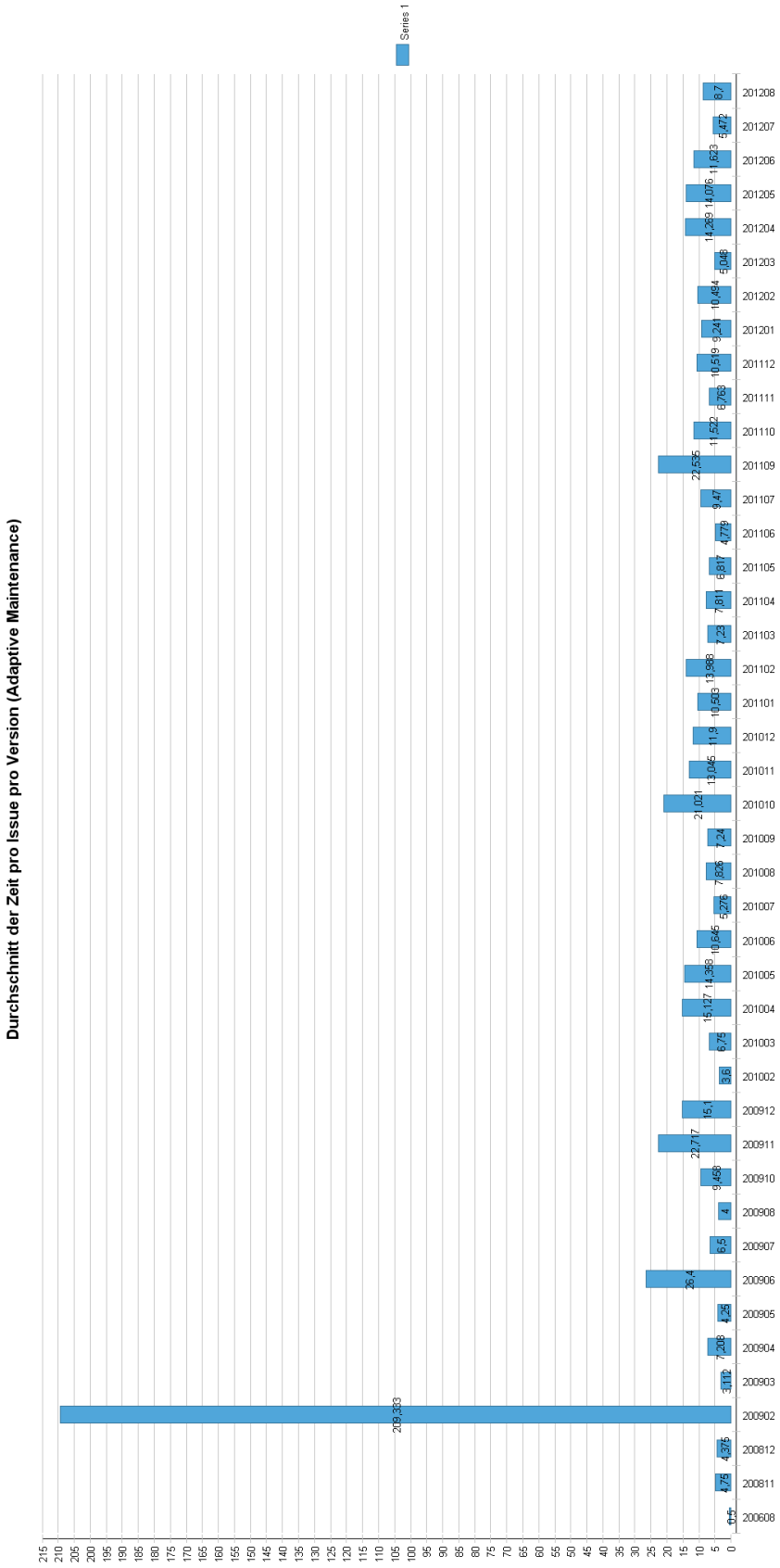

Abbildung A.1.: Durchschnittliche Zeit für ein Issue (adaptive Wartung)

