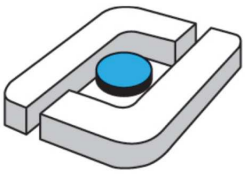

Fachhochschule Osnabrück
University of Applied Sciences

Fakultät Ingenieurswissenschaften und Informatik

Diplomarbeit

über das Thema

Konsolidierung eines IT-Systems zur Unterstützung fallbasierter Freier Software Dienstleistungen

vorgelegt durch

Torsten Irländer

Zusammenfassung

Zur Verwaltung von Kundenanfragen im Rahmen von Software Dienstleistungen werden heute spezialisierte IT-Systeme eingesetzt, die dabei helfen die anfallenden Aufgaben zu planen und organisieren. Dazu muss das System seine Benutzer in den verschiedenen Abläufen entlang des Kundendienstes bestmöglich unterstützen.

Gegenstand dieser Arbeit ist die Untersuchung und Verbesserung eines solchen Software Systems zur Unterstützung von Freien Software Dienstleistungen. Dazu werden zunächst die ablaufenden Prozesse in dem speziellen Umfeld des Kolab Kundendienstes analysiert, um festzustellen in wie weit die Prozesse von der vorhandenen Infrastruktur abgebildet und unterstützt werden können. Auf der Grundlage der Ergebnisse dieser Analyse findet dann eine Konsolidierung des IT-Systems statt, die im Wesentlichen aus der Konzeption und Entwicklung von Erweiterungen für das IT-System besteht.

Als Ergebnis dieser Konsolidierung stehen drei spezifische Erweiterungen einer Problemverfolgungs-Software bereit, mit deren Hilfe ausgewählte Prozesse während des Kundendienstes besser unterstützt werden können.

Danksagung

An dieser Stelle möchte ich all den Personen danken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Diplomarbeit beigetragen haben.

Besonderer Dank gebührt Herrn Prof. Dr. Jürgen Biermann und Herrn Dipl. Systemwiss. Bernhard Reiter für die Betreuung meiner Diplomarbeit. Sie standen mir stets bei allen aufkommenden Fragen in dieser Arbeit zur Hilfe und trugen durch viele kostbare Ratschläge und Hinweise zu einer Verbesserung dieser Arbeit bei.

Weiter bedanke ich mich bei meinen Eltern, die mir dieses Studium durch ihre Unterstützung ermöglicht haben.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Gestiegene Bedeutung von Software Dienstleistungen

Dienstleistungen für Software gewinnen zunehmend an Bedeutung. War in den Anfängen der kommerziellen IT die Software nur ein Nebenprodukt beim Verkauf der Hardware, kam es in der Mitte der 60er Jahre zu einem Phänomen, welches unter der Bezeichnung „Softwarekrise“ bekannt wurde. In dieser Zeit überstiegen die Kosten für Software erstmals die der Hardware. Die zunehmende Leistungsfähigkeit der Hardware ermöglichte durch Software weitaus komplexerere Lösungen , allerdings fehlten die nötigen Verfahren, um diese Art von komplexen Anwendungen qualitätsgesichtert entwickeln zu können. Edsger W. Dijkstras erläutert im Rahmen einer Dankesrede zum Touringpreis seine Sicht der Gründe für diese Krise:

> The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

Betrachtet man heute die TCO (Total Cost of Ownership) von Software, so stellt sich heraus, dass der Hauptkostenpunkt aus Sicht des Kunden nicht mehr die aus den Entwicklungskosten resultierenden Anschaffungskosten sind, sondern die Wartung und Pflege der auf Grundlage der Software errichteten Infrastruktur ist. Konsuequenter Weise müsste man in Analogie zur Softwarekrise nun von einer Wartungskrise sprechen.

Heute hat sich das Problem weiter verschoben. Die Entwicklung selbst großer Software ist durch viele Verfahren und Methodiken weitgehend beherrscht. Software ist mit den Jahren immer leistungsfähiger geworden und

durchdringt heute einen Großteil unseres täglichen Lebens. Unternehmen sind abhängig von dem fehlerfreien Betrieb ihrer IT-Infrastruktur, die in weiten Teilen durch Software bereit gestellt wird. Die Problematik heute liegt nicht mehr in der Entwicklung der Software, sondern in der Wartung und Pflege der Software. Diese Entwicklung führte dazu, dass Software heute zunehmend nicht mehr als Produkt verstanden wird sondern als Dienstleistung.

Software Dienstleister begleiten heute die Unternehmen bei dem Betrieb ihrer Software, kümmern sich um die Wartung und Pflege und stehen bei Fehlern und Fragen zur Hilfe. Um die verschiedenen Aufgaben, die sich im Rahmen einer solchen Dienstleistung ergeben, bewältigen zu können, werden spezialisierte IT-Systeme eingesetzt, die ihre Benutzer bei den täglichen Aufgaben während des Kundendienstes unterstützen.

Die Qualität und Effizienz einer solchen Software Dienstleistung ist maßgeblich davon abhängig, wie gut das eingesetzte IT-System in der Lage ist die im Kundendienst stattfindenen Prozesse abzubilden.

1.2 Problemstellung

Die Intevation GmbH bietet seinen Kunden die zuvor genannten Dienstleistungen auf der Basis von Freier Software. Das derzeit bei der Intevation zur Unterstützung eingesetzte IT-System wird den besonderen Anforderungen, unter denen Freie Software Dienstleistung stattfindet, nicht gerecht. Die während des Kundendienstes ablaufenden Prozesse werden nicht optimal unterstützt, was zu einem höheren Aufwand und Verzögerungen führt und somit zu einem Verlust von effektiver Arbeitszeit. Dies soll durch eine Konsolidierung des eingesetzten IT-Systems verbessert werden.

1.3 Zielsetzung

Im Rahmen dieser Arbeit findet vor dem Hintergrund dieser speziellen Anforderungen eine Untersuchung des IT-Systems bei dem Freien Software Dienstleister Intevation GmbH statt. Dazu werden zunächst die ablaufenden Prozesse in dem speziellen Umfeld des Kolab Kundendienstes analysiert, um festzustellen, in wie weit sich die Prozesse auf der vorhandenen Infrastruktur abbilden lassen. Auf der Grundlage der Ergebnisse dieser Analyse findet dann eine Konsolidierung des IT-Systems statt. Die Konsolidierung besteht im Wesentlichen aus der Entwicklung von Erweiterungen für das IT-System, mit deren Hilfe ausgewählte Abläufe während der Bearbeitung eines Falls besser unterstützt werden können.

Ziel dieser Arbeit ist die verbesserte Unterstützung von Abläufen im Rahmen von Freien Software Dienstleistungen durch das eingesetzte IT-System.

Neben diesem Hauptziel werden werden für einen erfolgreichen Abschluss folgende Sekundärziele festgelegt: Die Ergebnisse dieser Arbeit sind als Freie Software verfügbar und idealerweise in den Hauptstrom zurückgeflossen. Weiter befinden sich die Erweiterungen zum Abschluss dieser Arbeit im Produktivbetrieb bei der Intevation GmbH.

1.4 Gliederung der Arbeit

Kapitel 1 ist das aktuelle Kapitel und gibt über eine Einleitung eine erste Einführung in die Thematik dieser Arbeit. Es werden die Struktur und die Ziele der Arbeit beschrieben.

Kapitel 2 schafft die Grundlagen für das Verständnis dieser Arbeit. Neben der Beleuchtung des Umfeldes in dem diese Arbeit entstanden ist findet auch eine Deutung des Begriffs Freie Software, sowie eine Einführung in Trouble-Ticket Systeme und der Sprache Python statt.

Kapitel 3 gibt eine Übersicht über den Aufbau und Funktionalität des Roundup Server, der die Grundlage der Implementierung dieser Arbeit ist.

Kapitel 4 beschreibt die Ausgangssituation im Kundendienst der Intevation GmbH. Die Bestandsaufnahme beinhaltet eine Identifikation der Geschäftsprozesse, der beteiligten Personen und der zugrunde liegenden Infrastruktur.

Kapitel 5 nennt mit den Ergebnissen einer Analyse der Bestandsaufnahme die potentiellen Probleme in den Abläufen der Geschäftsprozesse. Zudem wird gezeigt an welchen Punkten eine Unterstützung der Geschäftsprozesse durch ein IT-System sinnvoll erscheint.

Kapitel 6 definiert die Erweiterungen des Roundup, die zur Lösung erkannten Probleme implementiert werden sollen. Die Erweiterungen werden in ihren grundsätzlichen Anforderungen beschrieben und in Muss- und Wunschkriterien unterteilt.

Kapitel 7-10 geben Einblicke in die Analyse und das Design der Erweiterungen. Nach einer Einführung in das Konzept werden die Datenstruktur und der Aufbau der Erweiterung beschrieben.

Kapitel 11 zeigt die Vorgehensweise bei einer Auswahl einiger Schritte während der Implementation. Die Betrachtung umfasst von der Installation des Roundup, über eine Beschreibung der Einbindung der Erweiterung bis hin zur Internationalisierung und der Migration bestehender Daten ein breites Spektrum verschiedener Aspekte der praktischen Realisierung.

Kapitel 12 schließt diese Arbeit mit einer kurzen Diskussion der Ergebnisse ab. Es findet eine Überprüfung der Ziele, sowie eine Bewertung der Vorgehensweise statt. Zuletzt werden offene Probleme benannt und ein Ausblick auf lohnende Untersuchungen gegeben, die auf dieser Arbeit aufbauen können.

Teil I

Grundlagen

Kapitel 2

Grundlegende Begriffe und Konzepte

Dieses Kapitel soll dem Leser helfen, einen möglichst einfachen Einstieg in die Thematik dieser Diplomarbeit zu finden. Hierzu werden einige für diese Arbeit grundlegende Begriffe und Konzepte erklärt.

2.1 Freie Software

Am Anfang war alle Software frei. In den frühen Jahren der Softwareentwicklung war es gängige Praxis Erweiterungen und Verbesserungen an der Software unter den Entwicklern auszutauschen, um so von der Arbeit und Wissen anderer zu profitieren. Es herrschte ein Klima des freien Wissensaustausches.

Die Frage der Freiheit dieser Entwicklungen spielte zu dieser Zeit noch keine Rolle, sondern wurde erst mit der zunehmenden Kommerzialisierung der Softwaresysteme, und der damit einhergehenden Gefährdung dieses "freien" Wissensaustausch aufgeworfen. Weiterentwicklungen waren zunehmend nicht mehr frei verfügbar, sondern konnten nur durch Zahlung stetig steigender Lizenzgebühren erworben werden.

Motiviert durch diese Entwicklung gründete Richard Stallman im Jahre 1983 das GNU-Projekt mit dem Ziel ein freies Unix zu entwickeln (Stallman, 1983), und legte damit den Grundstein des Begriffs der Freien Software, so wie er heute verstanden wird.

Für eine erste Annäherung an den Begriff der Freien Software eignet sich ein Zitat von Richard Stallman (Stallman, 1985), in dem er die Zweideutigkeit des Begriffs "free" in der englischen Sprache auflöst.

> "Free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech," not as in "free beer."

Der Kerngedanke der Freien Software ist die Rechte der Benutzer zu schützen. Das Recht Software zu nutzen, zu kopieren, zu modifizieren und zu verteilen. Software ist frei im Sinne von Freier Software, wenn sie dem Benutzer alle der folgenden Grundfreiheiten einräumt:

Freiheit 1) Die Software für jeden Zweck einzusetzen

Freiheit 2) Die Software in ihrer Funktionsweise zu untersuchen und auf die eigenen Bedürfnisse anzupassen. Der Zugang zu dem Quellcode der Software ist eine Vorbedingung für diesen Punkt

Freiheit 3) Die Software weiter zu verteilen

Freiheit 4) Die Software zu verändern und die Veränderungen weiterzugeben. Der Zugang zu dem Quellcode ist eine Vorbedingung zu diesem Punkt

Um diese Freiheiten dauerhaft zu sichern, entwickelte Richard Stallman die General Public License (GPL) (FSF, 1992), die gewährleistet, dass unter der GPL lizenzierte Software nur unter den gleichen Bedingungen weitergegeben werden kann. Richard Stallman schuf mit der GPL erstmals eine Lizenz, die die Rechte der Nutzer schützte und prägte mit ihr den Begriff des "Copylefts", der zusätzlich den Unterschied zu dem bislang bekannten "Copyrights" aufzeigt.

2.1.1 Freie Software und Open Source

Ende der 90er Jahre erfuhr die Freie Software große Aufmerksamkeit und das Interesse namhafter Firmen an den Konzepten Freier Software stieg. Die Aufmerksamkeit dieser Firmen galt aber weniger dem Gedanken der Freiheit einer Software, sondern vielmehr den praktischen Auswirkungen, die Freie Software mitbrachte: Freie Software gilt als leistungsfähig und zuverlässig. Diese Eigenschaften werden dem Entwicklungsmodell von Freier Software zugeschrieben in der die Offenheit der Quelldateien ein zentraler Punkt ist.

Als Folge dieser Entwicklung erkannten einige Kräfte der Freien Software Bewegung den Bedarf eines neuen Marketing-Begriffs für Freie Software (OSI, 2000). Dieser reduziert die Gedanken hinter der Freien Software auf den Punkt der Quelloffenheit. Der Gedanke der Freiheit einer Software, also die Frage was mit der Software getan werden kann, rückte nicht nur in den Hintergrund, sondern wurde im Sinne der Vermarktung als hinderlich empfunden.

Auch wenn die phänomenologischen Auswirkungen hinter beiden Begriffen in weiten Teilen die gleichen sind, so kommt es immer wieder zu einer fälschlichen Gleichsetzung beider Begriffe. Open Source beschreibt nur eine Teilmenge der Aspekte Freier Software.

Anders als "Open Source", in der die Quelloffenheit und kostenlose Weitergabe der Software im Vordergrund steht, befasst sich Freie Software darüber hinaus auch auch mit grundsätzlichen Fragen des freien Wissensaustausches als Grundlage einer menschlichen Gesellschaft (Grassmuck, 2002) und überschreitet hier klar die Grenzen in denen sich "Open Source" bewegt. In einer immer stärker von Computer und Software durchdrungenen Welt (Informationsrevolution) gewinnt der Kerngedanke der Freien Software auch in anderen Teilen der Gesellschaft an Bedeutung.

2.1.2 Aspekte Freier Software Entwicklung

Mit der proprietären und freien Softwareentwicklung stehen sich zwei grundlegend verschiedene Modelle gegenüber. Man spricht von Freier Software Entwicklung, wenn das Ergebnis der Entwicklung frei im Sinne von Freier Software ist.

Die Entwicklung kann dabei sowohl offen als auch geschlossen ablaufen. Eine offene Entwicklung zeichnet sich dadurch aus, dass eine Möglichkeit besteht, sich in den Entwicklungsprozess einbringen zu können. Ein Beitrag zur Entwicklung kann dabei auf vielfältige Weise geschehen, und reicht von einer Fehlerbehebungen (Patch), über die Übersetzungen der Dokumentation, bis hin zur Pflege der zugrunde liegenden Infrastruktur, oder dem Erstellen von Fehlerberichten (Bugreports).

Eine geschlossene Entwicklung bietet diese Möglichkeit nicht. Die Entwicklung verläuft nicht transparent für Außenstehende und es gibt keine Möglichkeit direkt auf die Entwicklung Einfluß zu nehmen. Das Ergebnis dieser Entwicklung ist aber trotz allem Freie Software. Tatsächlich findet ein großer Teil der Freien Software Entwicklung eher geschlossen statt. Die Entwicklungen der ersten Werkzeuge im Rahmen des GNU-Projekts gehören z.B. dazu. Auch bei der Intevation GmbH verläuft ein großer Teil der Entwicklung geschlossen.

Dies ist eine wichtige Feststellung, denn oftmals wird der Unterschied zwischen Freier Software Entwicklung und proprietärer Entwicklung mit der vermeintlich zwingenden Offenheit des Entwicklungsmodells beschrieben.

So beschreibt Eric S. Raymond in dem Buch "The Bazar and the Cathedral" (Raymond, 2001) an dem Beispiel des Linux-Kernels diesen Unterschied bildhaft, indem er proprietäre Entwicklung mit dem Bau einer Kathedrale beschreibt, wogegen die Entwicklung Freier Software eher dem brodelnden Treiben auf einem Basar gleicht. Während sich in der proprietären Entwicklung ein kleiner abgeschlossener Kreis von Entwicklern im Verborgenen über das Konzept, Design und die Implementation entscheiden, sind diese Prozesse bei der Entwicklung Freier Software für alle sichtbar und beeinflußbar. Was für den Linux-Kernel zutrifft lässt sich aber nicht verallgemeinern. Richtig ist, dass Freie Software die Voraussetzung für die von Raymond beschriebene offene, kooperative Entwicklung ist. Wirklich praktikabel wurde eine derart

offene Entwicklung aber erst mit der Verbreitung des Internet. Die Gründe hierfür liegen in den in Abschnitt 2.1 aufgeführten Freiheiten. Der tatsächliche Unterschied zwischen proprietärer und freier Softwareentwicklung liegt also nicht darin *wie* etwas entwickelt wird, sondern vielmehr *was* entwickelt wird.

Im Gegensatz zu proprietären Software ist der Quellcode bei Freier Software frei verfügbar und ermöglicht Entwicklern diese Software, oder Teile von ihr, als Grundlage ihrer eigenen Entwicklung zu nutzen. Freie Software Entwicklung fängt in der Regel nicht bei Null an, sondern baut auf etwas schon vorhandenem auf. Die Ergebnisse dieser Arbeit, sowie auch der Kolab Server (siehe Abschnitt 2.2.2) sind hierfür nur zwei Beispiele.

Grundsätzlich lässt sich feststellen, dass durch ein offenes Entwicklungsmodell die Voraussetzung für positive Synergieeffekte hergestellt werden.

2.2 Freie Software Dienstleistung

Als eine Teilmenge der allgemeinen IT-Dienstleistungen beschränken sich Freie Software Dienstleistungen auf Konzepte und Lösungen der Freien Software. Diese Dienstleistungen reichen von der Beratung über die Planung, bis hin zur Ausführung von Hard- und Software-Leistungen, und umfassen auch die unterstützende Tätigkeit (Kundendienst) beim Betrieb von Software Installationen.

Freie Software bietet aufgrund seiner Freiheiten die ideale Grundlage für eine solche Dienstleistung. Durch den freien Zugriff auf den Quellcode einer Software und der Freiheit diese zu verändern, eröffnet sich die Möglichkeit Fehler in einer Software zu beheben, sie in ihrer Funktionalität zu erweitern oder auf spezielle Gegebenheiten anzupassen. Dienstleister im Bereich der Freien Software können diese Freiheiten nutzen, um maßgeschneiderte Lösungen für den Kunden zu erstellen.

Das Modell der Freien Software Entwicklung befreit den Kunden aus der Abhängigkeit zu einem Softwarehersteller, indem es dem Kunden die Möglichkeit bietet aktiv in die Entwicklung der Software einzugreifen.