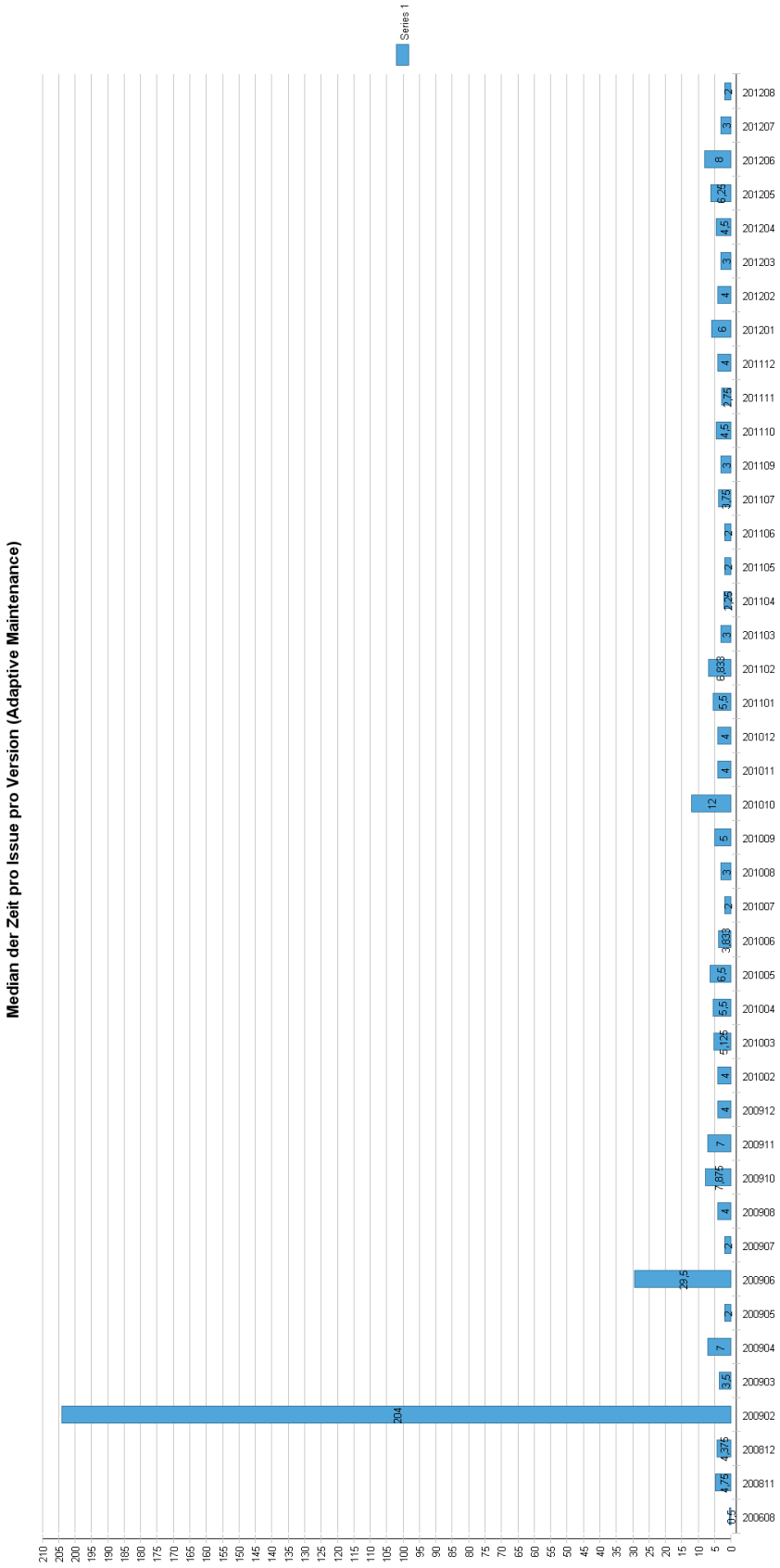

Abbildung A.2.: Median der Zeit für ein Issue (adaptive Wartung)

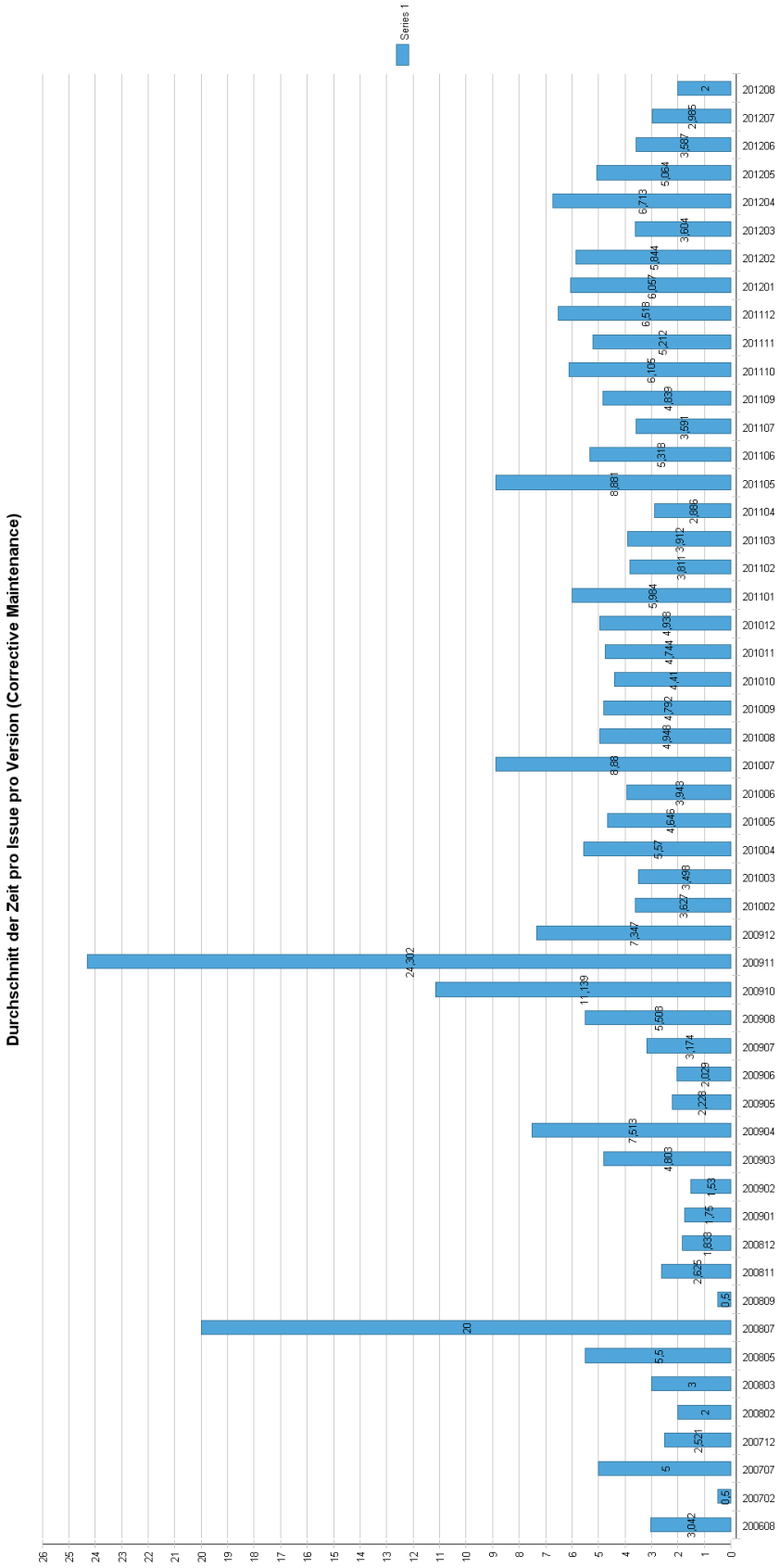

Abbildung A.3.: Durchschnittliche Zeit für ein Issue (korrektive Wartung)