2.2.1 Kolab-Kundendienst bei der Intevation GmbH

Als einer von drei Trägern des Kolab-Konsortiums[1] bietet die Intevation GmbH[2] mit ihren Partnern folgende Leistungen:

- Unterstützung bei dem Betrieb eines Kolab Servers und seiner Clients. Dazu gehört eine zeitnahe Erstellung von Sicherheits Updates der Software.

[1] http://www.kolab-konsortium.com
[2] http://www.intevation.de

- Hilfestellung bei der Realisierung der Einführung einer Kolab-basierten Groupware-Lösung unter Berücksichtigung der Migration bestehender Daten und Systeme.
- Individuelle Erweiterungen und Anpassungen an die speziellen Bedürfnisse der Kunden.

Im Rahmen dieser Dienstleistung spielt die Intevation GmbH die Rolle einer Schnittstelle zwischen den Kunden und den Partnern. Partner sind externe Firmen, die ebenfalls aus dem Bereich der Freien Software stammen und in erster Linie für die Weiterentwicklung und Wartung der Software verantwortlich sind. Hierdurch ergibt sich eine Dreiecksbeziehung (siehe Abbildung 2.1) zwischen Kunde, Partner und Intevation in der die Dienstleistung rund um den Betrieb eines Kolab Servers stattfindet. Der Kunde erhält auf-

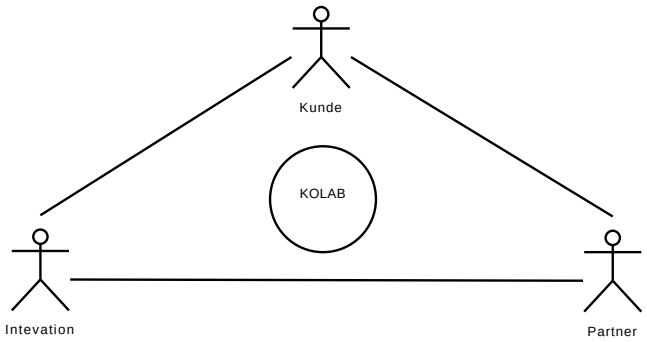

Abbildung 2.1: Dreiecksbeziehung im Kolab Support

grund der Freiheiten die Freie Software ihm bietet die Möglichkeit, über die Intevation gezielt die Weiterentwicklung des Kolab Server oder seines Clients zu beauftragen. Die Kosten für die Entwicklung trägt der Kunde, und die Ergebnisse sind als Freie Software verfügbar, bzw. fließen in den Hauptstrom zurück.

2.2.2 Kolab Server

Kolab ist eine umfassende und flexible Lösung für Arbeitsgruppen, mit der es möglich ist auch in heterogenen Netzen Email, Kontakte und Termine zu verwalten. Die Entstehung von Kolab geht zurück auf eine Ausschreibung des Bundesamt für Sicherheit und der Informationstechnologie[3] (BSI), mit dem Ziel eine auf Freier Software basierenden Arbeitsgruppen-Lösung zu erstellen. Clientseitig stehen eine speziell angepasste Version des Programm *Kontact*

[3] http://www.bund.bsi.de

aus dem KDE-Projekt[4] zur Verfügung, sowie durch die Erweiterungssoftware *Toltec* [5] auch Microsoft Outlook. Die Anpassungen an den *Kontact* Client fließen wiederum in das KDE-Projekt zurück. Herzstück dieser Lösung ist der Kolab-Server, der eine aufeinander abgestimmte Zusammenstellung aus bewährten und robusten Komponenten der Freien Software ist. Die Kommunikation zwischen diesen Komponenten wird über den Kolab-Daemon und einer Reihe von Perl und PHP-Skripten gewährleistet. So stellt sich der Kolab-Server seinen Nutzern, trotz seiner vielen einzelnen Komponenten, als homogene Einheit dar.

Es soll an dieser Stelle nicht weiter auf die interne Funktionsweise des Kolab-Servers eingegangen werden, da diese zum Verständnis der Arbeit nicht nötig ist. Für weitere Informationen bezüglich des Kolab-Servers wird der Leser auf die Webseiten des Kolab Projekts [6] verwiesen.

2.3 Trouble-Ticket Systeme

Der Begriff Trouble-Ticket System (TTS) bezeichnet im Allgemeinen ein Software System, das der Aufnahme und Dokumentation von Problemen und Fehlern dient. Alle zu einem Problem oder Fehler anfallenden Informationen werden in einem Trouble-Ticket (siehe Abschnitt 2.3.2) gespeichert. So wird aus einem TTS mit der Zeit ein wichtiges Informationssystem mit vielen Querverweisen für seine Benutzer. Innerhalb des Trouble-Ticket werden die Informationen oftmals in Form von chronologisch aufeinander folgenden Änderungsnotizen gespeichert. Diese dienen auch der Allgemeinen Kommunikation zwischen den Benutzern. Für seine Benutzer stellt das TTS ein zentrales Werkzeug, welches sie in allen Phasen des Fehler-Managements unterstützt. Eine ausführliche Beschreibung der Prozesse des Fehler-Managements bei der Intevation GmbH findet sich in Kapitel 4.

Ein Trouble-Ticket System verfügt meist über mehrere Schnittstellen, über die der Benutzer mit dem System kommunizieren kann. Die wohl am weitesten verbreitete Bedienung ist die über ein Web-Interface. Darüber hinaus bieten viele Systeme auch eine Email-Schnittstelle an, um neue Einträge erstellen bzw. verändern zu können. Zuletzt besteht auch der Zugriff über ein Konsolen-Interface als Möglichkeit offen. Insbesondere die letzten beiden genannten Wege werden oft dazu genutzt automatisiert neue Einträge in ein TTS erstellen zu können.

2.3.1 Ausprägungen von Trouble-Ticket Systemen

Trouble-Ticket Systeme bilden die Obermenge einer Vielzahl von spezialisierten Ausprägungen, die zu verschiedenen Zwecken eingesetzt werden, und

[4] http://www.kde.org
[5] http://www.toltec.co.za/
[6] http://www.kolab.org

dort zum Teil eine abweichende Bezeichnung tragen. Ähnliche Systeme sind Problem Tracking Systeme (PTS), Bugtracking Systeme (BTS), Feature Request Systeme (FRS) und Knowledge Data Base Systeme (KDB), welche alle sehr häufig in der Softwarebranche Verwendung finden. Es findet sich fast jede Kombination der Marketingschlagworte Trouble, Problem, Bug, Request, Feature, Ticket, Tracking und System (Trummer, 2004, S.9)

2.3.1.1 Bugtracker

Ein Bugtracker ist ein auf Softwarefehler spezialisiertes Tracking System, welches dem Entwickler ermöglicht den Überblick über die in einer Software gemeldeten Fehler (Bugs) zu behalten. In dieser Funktion stellt es ein zentrales Werkzeug in der Software-Entwicklung.

In einem offenen Entwicklungsmodell sind solche Bugtracker öffentlich zugänglich. Dies versetzt die Nutzer einer Software in die Lage sich über Fehler in der Software zu informieren, oder neue Fehlerberichte zu erstellen und so die Entwickler über ein Fehlverhalten der Software zu informieren. Bei solchen Fehlerberichten spricht man von "Bug-Reports". Ein solcher Bug-Report enthält eine Reproduktionsanleitung für den Fehler, Vermutungen über die Fehlerursache, Lösungsvorschläge, und neue Erkenntnisse zu einem Fehler.

An dieser Stelle soll darauf hingewiesen werden, dass die Verwendung des Begriffs *Bug* aus der Sicht eines Softwareentwicklers problematisch ist. Der Begriff wird gleichermaßen als Bezeichnung für eine fehlerhafte Stelle im Quellcode, einem fehlerhaften Programmzustand (z.B. Zeiger, die auf falsche Stellen im Speicher zeigen), oder einen tatsächlich sichtbaren Fehler (z.B. Absturz) eines Programms verwendet. Diese Unschärfe des Begriffs erschwert eine analytische Beschreibung eines Fehlverhaltens einer Software. Andreas Zeller empfiehlt in (Zeller, 2005) folgende Begriffe zu verwenden:

- *Defekt*: fehlerhafter Programmcode
- *Infektion*: fehlerhafter Programmzustand
- *Fehler*: sichtbares Fehlverhalten

Eine solche Unterscheidung ist im Rahmen dieser Arbeit aber nicht nötig, und wird daher nicht vorgenommen.

2.3.1.2 Support-Tracker

Support-Tracking Systeme verwalten und dokumentieren alle Formen von Kundenanfragen. Im Rahmen der in Abschnitt 2.2.1 genannten Kundendienstleistungen treten z.B. folgende Anfragen auf:

- Störungsmeldungen bzw. Fehlermeldungen im Betrieb der Software

- Erweiterungswünsche
- allgemeine Fragen

Es unterstützt den Mitarbeiter im Kundendienst bei der täglichen Arbeit mit dem Kunden, indem es alle zu einem Kunden anfallenden Daten speichert. Dies umfasst die Archivierung des gesamten Email-Verkehrs zwischen dem Kunden und dem Mitarbeiter, Gesprächsnotizen, sowie allgemeine Kommentare, oder Vermerke, die z.B. Hinweise auf das weitere Vorgehen enthalten. Für den Kunden eröffnet sich bei dem Einsatz eines Support-Tracking Systems oft die Möglichkeit den Fortschritt der Bearbeitung, der von ihm gemeldeten Fälle zu verfolgen. Dies erhöht die Transparenz des gesamten Vorgangs für den Kunden.

Im Kundendienst spielt ein solcher Support-Tracker oftmals die Rolle einer Schnittstelle zwischen dem Kunden und dem Mitarbeiter. Sämtliche Kommunikation zwischen dem Kunden und Mitarbeiter wird dann per Email, oder Einträgen über eine Web-Schnittstelle vorgenommen. In diesen Fällen spricht man von einem *Help Desk*, oder *Service Desk*.

2.3.2 Trouble-Ticket

Ein Trouble-Ticket stellt die elektronische Repräsentation eines Fehlers, eines Erweiterungswunsch, oder einer allgemeinen Kundenanfrage dar. Alle Daten und Zusatzinformationen, die während der Bearbeitung eines Falls anfallen, werden in diesem Ticket gespeichert. Auf diese Weise kann zu jedem Zeitpunkt genau nachvollzogen werden, aus welchem Verlauf sich der derzeitige Bearbeitungsstand ergibt. Besonders wenn mehrere Personen an der Bearbeitung eines Falls beteiligt sind, ist das zentrale Speichern der Dokumentation wichtig. In (rfc1297, 1992) werden zwei verschiedene Arten von Feldern unterschieden, die zur Speicherung der Daten dienen.

- Feste Felder
- Freie Felder

Feste Felder sind Auswahlfelder, die bereits eine Auswahl von gültigen Werten bereitstellen (Auswahlboxen), oder deren Bedeutung fest gelegt ist. Hierzu zählen z.B. Felder zur Speicherung von Namen, Telefonnummern und Schlüsselworte zu dem Fall. Freitextfelder können genutzt werden, um generische Informationen, wie z.B die Dokumentation der Fallbearbeitung, zu speichern.

Mit der Größe eines Trouble Ticket Systems steigt die Bedeutung von indizierbaren und überprüfbarer Informationen zu einem Fall. Feste Felder werden diesen Anforderungen weit besser gerecht als Freitextfelder. So kann bei der Eingabe eines Namens z.B. geprüft werden, ob die Person schon in der Datenbank existiert. Weiter können feste Felder als Suchkriterium in einer Suchanfrage genutzt werden.

2.4 Python

Python ist eine universelle, objektorientierte Interpretersprache, die sowohl für eigenständige Programme, als auch für Scripte in verschiedensten Anwendungsbereichen eingesetzt wird. Python wurde von Guido van Rossum im Jahre 1991 veröffentlicht und genießt den Ruf eine äußerst dynamische Sprache zu sein, die zu einer signifikanten Effektivitätssteigerung während des Programmierens führt.

Dieser Ruf wird von einer Studie von Lutz Prechelt gestützt (Prechelt, 2000). In der Studie werden sieben Sprachen in den Punkten Laufzeit, Programmstruktur, Sicherheit und Programmieraufwand (Lines of Code) vor dem Hintergrund eines zu lösenden Problems untersucht. Als Ergebnis dieser Studie zeigt sich, dass die Arbeit mit Scriptsprachen wie z.B. Python produktiver im Vergleich mit konventionalen Sprachen (C++, Java) ist. Der oft genannte Geschwindigkeitsvorteil der konventionellen Sprachen wirkte sich hingegen nur unwesentlich aus.

Eric S. Raymond beschreibt diese Produktivitätssteigerung in einem Artikel des Linux-Journals (Raymond, 2000) mit den folgenden Worten ...

> An important measure of effort in coding is the frequency with which you write something that doesn't actually match your mental representation of the problem, and have to backtrack on realizing that what you just typed won't actually tell the language to do what you're thinking. An important measure of good language design is how rapidly the percentage of missteps of this kind falls as you gain experience with the language.

Python fördert durch seinen einfachen Befehlssatz, der in wenigen Tagen nahezu komplett zu erlernen ist genau das von Raymond beschriebene.

Python ist als Interpretersprache plattformübergreifend. Der Quellcode wird ähnlich wie bei Java in einen Bytecode übersetzt, der von dem Interpreter ausgeführt wird. Dies führt neben der plattformunabhängigkeit auch zu einem Plus an Geschwindigkeit, da die Sprache nicht mehr zur Laufzeit übersetzt werden muss.

Die Sprache bringt eine Vielzahl von Bibliotheken mit, wodurch sie für den Einsatz in vielen Bereichen gut vorbereitet ist. Zudem verfügt Python über eine gute Bindung an andere Sprachen wie z.B. C, um dort zeitkritische Funktionen implementieren zu können. Derzeit steht die Sprache in der Version 2.5 zur Verfügung.

Für weitere Informationen zu Python wird auf die offizielle Python Internetseite (van Rossum, 1991), sowie eines vielen verfügbaren Büchern zu der Sprache verwiesen. Exemplarisch sei hier das Buch *"Learning Python"* aus dem O'Reilly Verlag genannt.(Lutz u. Ascher, 2003)

Kapitel 3

Roundup

Roundup bildet die Grundlage des in dieser Arbeit erstellten IT-Systems zur Unterstützung fallbasierter Freier Softwaredienstleistungen. Aufgrund dieser zentralen Bedeutung soll dieser Abschnitt eine grundlegende Einführung und Überblick über den Aufbau und Funktion des Roundup Servers geben. Der Leser erhält so das nötige Hintergrundwissen, um die Arbeitsschritte in den folgenden Kapiteln nachvollziehen zu können. Für eine umfassendere und ausführliche Beschreibung der Funktionalität des Roundup wird auf die Dokumentation (Jones, 2006c) verwiesen.

3.1 Roundup - Eine Zusammenfassung

Roundup ist ein in Python implementiertes Tracking System. Auf der Webseite des Roundup Projekts[1] wird dieses als Issue-Tracking System bezeichnet. Diese Bezeichnung deutet an, dass sich Roundup weder klar als ein Bugtracking System, noch als Support-Tracking System einordnen lässt. In der Standard-Installation präsentiert sich Roundup universell und ist gut vorbereitet für den Einsatz in einer Vielzahl von Szenarien.

Roundup zeichnet sich durch verschiedene Punkte aus. Eine vollständige Übersicht der Funktionen des Roundup findet sich im Anhang A.4.

Einfache Installation Die minimale Voraussetzung für den Betrieb eines Roundup ist die Installation von Python in einer Version >= 2.3. Roundup kann in der einfachsten Konfiguration als eigenständiger Server betrieben werden. Für mehr Leistung empfiehlt sich aber der Betrieb des Servers als Apache-Modul[2].

Einfache Bedienbarkeit Roundup lässt sich über drei Schnittstellen bedienen:

[1] http://roundup.sourceforge.net/
[2] http://www.apache.org

- Web-Oberfläche
- Email Schnittstelle
- Lokaler Zugriff über die Konsole

Roundup legt Wert auf eine gute Übersichtlichkeit der Weboberfläche, die alle wichtigen Informationen auf einen Blick präsentiert.
Ein Vergleich:
Bugzilla[3] zeigt auf der Startseite, abhängig von dem Layout und Auflösung des Bildschirms, auf den ersten Blick zwischen 7 und 8 Fälle. Roundup hingegen zeigt 25 wichtige Fälle.

Umfangreiche Dokumentation Die gute Bedienbarkeit wird zusätzlich von der guten verfügbaren Dokumentation unterstrichen. Neben der offiziellen Dokumentation gibt es ein von Benutzern gepflegtes Wiki[4], welches weitere Hinweise zur Erweiterung und Anpassung des Servers gibt.

Interne Mini-Mailinglisten Roundup kann für jeden Fall eine Mini-Mailingliste verwalten. Diese Mailingliste enthält Personen, die an Neuigkeiten in der Bearbeitung des Fall interessiert sind. Werden zu dem Fall neue Informationen hinzugefügt, so versendet Roundup automatisch eine Informationsmail an diese Personen. Eine solche Mailingliste wird "Nosy"-Liste genannt.

Hohe Flexibilität Roundup ist aufgrund seiner guten Erweiterbarkeit und Anpassbarkeit sehr flexibel und lässt sich daher gut für bestimmte Aufgaben spezialisieren. Zu den möglichen Einsatzfeldern gehört die

- Verwaltung von Softwarefehlern
- Verwaltung von Kundenanfragen (Helpdesk)
- Verfolgung der Firmen Akquise
- Organisation von Aufgaben in Arbeitsgruppen

oder Allgemein: Überall dort, wo eine große Anzahl von Aufgaben verwaltet, und unter verschiedenen Beteiligten organisiert werden müssen. In Abschnitt 3.5 wird näher auf die Erweiterbarkeit eingegangen, aus der diese hohe Flexibilität resultiert.

Auch in großen Umgebungen performant Roundup skaliert gut. Dank verschiedener unterstützter Datenbanken lässt sich Roundup in kleinen, mittleren und großen Umgebungen mit mehreren tausend Fällen einsetzen. Zu den unterstützen Datenbanken gehören:

- PostgreSQL[5]

[3] http://www.bugzilla.org/
[4] http://www.mechanicalcat.net/tech/roundup/wiki/FrontPage
[5] http://www.postgresql.org/

- MySQL[6]
- sqlite[7]
- metakit[8]
- anydbm

Rollenbasiertes Rechtesystem Roundup verfügt über ein rollenbasiertes Rechtesystem, über das fein graduierte Zugriffsrechte auf die Daten in der Datenbank erstellt werden können. Die Zugriffsrechte sind an Rollen gebunden, die dem Benutzer zugewiesen werden. Durch ein solches System ist es möglich eine komplexe Rechteverwaltung zu erstellen, in der Personen unterschiedliche Zugriffsrechte auf die gespeicherten Daten haben.

3.2 Design und Designziele von Roundup

Roundup basiert auf einem vom *Ka-Ping Ye*[9] erstellten Design, welches im Jahre 2000 den ersten Preis in Software Carpentry "Track" Wettbewerb[10] gewonnen hat. Roundup wurde das erste mal im Jahre 2001 veröffentlicht und wird bis heute durch eine Gruppe von Entwicklern unter der Leitung von Richard Jones weiterentwickelt.

Eines der Hauptziele bei dem Design von Roundup ist die Modellierung einer Lösung für das Problem, dass sich Fälle, oder generell Daten in der Datenbank, nicht immer klar in eine Kategorie einordnen lassen. Ziel war es eine optimale Modellierung der Beziehungen zwischen den Daten zu erreichen. Ka-Ping Ye sagt dazu in Yee (2004)

> Often when classifying information we are asked to select exactly one of a number of categories or to fit it into a rigid hierarchy. Yet things only sometimes fall into one category; often, a piece of information may be related to several concepts. For example, forcing each item into a single topic category is not just suboptimal but counterproductive: seekers of that item may expect to find it in a different category and conclude that the item is not present in the database – which has them worse off than if the items were not categorized at all.

Roundup setzt in bei dem Design auf ein Schichtenmodell (siehe Abschnitt 3.4), in dem sich mit der *Hyperdatabase* eine Schicht befindet, die genau diese Modellierung der Beziehung zwischen den Daten leistet. Die Daten werden

[6] http://www.mysql.de/
[7] http://www.sqlite.org/
[8] http://www.equi4.com/metakit/
[9] http://zesty.ca/
[10] http://www.software-carpentry.com/

dabei in Abschnitt 3.3 beschriebenen Datenmodell organisiert. In den folgenden Abschnitten findet eine weitere Vertiefung in einige der angesprochenen Bereiche statt.

3.3 Datenmodell

Roundup speichert seine Daten als Objekte die "Items" genannt werden. Diese Objekte enthalten eine Menge von Eigenschaften (Properties), in denen Informationen zu einem Objekt gespeichert werden können. Alle Objekte verfügen grundsätzlich über folgende Eigenschaften:

- Eine eindeutige ID zur Identifikation des Objekts in der Datenbank.
- Eine Historie, in der gespeichert ist wer, wann und was an einem Objekt geändert hat.
- Angaben über den Ersteller bzw. Bearbeiter des Objekts sowie Erstellungs und Bearbeitungszeitpunkt.

Diese werden durch den Roundup automatisch angelegt und gepflegt. Alle Eigenschaften, die über die genannten hinausgehen, sind optional und lassen sich über eine Konfigurationsdatei durch den Benutzer konfigurieren. Roundup verwendet zum Speichern dieser Eigenschaften sechs verschiedene Datentypen:

- Strings
- Boolsche Werte
- Numerische Werte (Floats)
- Datum bzw. Zeitintervalle
- Zeiger auf andere Objekte (Links)
- Liste von Zeigern auf andere Objekte (Multilink)

Eine zentrale Design-Entscheidung bei Roundup ist nach Yee (2004) eine starke Fokussierung auf die Modellierung der Beziehungen zwischen Objekten. Links bilden hierfür die Grundlage. Über sie lassen sich sowohl 1:1 Beziehungen (Links), also auch 1:N Beziehungen (Multilink) zwischen den Objekten herstellen.

Ein Beispiel:
Abbildung 3.1 zeigt einen sehr einfachen möglichen Aufbau eines Fall-Objektes (Issue). Um einen Fall für eine Suche besser kategorisieren zu können, lässt sich das Multilink-Feld *topic* dazu verwenden über die Angabe bestimmter

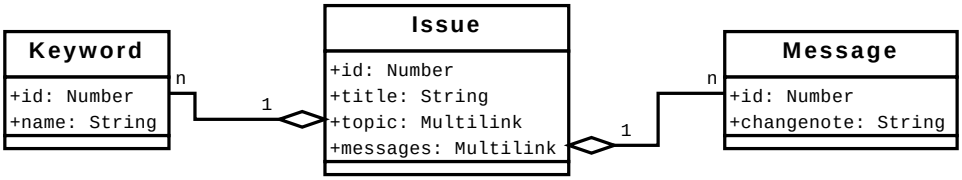

Abbildung 3.1: Einfacher Aufbau eines Falls

Schlüsselworte den Fall einen oder mehreren Themen zuordnen. Jedes dieser Schlüsselworte wird in Roundup als eigenes Objekt der Klasse Keyword gespeichert und kann auch von anderen Fällen zur Kategorisierung verwendet werden.

Durch die Verwendung von Links zur Modellierung der Beziehungen untern den Objekten, lassen sich beliebig komplexe Objekt-Konstruktionen erzeugen.

3.4 Schichtenmodell von Roundup

Roundup setzt in seinem Aufbau mit einem Schichtenmodell auf eine robustes und bewährtes Konzept der Informatik. Durch ein Schichtenmodell entsteht eine klare Trennung der Aufgaben der verschiedenen Schichten und eine Kommunikation findet nur über definierte Schnittstellen zwischen diesen Schichten statt. Roundup setzt sich aus insgesamt fünf Schichten zusammen.

Zwei der Schichten sind externen Komponenten zugeordnet. Abbildung 3.2 zeigt dies. Die grau hinterlegten Felder zeigen die externen Komponenten.

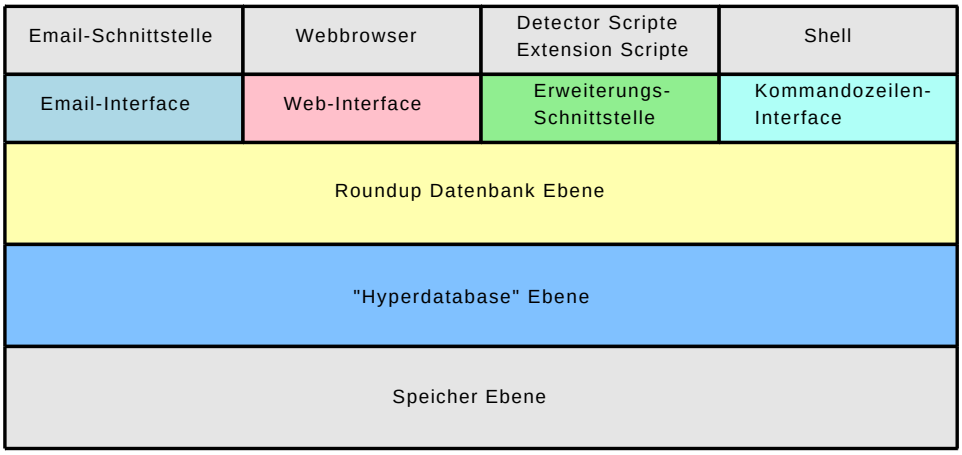

Abbildung 3.2: Schichten von Roundup nach Yee (2004)

In der obersten Schicht stehen Programme, mit denen auf den Roundup

zugegriffen werden kann, sowie Scripte die von dem Benutzer erstellt werden können, um das Verhalten des Roundup zu ändern bzw. zu erweitern. In der untersten Schicht (Speicherungs-Schicht) finden sich die Datenbanken in denen die Daten des Roundup persistent gespeichert werden.

In den mittleren drei Schichten liegen die Komponenten des Roundup.

3.4.1 "Hyperdatabase" Schicht

Die Hyperdatabase Schicht setzt auf der untersten Speicher-Schicht auf und dient der Abstraktion der Daten. Die Abstraktion findet auf der Grundlage der in Abschnitt 3.3 eingeführten Objekte statt. Sie bildet eine definierte Schnittstelle für alle darüber liegenden Schichten zur Speicherung der Daten. Für sie ist es völlig unerheblich welche Datenbank zur persistenten Speicherung in einem RDBM-System in der untersten "Speicher"-Schicht verwendet wird. Die Hyperdatabase sorgt für eine korrekte Übersetzung der Daten zwischen den beiden Schichten.

Die Einführung einer weiteren Datenbank-Schicht ist aus Sicht der Schnelligkeit und Leistung des Systems problematisch. Sie wird nach (Yee, 2004) mit der höheren Bedeutung einer Optimierung der Einfachheit im Vergleich zur Optimierung der Leistungsfähigkeit erklärt.

Das System ist nicht darauf optimiert mehrere tausend Anfragen pro Sekunde zu bearbeiten, sondern die Objekte und die Beziehung untereinander zu verwalten. Die Hyperdatabase modelliert die Beziehung, indem alle Objekte als Knoten gespeichert werden, die über die "Link"-Attribute der Objekte miteinander in Verbindung stehen.

3.4.2 "Roundupdatabase" Schicht

Die Roundupdatabase Schicht setzt auf der Hyperdatabase auf und übernimmt die Vermittlung der Anfragen darüber liegender Schichten an die Datenbank. Während die Hyperdatabase lediglich der allgemeinen Abstraktion der Daten auf der Grundlage von Objekten dient, erweitert die Roundupdatabase diese um Funktionen, die spezifisch für das Verwalten von Fällen sind. So erweitert die Roundupdatabase die Fälle um die Nosy-Listen und definiert die Benutzer. Zusätzlich wird die *Detector* Schnittstelle eingeführt, über die sich das Verhalten des Roundup durch den Benutzer verändern bzw. erweitern lässt.

3.4.3 "Interface" Schicht

Die Interface Schicht dient als Schnittstelle für den Zugriff durch externe Programme. Roundup bietet für den Zugriff drei verschiedene Methoden:

Web Schnittstelle Die Web Schnittstelle wird durch eine CGI Skript erzeugt, welches unter jedem Webserver betrieben werden kann. Die

Schnittstelle setzt sich dabei aus einer Reihe von HTML Dateien zusammen, die mit Elementen der Vorlage-Sprache TAL[11] durchsetzt sind. Diese Elemente sind Platzhalter für die tatsächlichen Werte aus der Datenbank und werden bei dem Aufbau der Seite durch diese ersetzt.

Email Schnittstelle Benutzer können durch das Senden einer Email an das System neue Fälle eröffnen, Eigenschaften eines Falls ändern oder weitere Kommentare zu dem Fall hinzuzufügen. Die Schnittstelle wird auch genutzt um Emails an den Benutzer zu verschicken, und ihn so z.B. über Neuigkeiten in der Fallbearbeitung, oder Fehler zu informieren.

Kommandozeilen Schnittstelle Die Kommandozeilen Schnittstelle ist einfach strukturiert und dient primär der Administration des Datenbestandes von Roundup. Trotz der Einfachheit bietet sie vollen Zugriff auf alle Daten.

Neben Schnittstellen für externe Programme (Webbrowser, Email) wird in dieser Schicht noch eine Erweiterungs-Schnittstelle definiert. Diese dient dem Einbinden von benutzerdefinierten Python-Scripten, über die sich die Funktionalität des Roundup erweitern und anpassen lässt.

3.5 Anpassung durch Modifikation der Trackerinstanz

Roundup bietet die Möglichkeit gleichzeitig mehrere verschiedene Tracker zu betreiben. Man spricht in diesem Fall von verschiedenen Trackerinstanzen. Jede Instanz verfügt über ein eigenes Verzeichnis, in dem die Dateien für den Aufbau der Weboberfläche, verschiedene Konfigurationsdateien und natürlich die Datenbank gespeichert sind. Die Instanzen sind daher völlig unabhängig voneinander. So ist es möglich, die Tracker durch Modifikation der Dateien in den jeweiligen Verzeichnissen, unabhängig von anderen Trackern auf spezielle Anforderungen anzupassen.

Die verschiedenen Möglichkeiten die sich zur Anpassung und Erweiterung einer Tracker Instanz anbieten werden im folgenden aufgeführt. Für ausführliche Erklärungen und Beispiele, wie solche Anpassungen in der Praxis konkret umgesetzt werden, wird auf Jones (2006a) verwiesen. Anpassungen und Erweiterungen des Roundup Tracker lassen sich an verschiedenen Stellen vornehmen:

Tracker Konfiguration Jede Instanz eines Roundup Trackers verfügt über ein Konfigurationsdatei, in der grundlegende Einstellungen vorgenommen werden. Dazu gehört z.B. die Angabe der Adresse unter der der

[11] http://wiki.zope.org/ZPT/TALSpecification14

Tracker erreichbar sein soll, oder die Definition von verschiedenen Email-Adressen über die der Tracker erreichbar ist und an die im Fehlerfall eine Benachrichtigung gesendet werden soll.

Datenmodell Wie bereits in Abschnitt 3.3 erwähnt, lässt sich das Datenmodell des Roundup durch eine Konfigurationsdatei erweitern und verändern. Um das Datenmodell auf seine eigenen Anforderungen anzupassen, lassen sich neue Tabellen in der Datenbank anlegen, oder bestehende erweitern.

Rechtesystem Roundup verfügt über eine rollenbasiertes Rechtesystem, das sich durch den Benutzer anpassen lässt. Es besteht die Möglichkeit neue Rollen und Befugnisse zu definieren, oder bestehende zu verändern.

Web-Schnittstelle Die Web-Schnittstelle wird aus mehreren HTML-Dateien zusammengebaut, die von dem Benutzer verändert werden können, um so das Erscheinungsbild der Weboberfläche anzupassen. Durch den Einsatz von Elementen der Vorlagesprache TAL lässt sich eine lesender und schreibender Zugriff auf Objekte in der Datenbank realisieren.

Detektoren Detektoren überwachen die Datenbank auf Veränderungen und werden automatisch ausgeführt sobald eine solche Veränderung festgestellt wird. Der Benutzer kann eigene Python-Scripte erstellen und diese als Detektoren einbinden, um so bestimmte Aktionen bei einem Zugriff auf die Datenbank ausführen zu lassen. Roundup unterscheidet bei den Detektoren dabei unter:

- Auditoren. Auditoren werden ausgeführt bevor der Schreibzugriff auf die Datenbank erfolgt.
- Reaktoren. Reaktoren werden ausgeführt nachdem der Schreibzugriff auf die Datenbank beendet wurde.

Sie bieten somit eine ähnliche Funktionalität wie die *Stored Procedures* und *Trigger*[12], eines PostgreSQL Server.

Erweiterungen (Extensions) Auch Extensions bieten dem Benutzer die Möglichkeit die Funktionalität des Trackers durch eigene Python Scripte zu erweitern. Die Möglichkeiten, die sich über eine solche Extension ergeben, sind vielfältig. Extensions können sowohl genutzt werden um komplexe Funktionserweiterungen in den Roundup einzubinden, als auch kleine Hilfsfunktionen zu schreiben. Eine solche Hilfsfunktion könnte die Generierung von Elementen in der Weboberfläche sein, die sich nicht ohne weiteres mit den Mitteln der Vorlagesprache TAL erstellen

[12] http://www.postgresql.org/docs/8.2/static/triggers.html

lassen. Die Funktionalität der Erweiterungen lässt sich an mehreren Stellen im Roundup nutzen. Die häufigste Nutzung ist aber das Einbinden in die die Weboberfläche.

Neben den Modifikationen innerhalb einer Instanz bietet sich aufgrund der Quelloffenheit des Roundup die Möglichkeit, Veränderungen direkt in den Bibliotheken des Roundup vorzunehmen. Diese Änderungen wirken sich dann jedoch nicht mehr lokal auf eine einzelne Instanz, sondern auf alle Instanzen aus.

Teil II

Analyse

Kapitel 4

Bestandsaufnahme

Im Rahmen dieses Kapitels wird eine Bestandsaufnahme vorgenommen, die zu einem Gesamtbild der Prozesse im Kundendienst führt. Hierzu wird in systematischen Schritten ein Fall aus dem Kundendienst in seine einzelnen Bestandteile zerlegt. Dies beinhaltet die Identifikation der ablaufenden Geschäftsprozesse, der involvierten Akteure, sowie der zugrunde liegenden Infrastruktur.

4.1 Identifikation der beteiligten Personen

Während der Bearbeitung eines Falls sind verschiedene Personen involviert, die dabei unterschiedliche Rollen übernehmen. Die Identifikation dieser Personen und ihrer Rollen ist Ziel dieses Abschnitts.

Mitarbeiter der Intevation Die Mitarbeiter der Intevation sind der zentrale Ansprechpartner für Kunden. Sie nehmen die Anfragen der Kunden entgegen und kümmern sich um eine qualitätsgesicherte Ausführung. In ihrer Rolle als Schnittstelle zwischen dem Kunden und den Entwicklern übernehmen Sie die Koordination und Organisation des gesamten Ablaufs einer Fallbearbeitung. Derzeit sind 5 Mitarbeiter der Intevation GmbH in die Fallbearbeitung involviert.

Kunden Der Kunde ist das Zentrum der gesamten durch die Intevation geleisteten Dienstleistung. Er tritt in erster Linie Anwender der Software auf. Kunden bestimmen maßgeblich die Entwicklung des Kolab Server und Client mit, indem sie Entwickler der Partnerfirmen beauftragen können Fehlerbehebungen oder Erweiterungen an der Software durchzuführen.

Die Menge der Kunden setzt sich derzeit aus drei festen Kunden und mehreren kleinen Kunden zusammen, die temporär den Kundendienst der Intevation in Anspruch nehmen.

Partner Partner unterstützen die Intevation GmbH bei der Erbringung der Dienstleistung rund um den Betrieb eines Kolab Servers. Aufgrund ihres Expertenwissen um die eingesetzte Software, sind sie in der Lage Fehler in dieser zu beheben, oder sie um Funktionen zu erweitern. Sie treten daher meist als Entwickler auf. Partner stehen in einem geschäftlichen Verhältnis zu der Intevation GmbH und sind daher in weiten Teilen über die internen Abläufe im Kundendienst informiert.

Entwickler aus dem Upstream Entwickler aus den Upstream-Projekten spielen im Kundendienst nur eine untergeordnete Rolle. Die Arbeit im Upstream ist aus Sicht der Intevation GmbH nicht planbar, da in diese Richtung keinerlei geschäftliche Beziehung besteht. Die Arbeit im Upstream erfolgt aus Sicht der Intevation freiwillig.

Anwender Anwender spielen in den Abläufen des Kundendienstes keine Rolle. Dennoch sollen sie an dieser Stelle erwähnt werden, da sie die Quelle von Fehlerberichten sein können neue Versionen der Software testen, und auf den Mailinglisten aktiv an einer Diskussion teilnehmen.