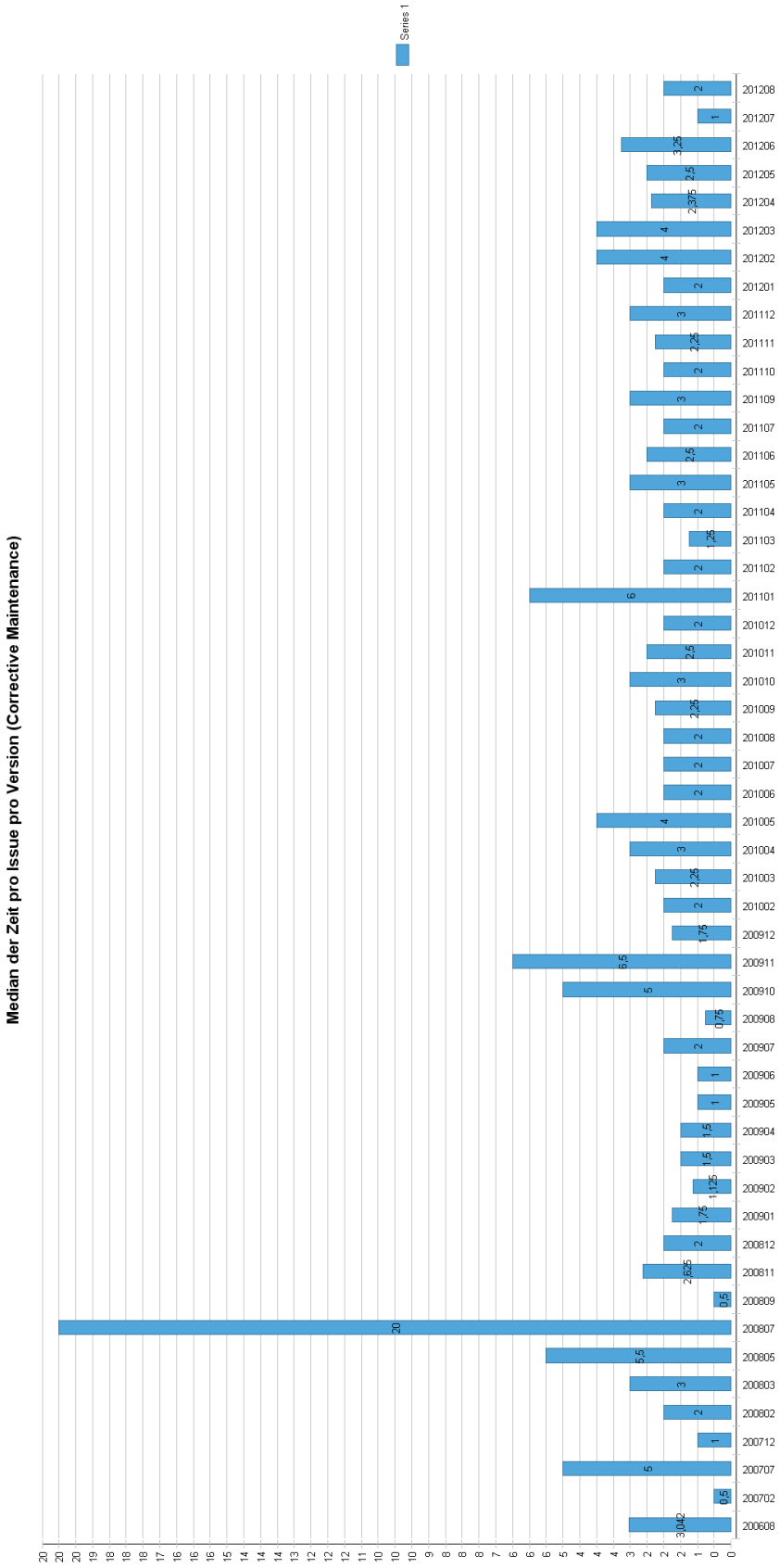

Abbildung A.4.: Median der Zeit für ein Issue (korrektive Wartung)