4.2 Betrachtung der Infrastruktur

Während der Bearbeitung eines Falls im Kundendienst laufen eine Vielzahl von Prozessen ab, die von dieser Infrastruktur möglichst optimal unterstützt werden sollen. Hierzu zählt die Kommunikation zwischen verschiedenen Personen, die Kontrolle und Steuerung der ablaufenden Prozesse, die Unterstützung bei der Softwareentwicklung und Fehleranalyse, sowie die Dokumentation des gesamten Vorgangs und deren Ergebnisse. Zu dieser Infrastruktur gehört:

Interner Support-Tracker Als interner Support-Tracker wird der in Abschnitt 2.3.1.2 beschriebene, *Request Tracker* (RT) in der Version 2 eingesetzt. Als Support-Tracker dient er der Speicherung, Verfolgung und Dokumentation des Bearbeitungsfortschritts von Kundenanfragen. Dies können sowohl Störungsmeldungen, Erweiterungswünsche, oder allgemeine Anfragen an den Kundendienst sein. Er enthält im Gegensatz zu dem öffentlichen Bugtracker kundenspezifische Angaben, die nicht nicht an die Öffentlichkeit gelangen dürfen und dem Datenschutz unterliegen. Aus diesem Grund ist der Support-Tracker nur für Kunden und Mitarbeitern der Intevation GmbH erreichbar.

Öffentlicher Bugtracker Der öffentliche Bugtracker (siehe hierzu auch Abschnitt 2.3.1.1), dient der Speicherung, Verfolgung und Dokumentation des Bearbeitungsfortschritts von Fehlern und Erweiterungswünschen in der Software (Server und Client). Diese Fehler sind bereits, oder können

potentielle Ursachen für Störungen bei dem Kunden sein. Im Gegensatz zu dem internen Support-Tracker beinhaltet der Bugtracker jedoch keine kundenspezifischen Daten mehr und ist öffentlich zugänglich. Der öffentliche Bugtracker basiert auf dem unter 3 beschriebenen Roundup Issue-Tracker.

Upstream Bug-Bugtracker Die Bugtracker der Upstream-Projekte enthalten Fehlerberichte zu den verwendeten Komponenten des Kolab Server oder des Kontact Client. Die Informationen in diesen Trackern können während der eigenen Fehleranalyse genutzt und erweitert werden.

Email Email ist neben dem internen und öffentlichen Tracker das primäre Kommunikationsmittel für alle an der Bearbeitung eines Falls beteiligten Personen.

Mailinglisten bieten die Möglichkeit mit einer ganzen Personengruppe in Kontakt zu treten, und so für alle transparent zu diskutieren. Hierzu gehören:

- Interne Partner Mailingliste (Partner, Mitarbeiter)
- Öffentliche Kolab Entwickler Mailingliste
- Öffentliche Kolab Benutzer Mailingliste
- Upstream Mailinglisten (KDE Kontact, Cyrus IMAP)

Die interne Mailingliste dient der Kommunikation zwischen der Intevation und seinen Partnern. Hier werden grundsätzlich Angelegenheiten behandelt, die nicht für die Öffentlichkeit bestimmt sind. Dies ist z.B. eine Aufwandsabschätzungen für die Implementation einer Erweiterung in der Software.

Auf der Entwickler Mailingliste sind die Entwickler, oder an der Entwicklung der Software interessierte Personen, zu finden.

Die Benutzer Mailingliste bietet Anwendern die Möglichkeit sich untereinander über die Anwendung, Installation und Konfiguration der Software auszutauschen. Zuletzt gibt es noch die Upstream Mailinglisten, die die Möglichkeit geben mit den Entwicklern des Upstream in Kontakt zu treten.

Telefon Der Kundendienst der Intevation GmbH ist für Kunden per Telefon erreichbar. Gerade die Aufnahme von neuen Fällen ist über das Telefon, aufgrund der direkten Kommunikation, viel besser zu erledigen als per Email.

Testsysteme Testsysteme dienen der Analyse von Störungen oder Testinstallationen (Smoke Tests). Um gesicherte Ergebnisse aus diesen Tests

zu erhalten, finden diese immer in einer Umgebung statt, die möglichst der Umgebung des Kunden gleicht.

Versionskontrollsysteme Während der Entwicklung werden die Versionskontrollsysteme SVN[1] und CVS[2] genutzt, um die gemeinsame Arbeit mehrerer Entwickler zu unterstützen. Die verschiedenen Entwicklungstände der Software sind hier verfügbar und können so für Tests genutzt werden.

4.3 Identifikation der Geschäftsprozesse

Die Identifikation der Geschäftsprozesse dient der Erfassung von Abläufen während der Bearbeitung eines Falls im Kundendienst. Ein Geschäftsprozess besteht aus mehreren zusammenhängenden, aufeinander abfolgenden Arbeitsschritten, die von einem Akteur durchgeführt werden, um ein Ziel zu erreichen. Sie haben einen definierten Anfang und Ende. Im folgenden werden die Geschäftsprozesse in chronologischer Reihenfolge aufgelistet.

4.3.1 Annahme und Erfassung

Jeder Fall im Kundendienst beginnt mit der Annahme und Erfassung eines Anliegen des Kunden. Initiiert wird dieser Schritt durch die Kontaktaufnahme des Kunden per Email oder Telefon. Abbildung 4.1 zeigt die an diesem Schritt beteiligten Personen und die Kommunikation anschaulich. Der Kunde

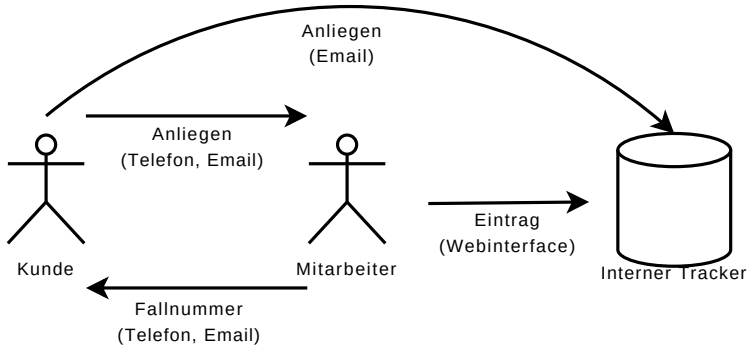

Abbildung 4.1: Beteiligte und Kommunikation bei der Fallaufnahme

kann auf zwei Wegen mit dem Kundendienst in Kontakt treten: Bei einem persönlichen Kontakt mit dem Kunden nimmt der Mitarbeiter das Anliegen des Kunden auf und legt einen neuen Fall im internen Tracker an. Diese

[1] http://svn.tigris.org
[2] http://www.nongnu.org/cvs/

Aufnahme umfasst zunächst nur eine umgangssprachliche Beschreibung des Anliegens, die mit ersten Vermutungen und Hinweisen für eine spätere Weiterbearbeitung angereichert sein kann.

Sendet der Kunde eine Email mit seinem Anliegen an das Tracker System, so wird dort durch die Email-Schnittstelle automatisch ein Eintrag generiert und der Mitarbeiter informiert.

Den Abschluss diese Schritts bildet die Zusendung der Fallnummer an den Kunden. Über diese Fallnummer kann der Kunde jederzeit Bezug auf den Fall nehmen, um beispielsweise Nachfragen zum Fortschritt zu stellen.

4.3.2 Klassifizierung

Fälle lassen sich meist bestimmten Kategorien zuordnen. Eine Kategorisierung findet hier sowohl auf Grundlage des Themenbereichs (Server, Client) statt, als auch nach der Priorität oder der Art des Falls (Störung, Anfrage, Erweiterungswunsch). Eine erste Klassifizierung findet in der Regel schon während der Aufnahme statt. Sie kann aber, aufgrund neuer Erkenntnisse, die sich bei der Bearbeitung des Falls ergeben, auch zu einem späteren Zeitpunkt erfolgen, und wird aus diesem Grund als eine eigenständige Aktivität betrachtet. Abbildung 4.2 macht die Zusammenhänge während einer Klassifizierung deutlich.

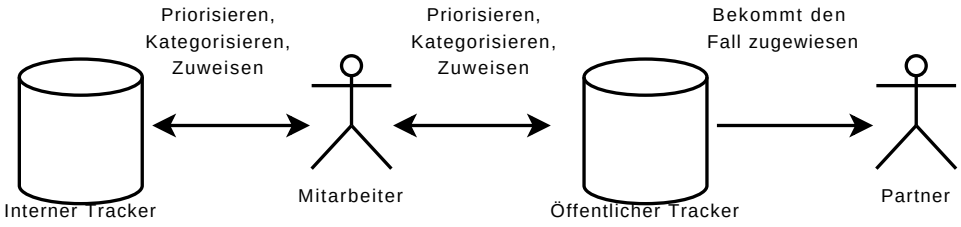

Abbildung 4.2: Beteiligte und Kommunikation bei der Klassifizierung

Die folgende Liste zeigt eine Reihe von Klassifizierungsmerkmalen:

- Zuweisen von Prioritäten
- Zuweisen einer Person, die die Bearbeitung übernimmt
- Zuweisen eines Bearbeitungsstatus (Neu, in Bearbeitung, Abgeschlossen)
- Einordnung in Kategorien
- Verknüpfen von Fällen

Eine Klassifizierung dient der Steuerung des Fallablaufs. Durch die Kategorisierung in Themenbereich und Prioritäten wird deutlich, welcher Mitarbeiter am Besten für die Bearbeitung geeignet ist, und wie dringend die

Bearbeitung ist. Zum anderen dient eine solche Klassifizierung der Aufbereitung der Falldaten in eine durchsuchbare Form. Das Einordnen des Falls in verschiedene Kategorien ist für eine spätere Suche im Rahmen der Überprüfung von Fehlermustern (siehe Abschnitt 4.3.3) hilfreich.

4.3.3 Überprüfung des Fallmusters

Nach einer Klassifizierung kann eine Überprüfung des Fallmusters erfolgen. Hierbei werden der interne Tracker, der externe Tracker und die Tracker aus den Upstream Projekten daraufhin untersucht, ob es bereits einen ähnlichen Fall gibt und ob für diesen eine Lösung verfügbar ist.

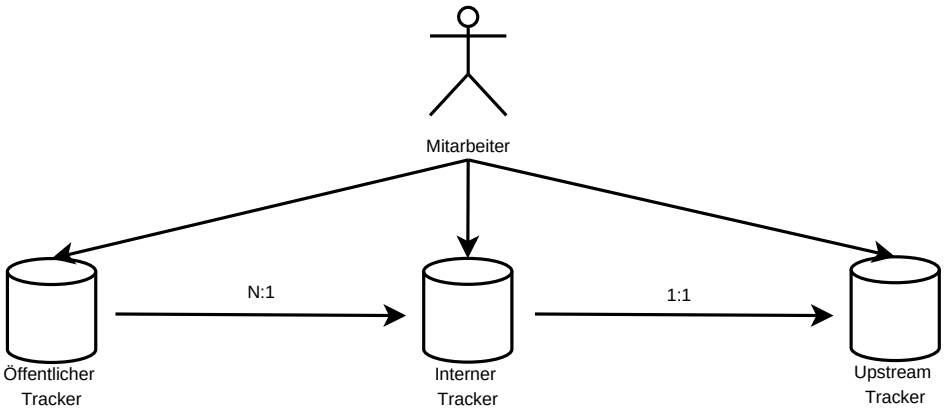

Abbildung 4.3: Überprüfung verschiedener Informationsquellen

Stellt sich heraus, dass es bereits einen ähnlichen Fall gibt, so wird auf diesen verwiesen, um eine Beziehung zwischen den Fällen herzustellen. Dies dient vor allem dazu auf bereits vorhandenes Wissen zu einem Fall zurückgreifen zu können. Eine solche Beziehung besteht oft zwischen Störungen bei dem Kunden und den technischen Ursachen im öffentlichen Tracker. Störungen bei verschiedenen Kunden haben oftmals dieselbe technische Ursache. Somit ergibt sich eine N:1 Beziehung zwischen den Fällen im internen und öffentlichen Tracker.

Liegt diese Fehlerursache in einer der Komponenten aus dem Upstream Projekten, so wird auf den Fall dort verwiesen. Somit ergibt sich eine Verkettung der Verweise zwischen den Trackern. Abbildung 4.3 veranschaulicht die Überprüfung der verschiedenen Informationsquellen.

4.3.4 Analyse und Diagnose

Die Analyse und Diagnose dienen der weiteren Klärung eines Falls. Typischer Weise handelt es sich bei der Analyse und Diagnose um die Unter-

suchung eines Fehlers. Grundsätzlich findet eine Analyse und Diagnose aber auch bei Erweiterungswünschen oder sonstigen Anfragen statt. Eine solche Untersuchung verläuft in zwei Phasen, die unterschiedliche Ziele verfolgen. Diese Ziele sind:

- die Erstellung eines detailierten Fehlerberichts (Analyse)
- die Klärung der Ursache für den Fehler (Diagnose)

Abbildung 4.4 zeigt die Kommunikation zwischen den beteiligten Personen und die genutzte Infrastruktur. Aufgrund der Übersichtlichkeit zeigt die Abbildung nur die Zusammenhänge aus Sicht des Mitarbeiters.

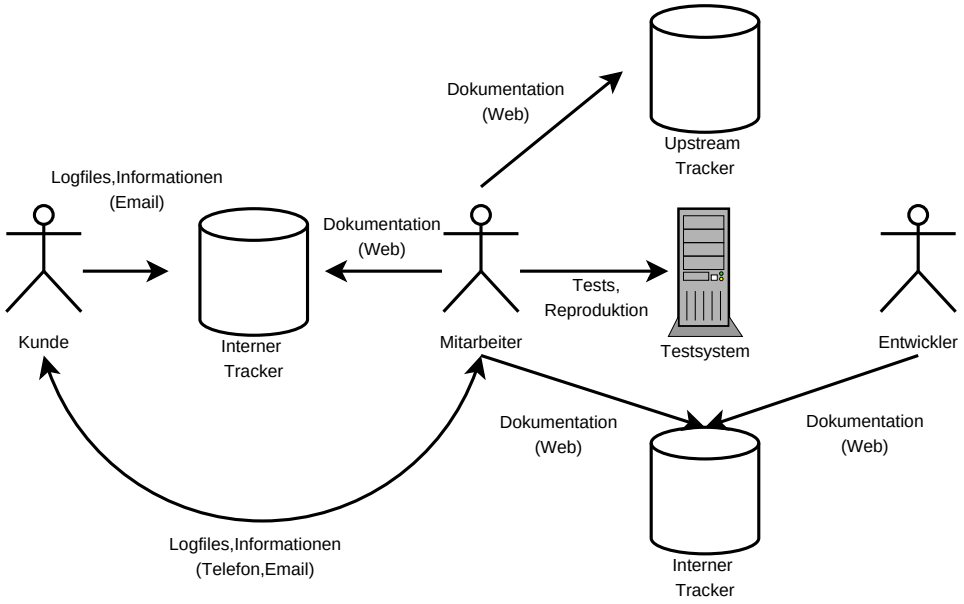

Abbildung 4.4: Abläufe während der Analyse und Diagnose einer Störung

In der ersten Analyse-Phase versucht der Mitarbeiter das beschriebene Fehlverhalten in einer Testumgebung nachzustellen. Der Mitarbeiter wird bei Bedarf durch den Kunden mit weiteren Informationen versorgt, die bei der Reproduktion des Fehlers hilfreich sein können (Logfiles, Benutzerdaten). Abhängig von den Ergebnissen der Untersuchung findet eine Veröffentlichung des Fehlerberichts im öffentlichen Tracker und/oder Upstream Tracker statt. Über die gesamte Zeit hinweg werden alle vorgenommenen Schritte im internen Tracker dokumentiert.

In der zweiten Phase wird der Fehler weiter untersucht, um die Ursache zu diagnostizieren. In dieser Phase sind meist Entwickler der Partnerfirmen eingebunden, die sich ihrerseits auf die Suche nach der Fehlerursache machen.

Die Kommunikation zwischen den Entwicklern und die Dokumentation des Vorgangs findet nun in großen Teilen über den öffentlichen Tracker statt.

Ist die Fehlerursache gefunden, wird diese mit dem Kunden kommuniziert. Der Kunde entscheidet dann, ob der Fehler behoben werden soll.

4.3.5 Beauftragung von Arbeiten

Wie in Abschnitt 4.3.4 beschrieben, haben Kunden die Möglichkeit Erweiterungen oder Fehlerbehebungen der Software zu beauftragen und so die Weiterentwicklung voranzutreiben.

Dieser Prozess wird durch den Kunden mit der Anfrage einer Aufwandsabschätzung zur Implementation der Erweiterung oder Fehlerbehebung angestoßen. Der Mitarbeiter dokumentiert diese Anfrage im internen Tracker und leitet sie über die interne Mailingliste an die Entwickler weiter. Die weitergeleitete Anfrage enthält einen Verweis auf den Fall im öffentlichen Tracker, in dem die technischen Details zu der Anfrage dokumentiert sind. Die auf der

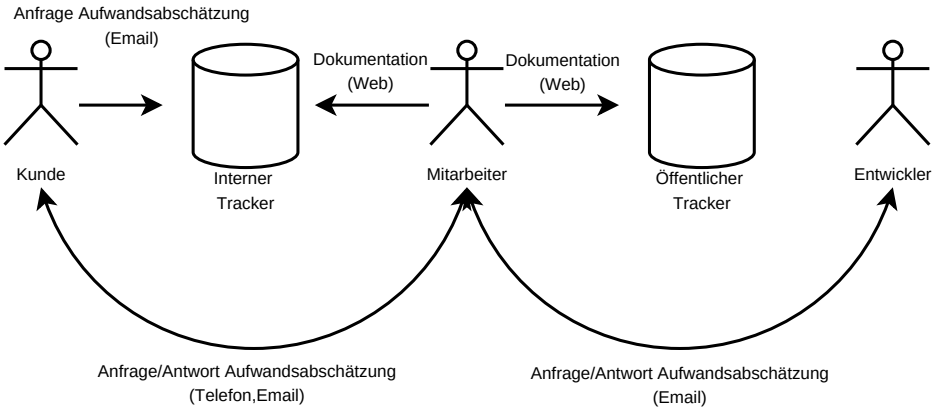

Abbildung 4.5: Abläufe während des Beauftragen einer Weiterentwicklung

Mailingliste vertretenen Entwickler geben nun eine Aufwandsabschätzung ab und senden diese an den Mitarbeiter zurück. Die Abschätzung wird erneut im internen Tracker dokumentiert und an den Kunden weitergeleitet.

Entscheidet der Kunde die Arbeit zu beauftragen, so wird der entsprechende Fall im öffentlichen Tracker als beauftragt markiert, wodurch der Fall für die Entwickler der Partnerfirmen eine höhere Priorität bekommt. Abbildung 4.5 zeigt die beteiligen Personen und genutzte Infrastruktur in diesem Prozess.

4.3.6 Beheben und Wiederherstellen

Steht eine Lösung aus der Analyse und Diagnose Phase bereit, so kann mit der Umsetzung dieser Lösung begonnen werden. Die Umsetzung der Lösung

findet großteils offen und transparent statt. Dies bedeutet vor allem,

- dass die Diskussionen über technische Details der Lösung über öffentlich zugängliche Medien, die Tracker oder Mailinglisten stattfindet und,
- dass die erstellten Lösungen frei verfügbar sind und,
- dass die Ergebnisse, sofern sinnvoll, wieder in die Upstream Projekte zurückfließen.

Der Prozess lässt sich in zwei Phasen einteilen: In die Implementation der Lösung und ihrer Qualitätsprüfung.

Die Implementation erfolgt durch die Entwickler, die ihre Ergebnisse in einem Versionskontrollsystem veröffentlichen. Steht eine Version bereit, wird der Fall einem Mitarbeiter zum Testen zugewiesen.

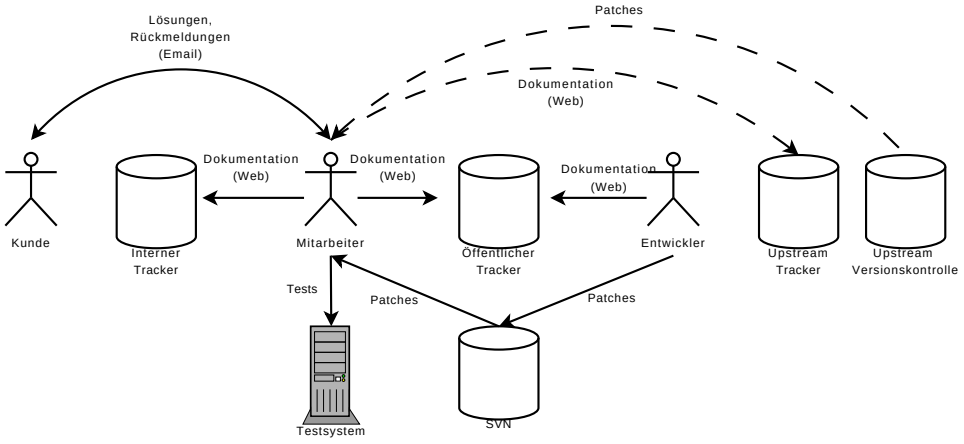

Abbildung 4.6: Abläufe während des Beheben und Wiederherstellen

Der Implementation folgt die Qualitätssicherung. Dazu bezieht der Mitarbeiter die fehlerbereinigte Version der Software aus dem Versionskontrollsystem und testet diese in einer Testumgebung. Diese Qualitätssicherung umfasst unter anderen folgende Punkte:

- Überprüfung auf Wechselwirkungen
- Rücksprachen
- Überprüfung der Fehlerfreiheit
- Vollständigkeit der Änderung
- Entspricht die Änderung dem Wunsch des Kunden?

Die Tests, sowie deren Ergebnisse werden über die öffentlichen Tracker kommuniziert und dokumentiert. Eine Ausnahme ist hier nur die Kommunikation mit dem Kunden.

4.3.7 Fallabschluss

Der letzte Schritt in der Bearbeitung eines Kundenfalls ist der Fallabschluss. Wurde das Anliegen eines Kunden zu seiner Zufriedenheit erfüllt, so kann der Fall abgeschlossen werden. Der Fallabschluss wird durch eine positive Rückmeldung des Kunden eingeleitet. Daraufhin wird ein ausführlicher Ab-

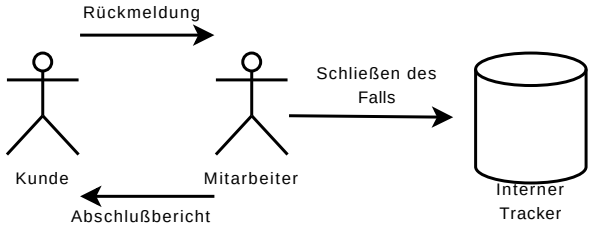

Abbildung 4.7: Abläufe während des Fallabschluß

schlussbericht erstellt, der dem Kunden zugesendet wird. Dieser beinhaltet eine allgemeine Beschreibung des beobachteten Fehlverhaltens, Erläuterungen zu dem Fehler, einer Beschreibung der erstellten Lösung, sowie ein Hinweis auf die Version der Software mit der die Lösung ausgeliefert wird.

Zuletzt wird die Lösung in dem internen Tracker dokumentiert und der Fall als abgeschlossen markiert. In Abbildung 4.7 ist der Fallabschluss dargestellt.

4.3.8 "Überwachen" von Fällen

Die Organisation von Abläufen bei der Bearbeitung von Kundenfällen ist eine der Hauptaufgaben im Rahmen des Kundendienstes. Ein großer Teil dieser Organisation ist die Überwachung des Fortschritts der Fälle. Die Überwachung ist ein Prozess ohne klaren Anfang und Ende und findet ähnlich wie die Dokumentation des Bearbeitungsfortschritts ständig statt.

Im Kundendienst für den Kolab-Server sind derzeit über 600 Fälle im öffentlichen Tracker in Bearbeitung. Zu diesen kommen noch einmal etwa 70 Kundenfälle in dem internen Tracker. Trotz der Verteilung der Fälle über mehrere Personen ist die Zahl der Fälle, die zur Bearbeitung einer Person zugewiesen wurden, noch hoch. Daher kann es vorkommen, dass ein Fall aufgrund einer geringeren Priorität über längere Zeit nicht bearbeitet wird. Wird dieser Zeitraum zu groß, gerät der Fall in Vergessenheit.

Die Überwachung des Bearbeitungsfortschritts von Fällen soll dieser Gefahr entgegenwirken. Fälle, die über einen längeren Zeitraum nicht bearbeitet

wurden, werden wieder angestoßen. Dies geschieht meist über eine kurze Meldung in dem entsprechenden Tracker, was das Versenden einer Email an die Personen auslöst, die an der Bearbeitung des Falls beteiligt sind. In besonderen Fällen wird auch persönlich mit den Entwicklern Kontakt aufgenommen.

Zu einer Überwachung der Fälle gehört auch das Aussortieren von veralteten Fällen. Ein Teil der 600 Fälle in dem öffentlichen Tracker sind Fehler, die einmalig unter besonderen Umständen aufgetreten sind. Da sich die Fehler nicht reproduzieren ließen, wurden sie nicht weiter bearbeitet. Manche der veralteten Fehler wurden eventuell bereits im Upstream behoben und können nachträglich geschlossen werden.

4.3.9 Abrechnung

Während der Abrechnung werden die im Kundendienst geleisteten Stunden zusammengetragen, um sie dem Kunden in Rechnung zu stellen. Der Kunde erhält eine Auflistung seiner Kundenfälle und der darin aufgewendeten Zeiten.

Eine Abrechnung erfolgt grundsätzlich in vier Schritten:

1. Zusammentragen der Zeiten
2. Zuordnung der Zeiten zu den Kundenfällen
3. Erstellen der Rechnung
4. Übertrag bereits abgerechneter Zeiten

Im folgenden sollen die einzelnen Schritte näher betrachtet werden.

Der erste Schritt zum Erstellen einer Abrechnung ist das Zusammentragen der Zeiten in den Kundenfällen. Die Mitarbeiter der Intevation GmbH dokumentieren den Zeitaufwand als Texteintrag direkt in den Änderungsnotizen der Kundenfälle im internen Tracker. Um an die Zeiten zu gelangen, müssen diese aus den Änderungsnotizen extrahiert werden. Partner arbeiten indirekt an den Kundenfällen, indem sie die technischen Ursachen der Kundenfälle im öffentlichen Tracker beheben. Sie übersenden eine Auflistung der aufgewendeten Zeiten per Email.

Um nun herauszufinden, wieviel Zeit insgesamt bei der Bearbeitung eines Kundenfalls aufgewendet wurde, müssen die Zeiten der Partner aus den öffentlichen Fällen den Kundenfällen zugeordnet werden. Dabei gilt zu beachten, dass ein Fall im öffentlichen Tracker mit mehreren Kundenfällen in Verbindung stehen kann. Dies bedeutet, dass die in dem öffentlichen Fall aufgewendete Zeit anteilig über mehrere Kunden abgerechnet werden könnte.

Nach der Zuordnung steht nun eine ausführliche Liste der Zeitaufwände zur Verfügung, die die Grundlage für eine Rechnung ist. In der Regel wird eine solche Liste immer pro Wartungsvertrag erstellt.

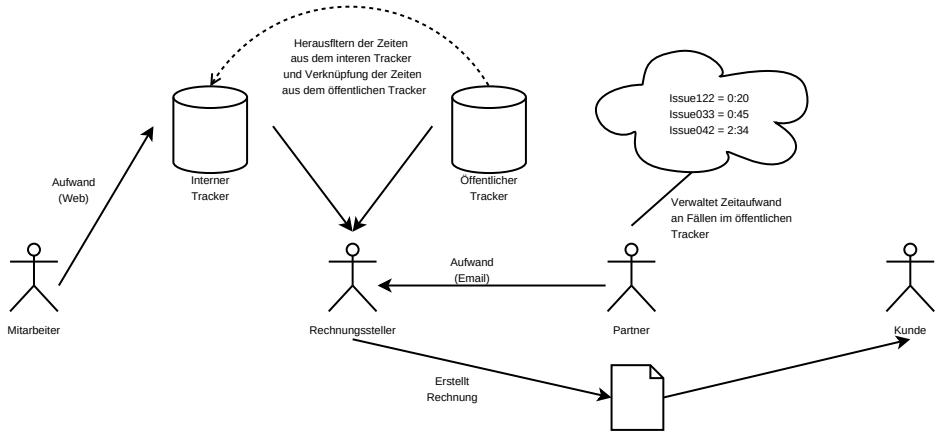

Abbildung 4.8: Beteiligte und Kommunikation bei der Fallabrechnung

Nachdem die Rechnung an den Kunden versendet wurde, müssen die bereits abgerechneten Zeiten in dem internen Tracker übertragen werden, um eine mehrfache Abrechnung der Zeiten zu vermeiden.

Abbildung 4.8 zeigt schematisch den Zusammenhang der beteiligten Personen und Infrastruktur.

4.4 Betrachtung der Kommunikationswege

Bei der Betrachtung der Geschäftsprozesse in Abschnitt 4.3 wurden bereits einige der Kommunikationswege zwischen den beteiligen Personen deutlich. In diesem Abschnitt sollen diese Kommunikationswege noch einmal genauer betrachtet werden. Hierzu wird sich auf die Kommunikation zwischen Kunde, Mitarbeiter und Entwickler beschränkt. Abbildung 4.9 zeigt neben den drei Akteuren auch die beiden Tracker Systeme. Die Pfeile kennzeichnen die Kommunikationswege. Es wird zwischen der Primärkommunikation (durchgezogene Linie) und der Sekundärkommunikation (gestrichelte Linie) unterschieden.

Der größte Teil der Kommunikation zwischen dem Mitarbeiter und dem Partner wird über den öffentlichen Tracker geführt. Vor allem während der Analyse und der Behebung und Wiederherstellung findet durch die Dokumentation des Bearbeitungsfortschritts im Tracker ein Informationsaustausch statt. Kommuniziert werden:

- Reproduktionsanleitungen
- Testergebnisse
- Patches

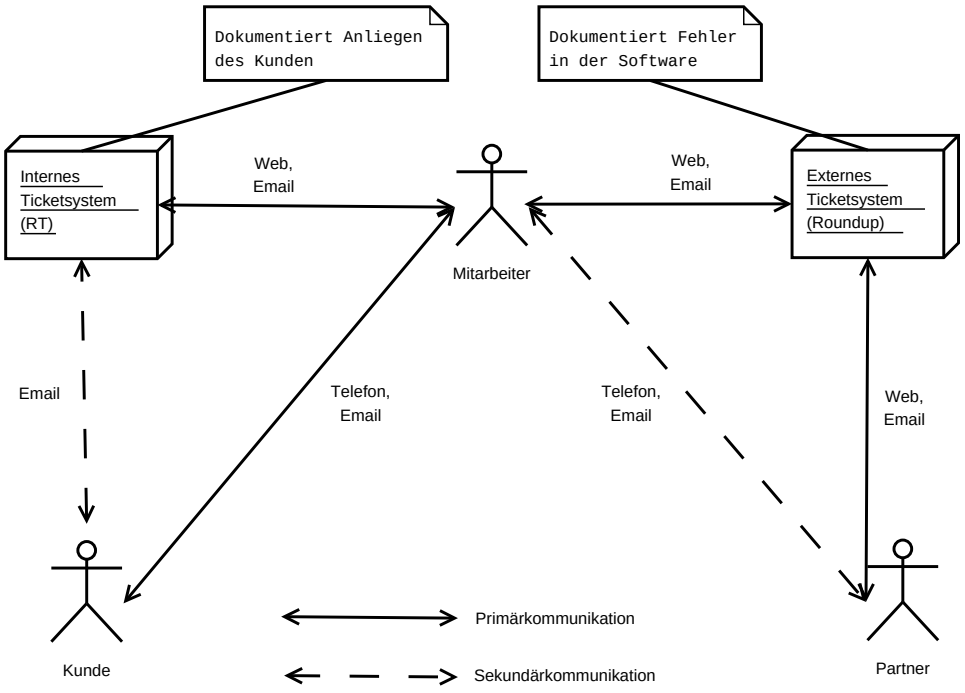

Abbildung 4.9: Kommunikationswege bei der Fallbearbeitung

- Implementationsfragen
- Verständnisfragen
- Verweise auf andere Tracker

Die Sekundärkommunikation wird genutzt, wenn eine zeitnahe Reaktion gewünscht ist, oder die auszutauschenden Informationen nicht für die Öffentlichkeit bestimmt sind. Ein Beispiel hierfür ist die Anfrage für eine Aufwandsabschätzung (siehe Abschnitt 4.3.5) zur Implementation einer bestimmten Funktion.

Mitarbeiter und Kunde stehen primär in einem direkten Kontakt miteinander. Der Kunde wendet sich an den Mitarbeiter, um per Email oder Telefon neue Fälle zu melden, sich über den Bearbeitungsfortschritt zu informieren oder um neue Erkenntnisse mitzuteilen, die dem Mitarbeiter bei der Analyse eines Vorfalls helfen können. Mit dem internen Tracker kann der Kunde nur per Email in Kontakt treten. Die Funktionalität der Email Schnittstelle des internen Tracker ist aber beschränkt auf das Anlegen neuer Fälle und dem Hinzufügen neuer Informationen zu einem bestehenden Fall. Für den Mitarbeiter dient der interne Tracker in erster Linie als Falldatenbank, in der alle Anliegen des Kunden, Gesprächsnotizen und die Dokumentation der Fallbearbeitung gespeichert sind.

Wichtig ist die Feststellung, dass der Kunde und der Partner nicht direkt in Kontakt zueinander stehen. Sämtliche Kommunikation zwischen dem Entwickler und dem Kunden läuft über den Mitarbeiter, der die Funktion einer Schnittstelle übernimmt. Beispiel für eine solche Kommunikation kann wieder die Aufwandsabschätzung herangezogen werden.

4.5 Feststellungen

Die in Abschnitt 4.3 genannten Prozesse lassen sich in nach (Probst, 2004) in die drei Phasen *Fehleraufnahme*, *Fehleranalyse* und *Fehlerbehebung* unterteilen. Durchgehend in allen Phasen findet eine Dokumentation der Bearbeitung statt. Abbildung 4.10 zeigt dies anschaulich.

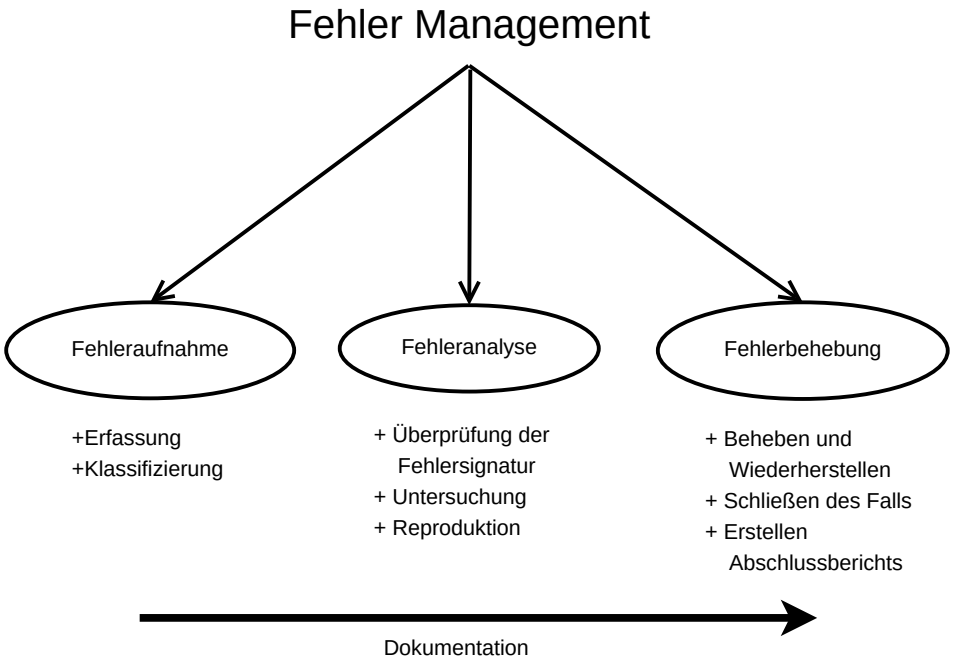

Abbildung 4.10: Phasen des Fehler-Managements

Insgesamt betrachtet fällt auf, dass die Prozesse durch ein hohes Maß an Kommunikation geprägt sind, die über verschiedene Kanäle stattfindet. Die beteiligten Personen stehen allerdings nicht in einem direktem Kontakt (siehe 4.4). Der Mitarbeiter übernimmt die Rolle einer Schnittstelle, um eine Kommunikation zwischen Partner und Kunde zu ermöglichen.

Es lässt sich feststellen, dass die Prozesse und Kommunikation sowohl geschlossen als auch offen ablaufen. Eine klare Trennung zwischen den Prozessen ist aber nicht immer klar möglich.

Tendenziell finden sich geschlossene Prozesse immer dort, wo mit sensiblen Daten des Kunden gearbeitet wird und der Datenschutz gewährt sein muss. Die Dokumentation des Bearbeitungsfortschritts sowie eine Diskussion findet hier über den nicht öffentlichen Teil der Infrastruktur statt. Dazu zählen Prozesse wie die Fehleraufnahme, Beauftragung von Arbeiten und der Fallabschluss.

Offene Prozesse, wie das Beheben und Wiederherstellen, und Teile der Analyse laufen transparent für alle Beteiligten ab. Diskussionen und die Dokumentation finden in dem öffentlichen Tracker oder sonstigen öffentlichen Medien statt. Die Ergebnisse der Arbeit sind für jedermann zugänglich.

Entwickler aus den Upstream Projekten sind in die Prozesse nicht direkt involviert. Die Ziele in diesen Projekten sind grundverschieden und die Entwickler arbeiten dort aus einer völlig anderen Motivation. Während für die Intevation Lösungen für den Kunden im Mittelpunkt stehen, ist für den Upstream vor allen Dingen die Weiterentwicklung der Software wichtig. Ein Fehler im Kolabserver oder Client, die sich durch fehlerhafte Komponenten aus dem Upstream ergeben, sind dort vielleicht nur von geringer Wichtigkeit. Es gibt also keine Sicherheit, dass sich jemand aus dem Upstream mit dem Fehler befassen wird. Eine Zusammenarbeit mit den Entwicklern im Upstream verläuft daher nur auf freiwilliger Basis.