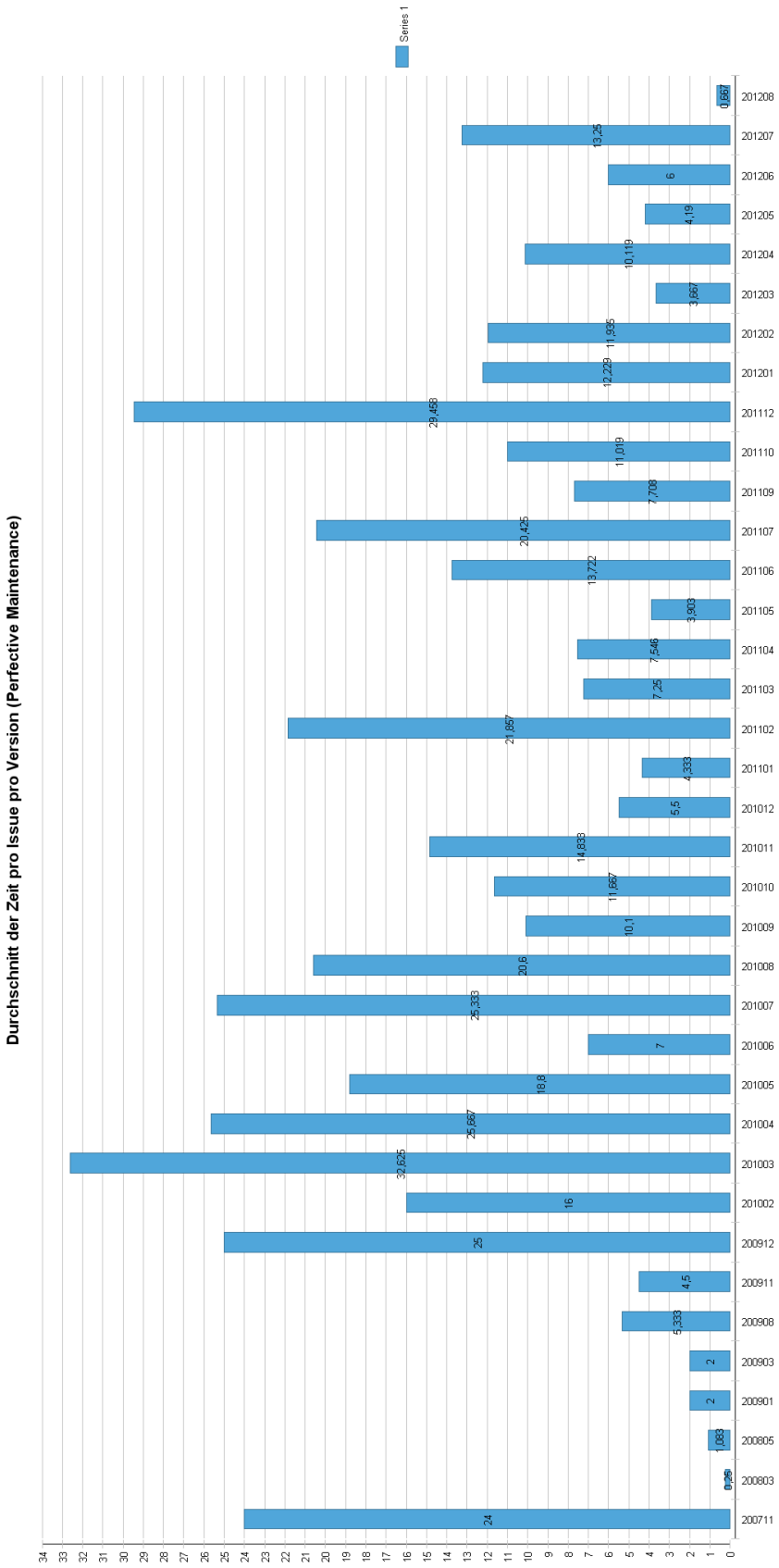

Abbildung A.5.: Durchschnittliche Zeit für ein Issue (perfektionierende Wartung)