Kapitel 5

Diagnose

Dieses Kapitel nennt potentielle Probleme während der Bearbeitung eines Kundenfalls. Sie sind das Ergebnis einer Analyse der Prozesse aus Kapitel 4. Während der Analyse wurden die Abläufe und Kommunikation vor dem Hintergrund der zugrunde liegenden Infrastruktur betrachtet und bewertet.

5.1 Einsatz verschiedener Tracker Systeme

Der Einsatz von zwei verschiedenen Tracker Systemen führt zu einer

1. größeren Gefährdung bei Angriffen,
2. erhöhten Administrationsaufwand und
3. unterschiedliche Datenhaltung und Bedienung

Durch den Einsatz zwei verschiedener Systeme erhöht sich die Anzahl von potentiellen sicherheitskritischen Fehlern in der Software, die einen Angriff auf die Infrastruktur der Intevation GmbH ermöglichen können.

Der Aufwand für die Pflege der Installation ist verdoppelt. Sicherheitsupdates, sowie allgemeine Aktualisierungen der Software müssen für zwei verschiedene Versionen eingepflegt werden. Hinzu kommt, dass die Datenhaltung zwischen den beiden Trackern nicht einheitlich ist.

Während Roundup die Dateien direkt zu dem Fall in dem Tracker speichern kann, ist dies in der Version 2 des Request Trackers nicht möglich. Dateien werden stattdessen auf einem Server abgelegt und im Tracker auf diese Dateien verwiesen. Der Mitarbeiter kann nicht direkt auf die Dateien zugreifen, sondern muss diese erst von dem Server laden. Dieser Umstand führt neben der unterschiedlichen Bedienung zu einem Mehraufwand bei der Bearbeitung eines Falls und behindert eine flüssige Arbeitsweise.

5.2 Keine direkte Kommunikation

Wie in Abschnitt 4.4 gezeigt, stehen Kunden und Partner nicht in direktem Kontakt. Sämtliche Kommunikation zwischen diesen Beiden verläuft über einen Mitarbeiter. Dies hat den Vorteil, dass so für den Kunden ein zentraler Ansprechpartner existiert, der die Organisation der Fallbearbeitung übernimmt. Für den Entwickler ist es von Vorteil, da ein Großteil der Anfragen der Kunden schon von den Mitarbeitern im Vorfeld beantwortet werden können. Durch diese Filterfunktion erhält der Entwickler weniger Anfragen, und kann sich besser auf seine eigentliche Arbeit konzentrieren. Hinzu kommt, dass z.B. Fehlerbeschreibungen eines Kunden von einem Mitarbeiter verifiziert und in einem ausführlichen Fehlerbericht aufbereitet werden.

Auf der anderen Seite bedeutet eine Filterfunktion aber auch eine Verfälschung im Informationsfluss zwischen dem Kunden und dem Entwickler. Das Prinzip dieser Verfälschung ist aus dem Spiel "Stille Post" bekannt: Eine Nachricht soll über mehrere Stationen an einen Empfänger übertragen werden, wobei sie immer mit jeder Station weiter verfälscht wird. Künstliche "Hierarchien" zwischen Entwickler und Kunden, wie sie über eine solche Schnittstelle aufgebaut werden, wirken als Kommunikationsbremse. Da der Kunde als Auftraggeber direkten Einfluss auf die Weiterentwicklung der Software hat, ist Softwareentwicklung in einem solchen Umfeld problematisch.

Das Weiterleiten der Informationen erzeugt einen großen Mehraufwand, der in Verbindung mit der möglichen Verfälschung der übertragenen Information ein großer Nachteil einer solchen Schnittstelle ist. Hier sollte eine direktere Kommunikation ermöglicht werden.

5.3 Abbrechung hoher Zeitaufwand

Der Zeitaufwand für die Erstellung einer Abbrechung wird von der Geschäftsführung der Intevation GmbH, je nach Umfang mit einen halben bis ganzen Manntag pro Kunde angegeben. Der hohe Zeitaufwand ergibt sich vor allem aus dem manuellen Zusammentragen der Zeiten aus den verschiedenen Quellen, sowie dem umständlichen Zuordnen der Zeiten aus den öffentlichen Trackern zu den Kundenfällen (siehe Abschnitt 4.3.9).

Betrachtet man diesen Arbeitsaufwand einmal vor der Annahme, dass eine Abrechnung vierteljährlich stattfindet, wird schnell deutlich, dass sich erhebliche Ausfälle von effektiver Arbeitszeit ergeben. Diese Problematik verschärft sich mit jedem zusätzlichem Kunden.

Der Prozess der Abrechnung ist ineffektiv und könnte durch den Einsatz einer einheitlichen Zeiterfassung stark verbessert werden.

5.4 Fehlende Verbindung zwischen den Trackersystemen

Zwischen den Fällen im internen und öffentlichen Tracker besteht oft ein direkter Zusammenhang. Viele der von dem Kunden gemeldeten Fehlern finden ihre Ursache in einen Softwarefehler, der im öffentlichen Tracker dokumentiert ist. Gleiches gilt für Erweiterungswünsche. Abbildung 5.1 zeigt die Abhängigkeit zwischen den Fällen im öffentlichen und internen Tracker. Es besteht also eine direkte Abhängigkeit zwischen dem Fortschritt der

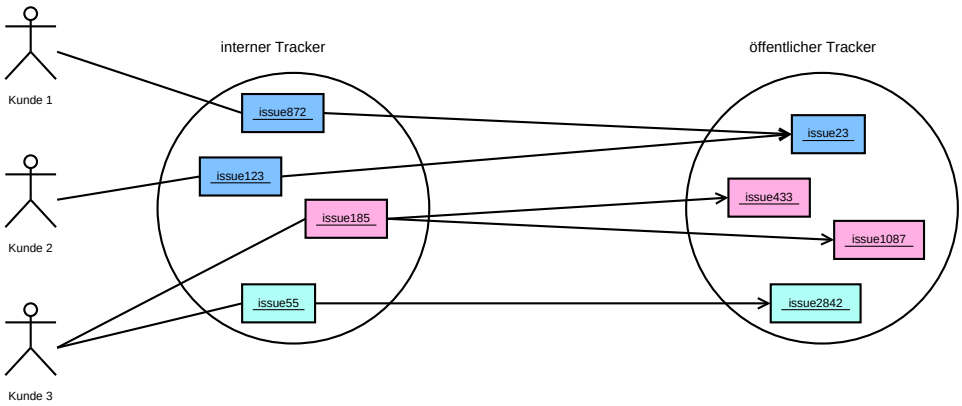

Abbildung 5.1: Abhängigkeit zwischen internen und öffentlichen Fällen

Behebung einer Fehlerursache im öffentlichen Tracker und dem Beseitigen der Auswirkungen bei dem Kunden. Die Abhängigkeit zwischen den Fällen (intern:öffentlich) kann der Art 1:1, N:1 oder 1:N sein. Derzeit wird auf diese Abhängigkeiten nur als Textnachricht innerhalb der Änderungsnotizen eines Falls (Trouble-Ticket) hingewiesen. Dies ist aus vielerlei Gründen unzureichend: Als Hinweise innerhalb der Textnachachrichten sind die Abhängigkeiten für den Benutzer nicht direkt sichtbar. Um zu überprüfen, ob eine Abhängigkeit besteht, müssen unter Umständen alle Änderungsnotizen betrachtet werden. Informationen zu den Fällen im öffentlichen Tracker, die bei der Bearbeitung eines Kundenfalls von Interesse sein können müssen durch den Mitarbeiter jedes mal neu zusammengetragen werden, da es nicht möglich ist auf diese Informationen direkt zuzugreifen. Zu den interessanten Informationen einen Falls im öffentlichen Tracker gehören unter anderen:

- sein aktueller Bearbeitungszustand (Neu, in Bearbeitung, ...),
- der Zeitpunkt der letzten Aktivität in dem Fall oder
- die Priorität mit der dieser Fall bearbeitet wird.

Die Überprüfung der Abhängigkeiten zwischen den Fällen, sowie das Zusammentragen der Informationen ist in einem hohen Maß automatisierbar und erzeugt momentan einen unnötigen Mehraufwand. Die manuelle Synchronisation der Informationen zwischen den Trackern ist fehleranfällig und birgt die große Gefahr von Inkonsistenzen der Informationen über einen Fall. Diese ergeben sich durch das manuelle Übertragen von Informationen eines öffentlichen Falls in einen Kundenfall.

Bei einer hohen Anzahl von Fällen und langer Bearbeitungsdauer besteht zudem das große Risiko, dass Abhängigkeiten zwischen den Fällen in Vergessenheit geraten. Eine Beseitigung einer Fehlerursache kann dann unter Umständen von niemanden mehr mit einem Kundenfall in Verbindung gebracht werden. Der Kundenfall bliebe dann im schlimmsten Fall so lange unbearbeitet, bis jemandem die Behebung der Fehlerursache auffällt.

5.5 Mangelnde Transparenz für den Kunden und Entwickler

Die Vorgänge während der Bearbeitung eines Falls sind für den Kunden und Entwickler nicht ausreichend transparent. Dies ergibt sich aus dem fehlenden Zugriff auf die Fälle im internen Tracker (siehe Abschnitt 4.4).

Für den Kunden gibt es keine Möglichkeit sich einen Gesamtüberblick über seine Fälle im internen Tracker zu verschaffen oder den Bearbeitungsfortschritt einzelner Fälle zu verfolgen. Der Partner kann nicht auf die Problembeschreibung des Kunden im internen Tracker zugreifen, obwohl diese eventuell nützliche Informationen zur Behebung der Fehlerursache enthalten kann. Dem Partner fehlt eine potentiell wichtige Informationsquelle. Auch er hat keine Möglichkeit sich über den Bearbeitungsfortschritt eines Falls im internen Tracker zu informieren.

Sowohl Kunde als auch Partner können nur indirekt über die Schnittstelle des Mitarbeiters an diese Informationen gelangen. Der Prozess der Bearbeitung eines Kundenfalls sollte so offen wie möglich, aber aus Gründen des Datenschutzes so geschlossen wie nötig gestaltet werden.

5.6 Fehlende Statistiken

Wenn es darum geht Aussagen über die Qualität von Prozessen, und letztendlich des gesamten Kundendienstes, zu machen, dann spielen Statistiken eine wichtige Rolle. Statistiken fassen Kennzahlen übersichtlich zusammen, lassen Prognosen für zukünftige Entwicklungen zu oder decken Schwachstellen in den Abläufen auf.

So kann eine steigende Anzahl von offenen Kundenfällen darauf hindeuten, dass nicht genügend Kapazitäten vorhanden sind, um die Fälle zu bearbeiten. Eine Zeiterfassung könnte Aufschluss darüber geben, in welcher

Phase der Bearbeitung die meiste Zeit verbraucht wird. Das gezielte Ergreifen von Maßnahmen, um die Bearbeitung des Falls zu beschleunigen und die Gesamtdurchlaufzeit eines Falls zu verringern, wird so erst möglich. Statistiken geben gesicherte Aussagen über die Abläufe in den Prozessen und bilden die Grundlage für deren Steuerung und Optimierung.

Die Intevation GmbH hat derzeit noch keine automatisierten Werkzeuge zur Erfassung von solchen Kennzahlen aus dem Kundendienst, die zur Erzeugung von Statistiken genutzt werden können.

5.7 Fehlende Email-Verschlüsselung/Signierung

Die Kundenfälle und sämtliche Informationen, die von dem Kunden zu dem Fall mitgeliefert wurden, unterliegen dem Datenschutz und werden in dem internen Tracker gespeichert, wo sie nicht öffentlich zugänglich sind. Der Datenschutz ist an dieser Stelle somit gewährt. Wie in Abschnitt 4.4 gezeigt wird, kann der Kunde durch eine Email an den internen Tracker neue Fälle eröffnen oder bestehende Fälle erweitern. Email gilt grundsätzlich als ein unsicherer Kommunikationskanal, da die Daten im Klartext übertragen werden und somit von jedem mitgelesen werden können, der Zugriff auf den Übertragungsweg hat. Daher sind die Daten, die von einem Kunden per Email an den Tracker gesendet werden, nicht ausreichend geschützt.

Neben der Verschlüsselung ist auch die Authentifizierung der Daten durch eine Signierung wünschenswert. Über eine Authentifizierung kann effektiv gesichert werden, dass

- die Daten unverändert den Empfänger erreichen,
- der Absender der Daten tatsächlich der ist, als der er sich ausgibt und
- nur befugte Personen Daten an den Tracker senden können.

Die fehlende Unterstützung zur Behandlung verschlüsselter Emails ist insbesondere dann ein Problem, wenn interne Richtlinien bei dem Kunden eine gesicherte Kommunikation vorschreiben. In diesem Fall kann der Kunde die Informationen nicht direkt an den Tracker senden, sondern verschickt sie verschlüsselt an den Mitarbeiter, der die Daten entschlüsselt und in den Tracker einträgt. Dies führt zu einem unnötigen Mehraufwand auf Seiten der Intevation GmbH.

Kapitel 6

Lösungen

Dieses Kapitel nennt Lösungsansätze zu den diagnostizierten Problemen aus Kapitel 5 . Zunächst wird die Vereinheitlichung der Trackersysteme beschrieben, die die Grundlage für die weiteren Lösungen ist. Danach werden die Lösungen konzeptionell vorgestellt und die erwartete Verbesserung genannt. Die Lösungen werden in Muss- und Wunschkriterien unterschieden. Eine detailierte Beschreibung des Designs bzw. der Umsetzung findet sich in folgenden Kapiteln.

6.1 Vereinheitlichung der eingesetzten Tracker

Das Ziel der Vereinheitlichung ist die Menge der verschiedenen Tracker Systeme, die im Kundendienst eingesetzt werden, von zwei auf ein System zu reduzieren und so die in Abschnitt 5.1 diagnostizierten Probleme zu lösen. Hierzu zählt:

- Verringerung des Risikos von Angriffen
- Senkung des Administrationsaufwand
- Vereinheitlichung der Datenhaltung und Bedienung

Die Vereinheitlichung der eingesetzten Tracker Systeme spielt eine besondere Rolle, da alle künftigen Erweiterungen und Anpassungen auf der Grundlage des eingesetzten Trackersystems umgesetzt werden. Aus diesem Grund wird sie getrennt von den übrigen Musskriterien in Abschnitt 6.2 betrachtet.

6.1.1 Sondierung verfügbarer Tracker Systeme

Zu Beginn der Auswahl eines neuen Tracker Systems wurden eine Reihe von verfügbaren Systemen untersucht, die als Freie Software verfügbar sind. Ziel dieser Untersuchung sollte die Identifikation eines Tracker Systems sein, dass in seiner Grundfunktionalität am Besten für den Einsatz im Kolab Kundendienst geeignet ist. Es wurden folgende Systeme betrachtet:

Bugzilla Bugzilla[1] ist der vielleicht bekannteste Vertreter der Bugtracker und in der Sprache Perl implementiert. Er wird in vielen Projekten eingesetzt. Darunter unter anderen das KDE Projekt, Mozilla, die Linux Kernel und einer Reihe von Distributionen. Bugzilla befindet sich in einer ständigen Weiterentwicklung. Eine erweiterte Liste von Installationen des Bugzilla Bugtracker findet sich auf der Webseite.

Debian Bug-Tracker (debbugs) Das Debian-Projekt[2] verfügt über einen eigenen Bugtracker[3] der verwendet wird um Debian-spezifische Fehler zu verwalten. Dies können Fehler sein, die sich durch die Anpassung der Pakete auf die Debian Distribution ergeben, wie auch grundsätzliche Fehler in der Funktionalität der Software. *debugs* ist wie auch der zuvor genannte Bugzilla in Perl implementiert.

Trac Trac[4] ist ein erweitertes BTS, welches über den Umfang eines normalen BTS hinaus auch Möglichkeiten zur Organisation eines Projekts bietet. Es unterstützt neben dem Anlegen von Meilensteinen und Veröffentlichungs-Plänen auch die Verwendung eines Wiki, um Dokumentation vornehmen zu können. Zur Implementation wurde die Sprache Python verwendet.

Roundup Ein weiterer Vertreter, eines in Python implementieren BTS ist Roundup[5]. Roundup zeichnet sich vor allem durch die vielen Freiheiten aus, die es zur Anpassung bietet. In 3 wird ausführlich auf Roundup eingegangen.

Mantis Mantis[6] ist ein in PHP implementiertes BTS. Aufgrund der großen Popularität von PHP erfreut sich Mantis einer ständigen Weiterentwicklung und verfügt über ein reichhaltiges Angebot von Erweiterungen.

OTRS Mit dem Open Ticket Request System[7] (OTRS) lassen sich Probleme und Anfragen per Email, Fax, Telefon oder Interface bearbeiten. Das System ist in Perl implementiert und wurde im Jahre 2001 das erste Mal veröffentlicht. Anfang des Jahres 2007 war OTRS nach (Users, 2007) 35000 mal installiert. Darunter viele große Firmen und Organisationen wie die *NASA*, *ESA*, *Siemens*, *General Motors* und andere. OTRS kann auch in sehr großen Umgebungen eingesetzt werden, in denen pro Tag mehrere tausend Anfragen in das System eingetragen

[1] http://www.bugzilla.org
[2] http://www.debian.org
[3] http://bugs.debian.org
[4] http://trac.edgewall.org/
[5] http://roundup.sourceforge.net
[6] http://www.mantisbt.org/
[7] http://www.otrs.org

werden können. Die Unterstützung für eine Verschlüsselung des Email-Verkehrs durch *GnuPG*[8]. oder *S/MIME*[9] ist einer der Funktionen, die OTRS von allen anderen untersuchten System in dieser Arbeit unterscheidet.

Request Tracker Der Request Tracker[10] (RT) steht seit der ersten Veröffentlichung im Jahre 1996 in der Version 3 zur Verfügung. RT ist ebenso wie *OTRS* in *Perl* implementiert, und zur Veröffentlichung der Version 3 nahezu komplett neu geschrieben worden. RT bietet seinen Benutzer sowohl eine Web-Schnittstelle als auch eine Email-Schnittstelle. RT wird nach (BPS, 2007) unter anderen von Firmen wie der *NASA*, *Merrill Lynch*, dem *Perl*-Projekt, eingesetzt.

In Anhang A findet sich eine ausführliche Auflistung der untersuchten Tracker.

6.1.2 Auswahl des Roundup Issue Tracker

Als Ergebnis der Untersuchung wurde der Roundup Issue Tracker (siehe Kapitel 3) als künftige Grundlage eines IT-Systems im Kolab Kundendienst ausgewählt. Die Entscheidung für die Auswahl des Roundup Servers begründet sich durch die folgenden Punkte:

1. Roundup ist bereits im Einsatz und es wurden gute Erfahrungen mit dem System gemacht. So entfällt die Umgewöhnung auf ein anderes System.

2. Roundup ist im Gegensatz zu den anderen Trackern gleichermaßen für den Einsatz als Support-Tracker sowie als Bug-Tracker geeignet. Das System bietet im Vergleich mit den anderen Systemen zwar weniger Funktionalität, jedoch lässt es sich aufgrund seiner sehr guten Erweiterbarkeit gut auf die speziellen Gegebenheiten im Kolab Kundendienst anpassen und erweitern.

3. Roundup ist in Python implementiert. Diese Sprache wird von vielen der Mitarbeiter der Intevation GmbH bevorzugt wird. Dies stellt sicher, dass Anpassungen und Erweiterungen auch von den Mitarbeitern gepflegt werden können.

[8] http://www.gnupg.org
[9] http://www.ietf.org/html.charters/smime-charter.html
[10] http://www.bestpractical.com/rt/

6.2 Musskriterien

6.2.1 Zeiterfassung

Der Roundup Server wird um die Funktionalität einer Zeiterfassung erweitert. Diese bietet den Partnern und Mitarbeitern die Möglichkeit den Zeitaufwand, während der Bearbeitung eines Falls, direkt in dem Fall im Tracker zu erfassen, an dem Sie gerade arbeiten. Die Zeiten stehen für eine Abbrechung direkt zur Verfügung und müssen nicht mehr mühsam zusammen getragen und zugeordnet werden (siehe Abschnitt 5.3). Folgende Funktionalität wird von einer Zeiterfassung erwartet:

- Zeiten müssen fallbezogen erfasst werden.
- Die erfassten Zeiten sollen unter Berücksichtigung verschiedener Abrechungsraten bewertet werden können.
- Ein Fall soll anteilig über mehrere Kunden (Rechnungen) abgerechnet werden können.
- Es muss verschiedene Übersichten zu der Beziehung zwischen
  - Kunde $\Longleftrightarrow$ Rechnung
  - Rechnung $\Longleftrightarrow$Fall
  - Rechnung $\Longrightarrow$ Zeiten (abrechenbar, bereits abgerechnet)
  - Fall $\Longrightarrow$ Zeit (Rechnung, Bewertung)

  geben.
- Die Zeiterfassung muss durch ein Rechtesystem geschützt werden, welches gewährleistet, dass nur privilegierte Benutzer Zugriff auf die Zeiterfassung haben.

Die Einrichtung einer Zeiterfassung führt zu einer direkten Zeitersparnis während der Erstellung einer Abrechnung. Weiter bietet die Zeiterfassung eine potentielle Quelle wertvoller Informationen für eine Statistik, und somit zur Bewertung und Steuerung von Abläufen.

6.2.2 Verbindung verschiedener Trackern (Remote Events)

Um die Beziehungen und Abhängigkeiten (siehe Abschnitt 5.4) zwischen den Fällen besser darstellen zu können, soll eine Verbindung zwischen Fällen in verschiedenen Tracker Systemen hergestellt werden können. Ein typischer Anwendungsfall ist die Verbindung von Kundenfällen (Fehlerauswirkung) im internen Tracker mit den Fällen im öffentlichen Tracker (Fehlerursachen).

Über eine solche Verbindung soll es möglich werden, auf Informationen eines Falls in einem entfernten Tracker lesend zuzugreifen und diesen auf

Änderungen zu überwachen. Ein solcher Fall in einem entfernten Tracker wird im folgenden als Remote Event bezeichnet.

Ein Beispiel: Der Mitarbeiter arbeitet an einem Kundenfall im internen Tracker. Dieser Fall steht mit einem Fall im öffentlichen Tracker (Remote Event) in Verbindung. Der Mitarbeiter erhält nun im Kundenfall die Möglichkeit auf wichtige Information des Remote Event wie den Status oder den letzten Bearbeitungszeitpunkt zuzugreifen. Abbildung 6.1 visualisiert das Beispiel.

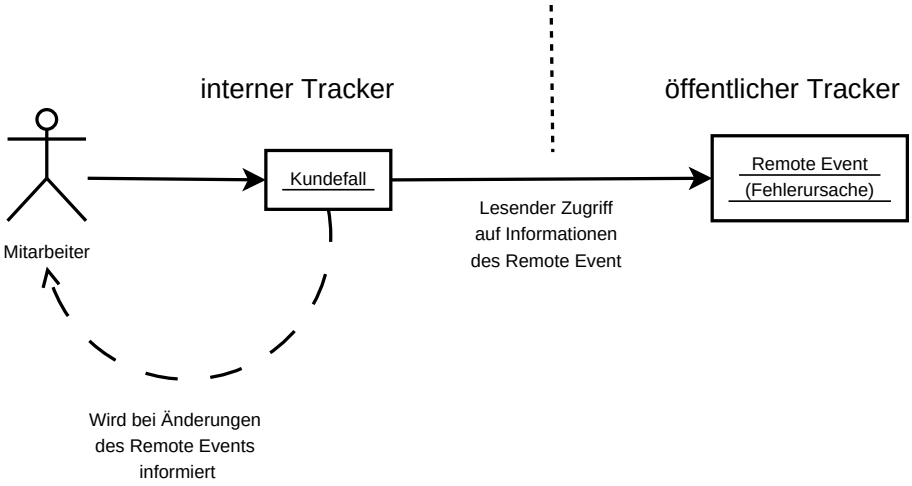

Abbildung 6.1: Zugriff auf Informationen eines *Remote Event*

Es ist somit nicht mehr nötig zusätzlich auf den entfernten Fall zu explizit öffnen, um an wichtige Informationen zu gelangen. Die Informationen propagieren sich automatisch in den anderen Fall. Diese Automatisierung führt zu einer Zeitersparnis und fördert die Konsistenz der Informationen in den Fällen.

Neben dem Zugriff auf den Fall spielt auch die Überwachung seines Bearbeitungsfortschritt eine wichtige Rolle. Sobald sich dieser ändert, wird eine Benachrichtigung verschickt. Auf diese Weise kann sich der Mitarbeiter z.B auf die Arbeit an Fällen im internen Tracker konzentrieren und wird automatisch benachrichtigt, wenn es Veränderungen in einem Remote Event gibt. Es ist für den Mitarbeiter nicht mehr nötig den entfernten Fall regelmäßig auf Aktualisierungen zu überprüfen.

Folgende Punkte müssen bei der Verbindung von Fällen in verschiedenen Trackern berücksichtigt werden:

- Die Überwachung der Remote Events muß automatisierbar sein und in regelmäßigen Abständen erfolgen.
- Das Überwachungs-Intervall muß vom Benutzer konfiguriert werden

können.

- Das System sollte nicht auf die Überwachung von Trackersystemen beschränkt werden, sondern um andere Informationsquellen erweiterbar sein (Mailinglisten, Webserver).
- Bei einer Änderung in dem entfernten Tracker muß eine Benachrichtigung erfolgen.

Eine detailierte Betrachtung des Designs und der Umsetzung der Abfrage und Überwachung der Remote Events ist Gegenstand des Kapitel 9.

6.2.3 Benutzergruppen und Sichten

Die Kommunikation während der Bearbeitung eines Falls soll direkter gestaltet werden, um so den Informationsfluss zu beschleunigen. Weiter soll die Transparenz der Abläufe bei der Fallbearbeitung für Kunden und Partner erhöht werden. Dazu wird allen in die Bearbeitung eines Falles involvierten Personen Zugriff auf die Fälle im internen Tracker gegeben. Da in dem Track-

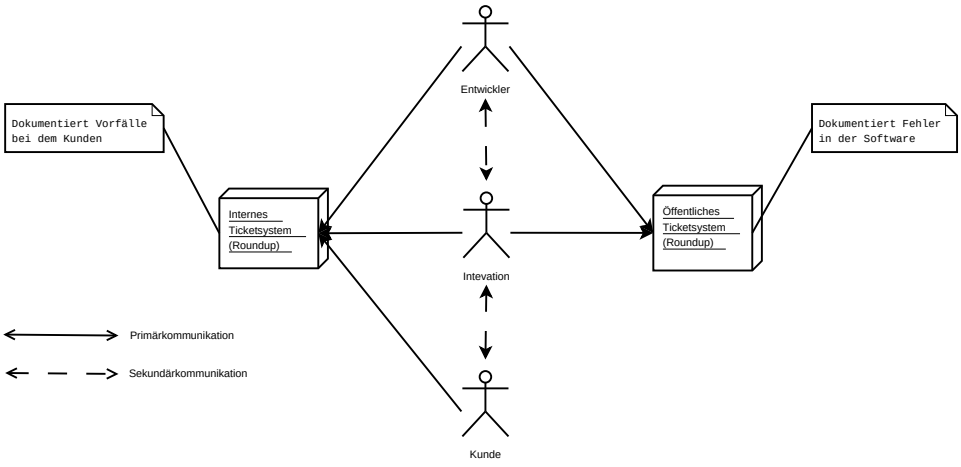

Abbildung 6.2: Gemeinsamer Zugriff auf den internen Tracker

er Fälle verschiedener Kunden gespeichert sind, die dem Datenschutz unterliegen, muß reglementiert werden können, wer in welchem Umfang Zugriff auf diese Daten erhält. Um diese Forderung zu erfüllen, wird der Roundup Server um das Konzept der Benutzergruppen und verschiedener Sichten erweitert. Die Forderung eines kontrollierten Zugriffs der beteiligten Personen umfasst dabei:

- Kunden dürfen nur ihre eigenen Fälle sehen.
- Partner dürfen nur die Fälle sehen in die sie involviert sind.

- Sichten legen fest welche Daten eines Falls für den Betrachter sichtbar sind.

Abbildung 6.2 visualisiert die veränderten Kommunikationswege nach der Einführung von Benutzergruppen und Sichten. Kunden, Partner und Mitarbeiter haben nun gemeinsam Zugriff auf den internen Tracker. Die beiden Tracker werden zur Grundlage für die Primärkommunikation zwischen allen beteiligten Personen. Kunden und Partner können jetzt über den Tracker, ohne den Mitarbeiter als Schnittstelle, in Kontakt treten und den Fortschritt der Bearbeitung verfolgen. Diese Veränderungen wirken sich positiv auf die Probleme der mangelnden Transparenz und der gebremsten Kommunikation aus (vgl. Abschnitt 5.5 und 5.2).

Eine Beschreibung des Designs und der Umsetzung der Benutzergruppen und Sichten in Roundup finden sich in Kapitel 7.

6.3 Wunschkriterien

6.3.1 Arbeitspakete

Arbeitspakete bilden eine logische Gruppierung von Fällen innerhalb eines Trackers. Ein Fall kann in mehreren Gruppierungen vorkommen. Benutzern bietet ein Arbeitspaket die Möglichkeit einen Überblick über zusammenhängende Aufgaben zu behalten und gemeinsam über deren Ablauf zu diskutieren. Ein Arbeitspaket kann beispielsweise für folgende Zwecke eingesetzt werden:

- Erstellung von Rechnungen Kunden
- Releaseplan
- Meilenstein
- allgemeine Aufgabenlisten

Abbildung 6.3 zeigt das Konzept der Arbeitspakete grafisch. Arbeitspakete helfen bei der Organisation und Strukturierung von zu erledigenden Aufgaben. Sie stehen in enger Verbindung mit der Zeiterfassung. Die Kombination dieser beiden Erweiterungen macht es möglich den zeitlichen Aufwand mehrerer Aufgaben zusammen auf einen Blick anzuzeigen. Dies ist besonders für die Erstellung von Rechnungen hilfreich. In diesem Fall enthält ein Arbeitspaket all die Fälle, die dem Kunden in Rechnung gestellte werden sollen.

6.3.2 Unterstützung einer Emailverschlüsselung

Die Erweiterung der Emailschnittstelle des Roundup Servers um kryptographische Funktionen wäre wünschenswert. Dazu gehört:

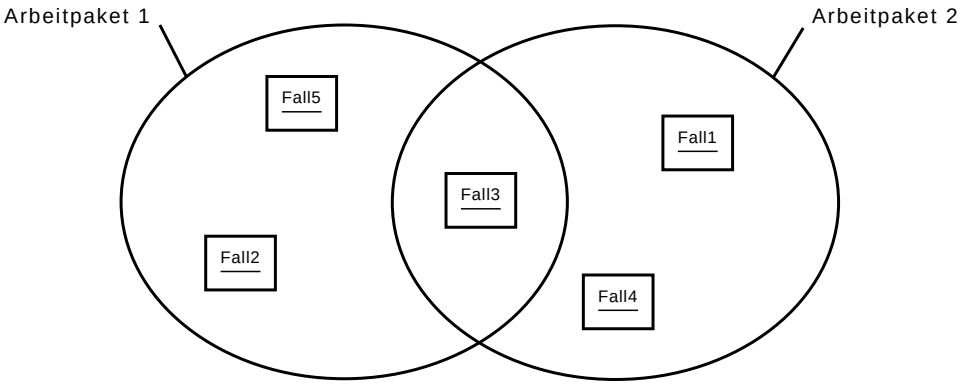

Abbildung 6.3: Logische Gruppierung der Fälle durch Arbeitspakete

- Unterstützung von eingehender und ausgehender verschlüsselter Email durch den Tracker
- Unterstützung der Erstellung und Überprüfung von signierten Emails

Unter `http://wiki.python.org/moin/GnuPrivacyGuard` findet sich eine Auflistung verfügbarer Bibliotheken, die die Funktionalität von GPG[11] unter Python zur Verfügung stellen. Diese könnten die Grundlage für eine solche Erweiterung sein. Eine kurze Untersuchung dieser Bibliotheken zeigte allerdings, dass ein Großteil der Bibliotheken schon seit langer Zeit nicht mehr gepflegt werden oder aus anderen Gründen nicht anwendbar waren.

Die Erweiterung des Roundup Servers um Möglichkeiten der Emailverschlüsselung ist ein bereits geäußerter Wunsch der Roundup Benutzer. Der Hauptentwickler des Roundup Servers verweist als eine mögliche Grundlage dieser Erweiterung auf `http://www.amk.ca/files/python/GPG.txt`.

6.3.3 Statistiken

Der Roundup Server soll wichtige Kennzahlen aus den Falldaten generieren können, um so eine umfassende Statistik erstellen zu können. Diese Statistik umfasst dabei unter anderen:

- Die Anzahl der zugewiesenen Fälle pro Bearbeiter
- Die Anzahl der Fälle in den verschiedenen Fehlerkategorien
- Die Anzahl der Fälle in den verschiedenen Fallzuständen
- Die Bearbeitungsdauer von Fällen

[11] http://www.gnupg.org

Eine Beobachtung dieser Kennzahlen über einen längeren Zeitraum hinweg ermöglicht es Schwachstellen aufzudecken und Prognosen über zukünftige Entwicklungen zu geben. Die Intevation GmbH kann auf Grundlage dieser Statistik Maßnahmen zur Steuerung der Prozesse im Kundendienst einleiten.

Das Wiki[12] des Roundup Projekts enthält bereits einige Ansätze zur Erstellung einer Statistik. Diese könnten als Basis für eine Weiterentwicklung genutzt werden.

6.4 Abgrenzungskriterien

Ziel dieser Lösungen ist es nicht die bisherige Arbeitsweise der Mitarbeiter grundsätzlich zu verändern, sondern sie an einzelnen Stellen besser zu unterstützen. Die Erweiterungen sollen so allgemein gehalten sein, dass sie auch in einem anderen Rahmen als den Kundendienst für den Kolab Server eingesetzt werden können. Auch wenn in manchen Punkten eine tiefergehende Spezialisierung auf besondere Abläufe im Kolab Kundendienst möglich wäre, soll diese bewusst nicht vorgenommen werden.

[12] http://www.mechanicalcat.net/tech/roundup/wiki/FrontPage

Teil III

Design

Kapitel 7

Benutzergruppen und Sichten

In der Analyse-Phase zeigte sich, dass an der Bearbeitung eines Falls mehrere Personen bzw. Personengruppen beteiligt sein können. In diesem Kapitel wird nun auf Grundlage von Benutzergruppen und Sichten eine Lösung erarbeitet, die die Zusammenarbeit mehrerer Personen an einem Fall ermöglichen und besser unterstützen soll.

7.1 Benutzergruppen

7.1.1 Konzept

Benutzergruppen sind eine logische Gruppierung von mehreren Benutzern innerhalb des Trackersystems. Durch die Analyse zeigt sich, dass drei Personengruppen auftreten, die an der Bearbeitung eines Falls beteiligt sein können:

- Kunden
- Partner
- Interne (Mitarbeiter)

Benutzergruppen sollen grundsätzlich sicherstellen, dass nur die Personen Zugriff auf einen Fall erhalten, die an dem Fall auch tatsächlich arbeiten.

Eine feiner graduierte Regulierung des Zugriffs auf die Daten eines Falls erfolgt durch die Zuweisung von Rollen an die Benutzergruppen. Alle Mitglieder der Benutzergruppe erben die in der Rolle festgelegten Zugriffsrechte auf die Daten der Datenbank.

Um sicherzustellen, dass auf die Fälle nur von den Personen zugegriffen werden kann, die auch an der Bearbeitung beteiligt sind, muss eine Zuordnung zwischen dem Fall und den Personengruppen möglich sein. Abbildung 7.1 zeigt diese Zuordnung zwischen den Benutzergruppen und einem Fall. Das

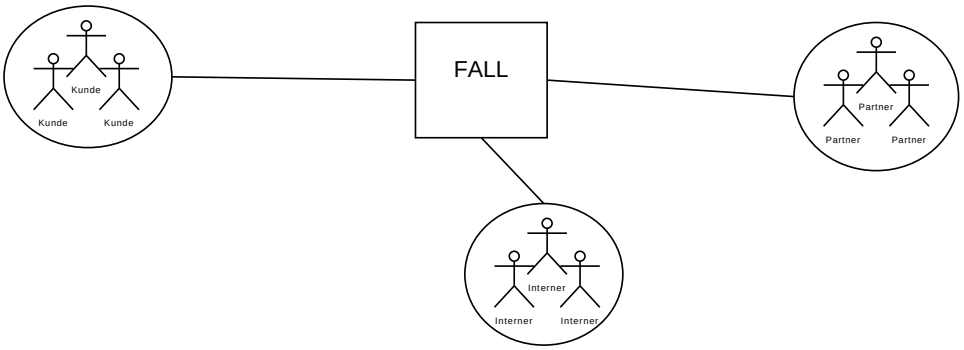

Abbildung 7.1: Zuordnung eines Falls zu verschiednen Benutzergruppen

Rechtesystem des Roundup muss dann um einen Mechanismus erweitert werden, der überprüft, ob die zugreifende Person Mitglied in einer der Gruppen ist, die dem Fall zugewiesen wurden. Roundup bietet die Möglichkeit eigene Funktionen in das Rechtesystem einzubinden, über die sich eine solche Überprüfung realisieren lässt.