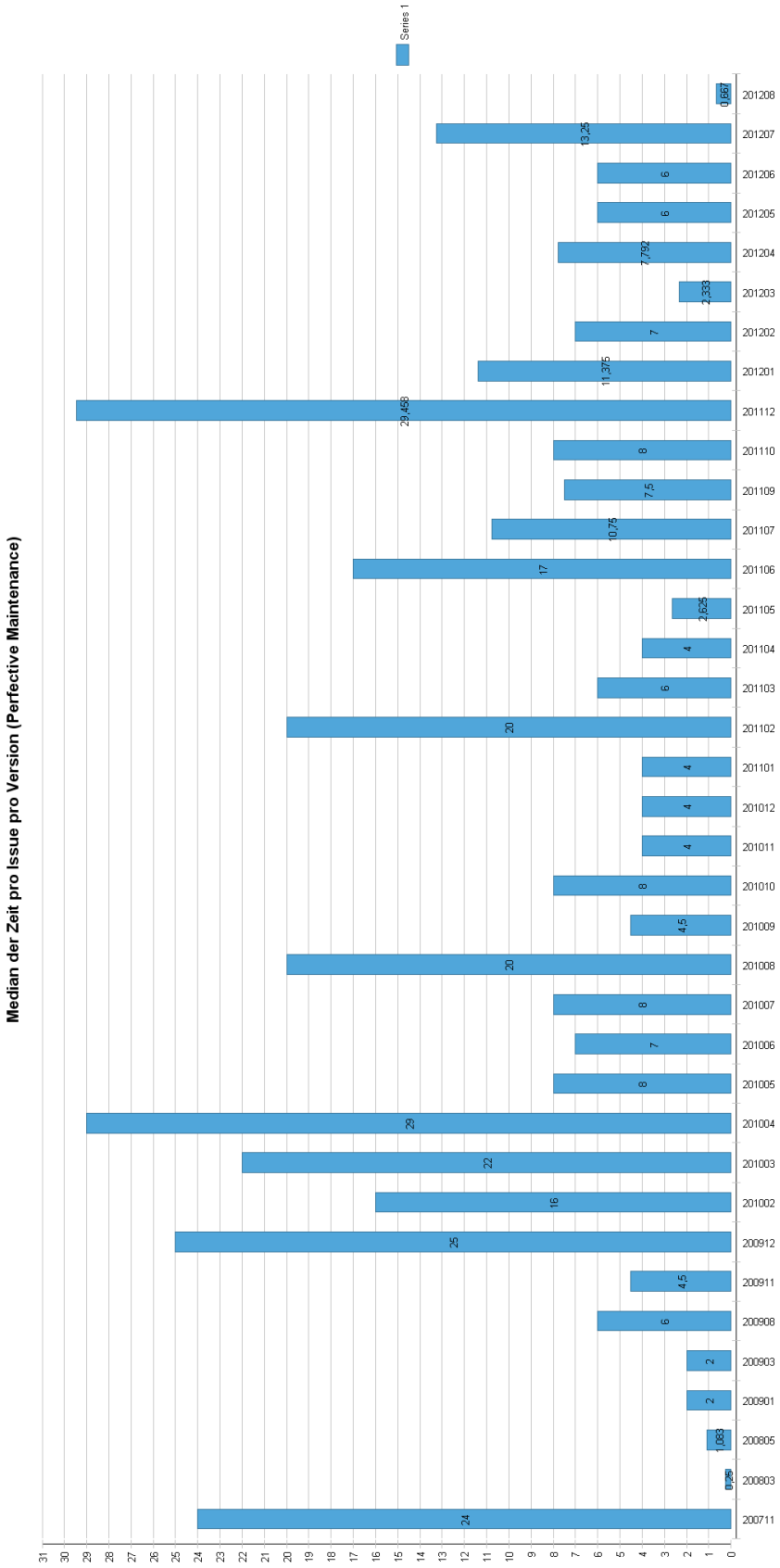

Abbildung A.6.: Median der Zeit für ein Issue (perfektionierende Wartung)

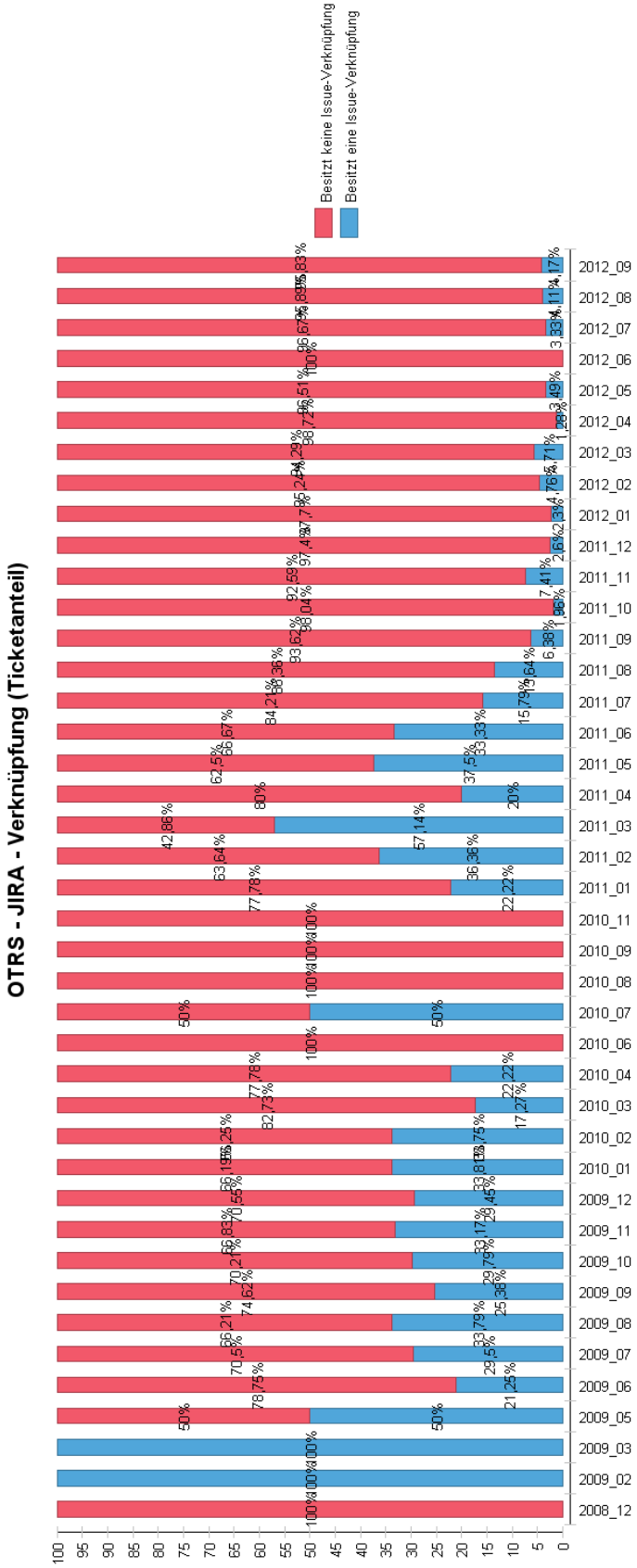

Abbildung A.7.: Verlauf der Verknüpfung zwischen OTRS und JIRA im 1st-Level-Support

LITERATURVERZEICHNIS

[AG] O. AG. OTRS. Website. Available online at http://www.otrs.com/de/;. (Zitiert auf Seite 14)

[Atla] Atlassian. Advanced Searching for JIRA. Website. Available online at https://confluence.atlassian.com/display/JIRA/Advanced+Searching;. (Zitiert auf Seite 20)

[Atlb] Atlassian. JIRA. Website. Available online at http://atlassian.com/software/jira/overview;. (Zitiert auf Seite 14)

[Atlc] Atlassian. JIRA. Website. Available online at https://confluence.atlassian.com/display/JIRA/Supported+Platforms;. (Zitiert auf Seite 20)

[Atld] Atlassian. JIRA REST Java Client. Website. Available online at https://studio.atlassian.com/wiki/display/JRJC/Home;. (Zitiert auf Seite 20)

[Bas92] V. R. Basili. Software modeling and measurement: the Goal/Question/Metric paradigm. Technischer Bericht, Institute for Advanced Computer Studies Department of Computer Science University of Mayland, College Park, MD, USA, 1992. (Zitiert auf Seite 17)

[IEE90] IEEE. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, S. 1, 1990. doi:10.1109/IEEESTD.1990.101064. (Zitiert auf Seite 8)

[IEE10] IEEE. IEEE Standard Classification for Software Anomalies. *IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)*, -:C1 –15, 2010. doi:10.1109/IEEESTD.2010.5399061. (Zitiert auf den Seiten 9, 10 und 16)

[Lud10] H. L. Ludewig, J. *Software Engineering – Grundlagen, Menschen, Prozesse, Techniken. 2. Aufl.* dpunkt.verlag Heidelberg, 2010. (Zitiert auf den Seiten 8 und 10)

[Opf04] S. Opferkuch. Software-Wartungsprozesse - ein Einblick in die Industrie. *Fachbericht Informatik, Nr. 11/2004*, 2004. (Zitiert auf Seite 11)

Alle URLs wurden zuletzt am 29. 11. 2012 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Rick Krüger)