7.1.2 Anforderungen

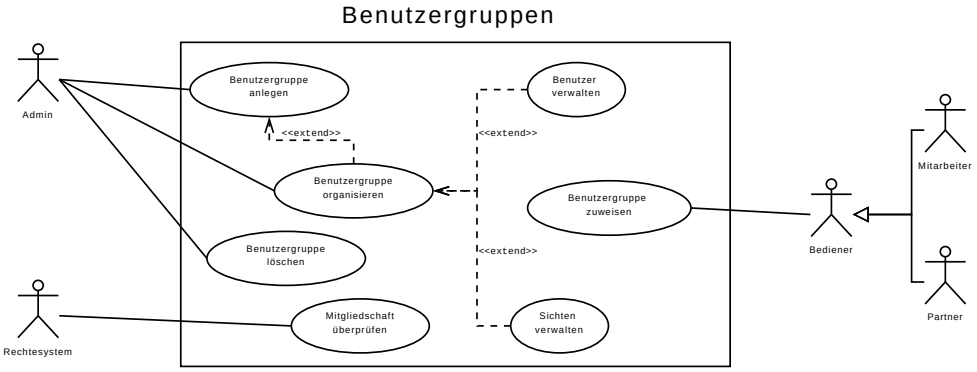

Abbildung 7.2: Anwendungsfalldiagramm der Benutzergruppen

Benutzergruppe anlegen Der Administrator legt eine neue Benutzergruppe in dem System an. Dieser Fall kann durch die Organisation der Benutzergruppe erweitert werden.

Benutzergruppe organisieren Zur Organisation einer Benutzergruppe gehören die folgenden Tätigkeiten:

- Benutzer verwalten

- Sichten verwalten

Der Administrator kann einer bestehenden Benutzergruppe Mitglieder hinzufügen oder entfernen. Desweiteren muss festgelegt werden, welche Sichten die Mitglieder der Gruppe sehen dürfen.

Benutzergruppe entfernen Der Administrator entfernt eine Benutzergruppe aus dem System.

Benutzergruppe zuweisen Bei dem Anlegen und Bearbeiten eines Falls kann durch den Benutzer festgelegt werden, welchen Gruppen dieser Fall zuzuordnen ist.

Mitgliedschaft prüfen Bei einem Aufruf eines Falls muss das Rechtesystem prüfen, ob die aufrufende Person Mitglied in einer der Gruppen ist, die dem Fall zugewiesen wurden.

7.1.3 Erweiterung des Roundup Datenbank Schema

Abbildung 7.3 zeigt die Erweiterung des Datenbankschemas, die vorgenommen wurde, um die Benutzergruppen in die Datenbank des Roundup zu intigrieren. Die Tabellen *usergroup* und *view* wurden nachträglich hinzugefügt.

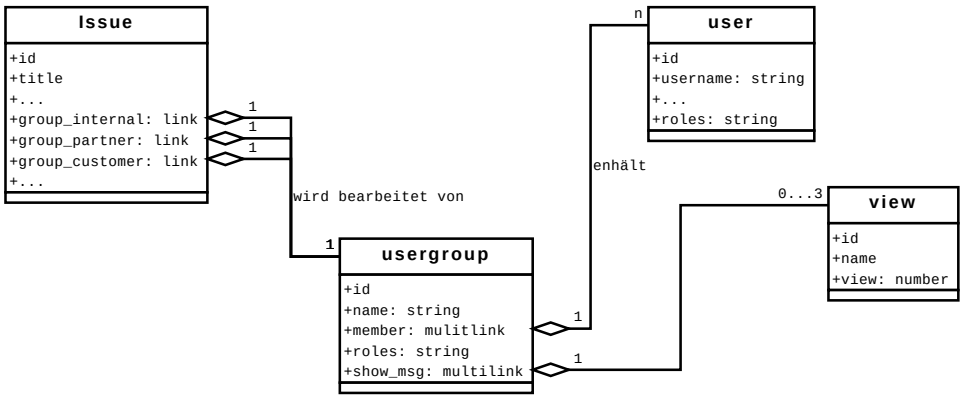

Abbildung 7.3: Erweitertes Datenbank Schema für die Benutzergruppen

Die Tabelle usergroup dient der Speicherung von Benutzergruppen. Jede Benutzergruppe erhält einen Namen, der als Zeichenkette abgespeichert wird.

Rollen werden durch den Roundup als kommaseparierte Zeichenkette von Rollennamen gespeichert. Nach dem Vorbild der Tabelle user sollen auch Benutzergruppen über Rollen verfügen. Dazu erhält die Tabelle usergroup das Feld roles.

Dies hat den Vorteil, dass nun eine ganze Gruppe von Personen unkompliziert mit Befugnissen für den Zugriff auf einen Fall ausgerüstet werden

können. Rollen, die einer Benutzergruppe zugewiesen wurden, vererben sich auf die Mitglieder der Gruppe. Es ist nicht nötig jedem Benutzer die Rollen einzeln zuzuweisen. Um die Vererbung der Rollen zu ermöglichen, ist das Modifizieren der Roundup Bibliothek nötig. Die Modifizierung wird in Abschnitt 10.5.1 erläutert.

Die Mitglieder der Gruppe werden über das Multilink-Feld *member* in der Tabelle usergroup festgelegt. Sie verlinkt mit den Benutzern in der Tabelle user. Jeder Gruppe sind über das Feld *show_msg* bis zu maximal drei Sichten zugeteilt. Über dieses Feld werden zwei Dinge festgelegt:

1. Aus welchen Sichten der Benutzer Änderungsnotizen eines Falls sehen darf.

2. Für welche Sichten der Benutzer Änderungsnotizen eines Falls erstellen darf.

Der Grund für diese gesonderte Behandlung der Änderungsnotizen wird in Abschnitt 7.2.1.2 behandelt.

Um festzulegen welche Benutzergruppen grundsätzlich Zugriff auf einen Fall erhalten sollen, wird die Tabelle *issue* um drei Felder erweitert, die den Fall mit den Benutzergruppen verbinden.

7.2 Sichten

7.2.1 Konzept

Im Gegensatz zu den Benutzergruppen, die *grundsätzlich* den Zugriff auf die Fälle organisieren, dienen Sichten einer feiner graduierten Regulierung der Sichtbarkeit von Informationen zu einem Fall. Eine Sicht bestimmt eine Teilmenge der Informationen zu einem Fall, auf die die Benutzer bzw. Benutzergruppen zugreifen können. Abbildung 7.4 veranschaulicht das Konzept der Teilmengen von Informationen. Jede der drei in die Fallbearbeitung involvierten Personengruppen soll eine eigene Sicht auf die Fälle haben. Somit ergeben sich folgende drei Sichten:

Interne-Sicht Die Sicht für die Mitarbeiter der Intevation (Interne) ist die umfassendste der drei verfügbaren Sichten. In dieser Sicht sind alle Informationen zu einem Fall sichtbar und es können alle Daten eines Falls bearbeitet werden.

Partner-Sicht Partner erhalten in der Partner-Sicht nur Zugriff auf einen Teil der Informationen eines Falls. Die interne Kommunikation zwischen den Mitarbeitern der Intevation GmbH bleibt den Partner so z.B. verborgen.

Kunden-Sicht Die eingeschränkteste Sicht auf die Falldaten haben die Kunden. In der Kundensicht fehlen eine Vielzahl der internen Informationen zu einem Fall. Dazu gehören z.B. der Zeitaufwand der Mitarbeiter, oder interne Kommunikation zwischen den Partnern und Mitarbeitern.

Manager-Sicht Die Manager Sicht ist keine eigenständige Sicht, sondern erweitert die Interne-Sicht um Informationen über den Zeitaufwand bei der Fallbearbeitung.

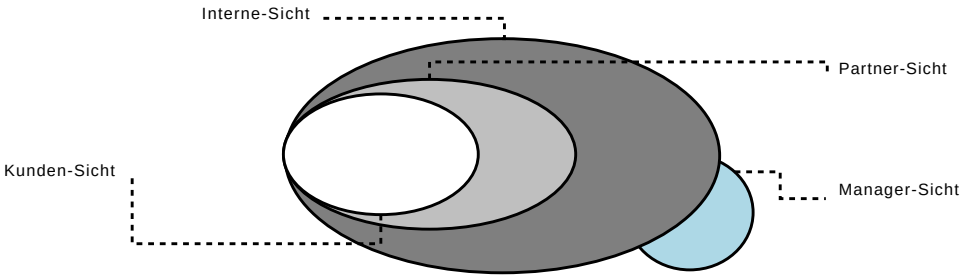

Abbildung 7.4: Darstellung der Sichten als Mengen

In einem rollenbasierten Rechtesystem, wie es in Roundup zu finden ist, werden die Zugriffsrechte auf die Attribute eines Falls durch Rollen an seine Benutzer übertragen. Neue Rollen können von dem Benutzer definiert werden, um so angepasste Zugriffsrechte für die Benutzer zu definieren. Tatsächlich wird für einen Benutzer die Sicht auf die Daten in der Datenbank durch die Rolle bestimmt, die er besitzt. Der Begriff der *Sicht* ist somit eine andere Beschreibung einer *Rolle*, die aber in dem gegebenen Fall besser den Kern des Ziels beschreibt und daher verwendet wird. Das Rollensystem in Roundup bietet sich daher zur Realisierung der Sichten an.

7.2.1.1 Realisation durch Rollen

Roundup kennt grundsätzlich drei Arten von Zugriffen auf die gespeicherten Daten in der Datenbank:

- Create (Erstellen)
- Edit (Bearbeiten)
- View (Ansehen)

Für jede Tabelle in der Datenbank können Zugriffsrechte erstellt werden. Die Zugriffsrechte definieren welche Daten erstellt, bearbeitet oder angesehen werden dürfen und werden einer Rolle zugewiesen. Ein Benutzer erlangt dann, über die ihm zugeteilten Rollen, die entsprechenden Befugnisse

für den Zugriff auf die Daten in der Datenbank. Die Überprüfung der Zugriffsrechte liegt in der Verantwortung des Rechtesystems in Roundup. Wie auch schon bei den Benutzergruppen lässt sich diese Überprüfung durch selbstgeschriebene Funktionen realisieren, die in das bestehende Rechtesystem eingebunden werden können.

Die genauen Befugnisse für den Zugriff auf die Informationen eines Falls sind im Anhang B aufgeführt.

7.2.1.2 Sichten auf Änderungsnotizen

Das Übertragen des Konzeptes der Sichten auf Änderungsnotizen und Dateianhänge erfordert eine besondere Behandlung. Änderungsnotizen sind in den Fall über ein Multilink Attribut eingebunden. Über das Rechtesystem kann der Zugriff auf einzelne Attribute einer Klasse reguliert werden. Eine Regulierung des Zugriffs auf dieses Multilink Attribut führt daher dazu, dass entweder alle oder keine Änderungsnotizen angezeigt werden. Eine Unterscheidung zwischen den Nachrichten ist auf diesen Weg nicht möglich. Daraus folgt, dass nicht festgelegt werden kann, *welche* Nachrichten innerhalb einer Sicht angezeigt werden sollen.

Um eine solche Unterscheidung möglich zu machen wird sowohl die Nachricht, als auch der Dateianhang um ein weiteres Attribut ergänzt, über dass sich die Nachricht mit einer bestimmten Sicht verknüpfen lässt.

Während für die Nachrichten festgelegt wird in welcher Sicht sie erscheinen sollen, ist für die Benutzer über ihre Benutzergruppe festgelegt auf welche Sichten sie Zugriff haben (siehe Abschnitt 7.1.3).

Durch die Überprüfung dieser Zuordnung lässt sich feststellen, ob ein Benutzer berechtigt ist auf die Änderungsnotiz zuzugreifen. Die Implementation dieser Überprüfung wird in Abschnitt 10.5 erläutert.

7.2.2 Anforderungen

Zuweisen einer Sicht zur Änderungsnotiz Benutzer müssen bei dem Erstellen einer Nachricht angeben in welchen Sichten diese Nachricht zu sehen sein soll.

Versenden von Benachrichtigungsmails Wird eine neue Änderungsnotiz eingetragen, so versendet das Mailsystem eine Nachricht an alle in der Nosy-List eingetragenen Personen. Das das Rechtesystem hier gewährleisten, dass nur die Personen eine Benachrichtigungsmail bekommen, die aufgrund ihrer Befugnisse (Benutzergruppe) die Änderungsnotiz sehen dürfen. Dazu muss das Rechtesystem erweitert werden.

Überprüfen des Zugriffs auf die Falldaten Bei jedem Zugriff auf einen Fall muss das Rechtesystem prüfen welche der Attribute und welche

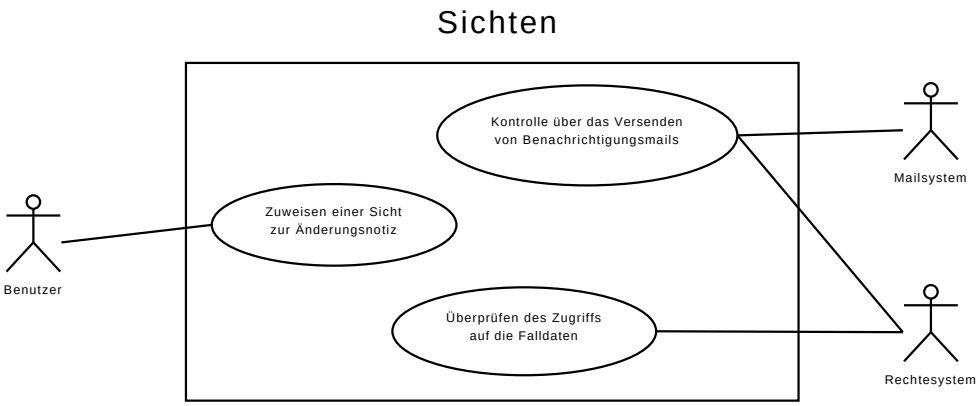

Abbildung 7.5: Anwendungsfalldiagramm der Sichten

Abbildung 7.5 zeigt die Anwendungsfälle der Sichten innerhalb eines Diagramms.

Änderungsnotizen für die zugreifende Person sichtbar sein sollen.

7.2.3 Erweiterung des Roundup Datenbank Schema

Die Erweiterung des Datenbankschemas, die vorgenommen wurde um die Sichten in die Datenbank des Roundup zu intigrieren zeigt Abbildung 7.6. Da die Sichten in großen Teilen vollständig von dem Rollenmechanismus in

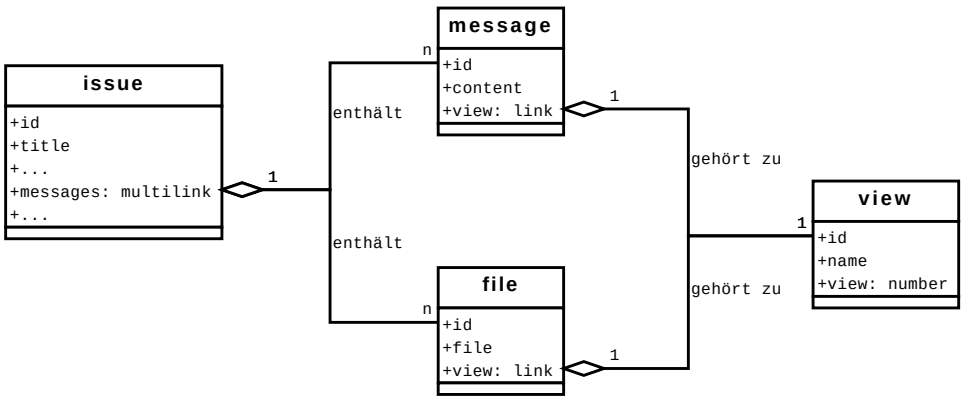

Abbildung 7.6: Erweitertes Datenbank Schema für die Sichten

Roundup abgedeckt werden können, fällt die Erweiterung des Datenbank Schemas nur klein aus. Neu hinzugekommen ist hier nur die Tabelle *view*, deren Einträge eine bestimmte Sicht darstellen. Über ein Linkattribut verweisen Änderungsnotizen und Dateien auf einen Eintrag in der Tabelle view.

Auf diese Weise wird festgelegt zu welcher Sicht die Änderungsnotiz bzw. die Datei gehört.

Kapitel 8

Zeiterfassung

8.1 Anforderungen

An die Zeiterfassung werden die in Abbildung 8.1 grafisch dargestellten Anforderungen gestellt. Partner und Mitarbeiter haben grundsätzlich Zugriff auf die Zeiterfassung. Sie treten dabei in den Rollen *Partner*, *Internal* und *Manager* auf.

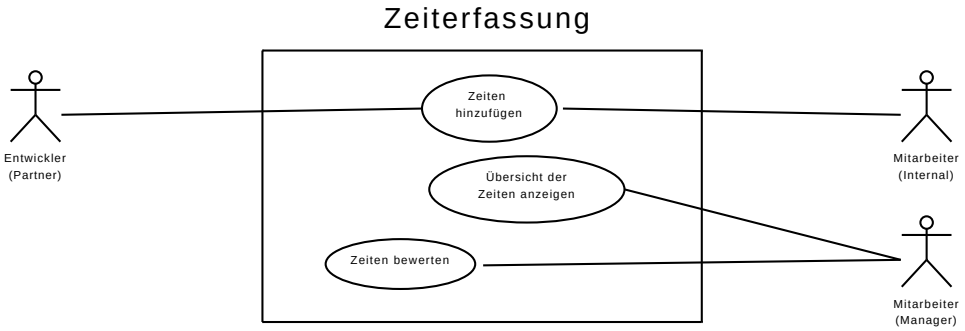

Abbildung 8.1: Anwendungsfalldiagramm der Zeiterfassung

Zeiten hinzufügen Zeiten können einem Fall hinzugefügt werden, um den zeitlichen Aufwand bei der Bearbeitung zu dokumentieren.

Zeiten bewerten Zeiten können bewertet werden. Die Funktionalität wird insbesondere im Hinblick auf die Erstellung einer Rechnung wichtig. Über eine Bewertung kann festgelegt werden, zu welchem Prozentsatz die Zeit in den Rechnungen auftauchen sollen. So könnte die Bearbeitung eines dringenden Kundenfalls am Wochenende z.B. mit mehr als 100% abgerechnet werden.

Übersicht der anzeigen Der Benutzer erhält eine Übersicht über die bereits festgehaltenen Zeiten in einem Fall. Diese gibt Übersicht darüber

wer, wann, wieviel Zeit eingetragen hat und zu welchem Prozentsatz diese Zeit abgerechnet wurde.

8.2 Konzept

Um eine einfache Zuordnung zwischen Fällen und der bei ihrer Bearbeitung aufgewendeten Zeit zu ermöglichen, werden die Zeiten in dem Tracker gespeichert.

Die bisherigen Ansätze zur Zeiterfassung, die im Wiki des Roundup Projekts verfügbar sind, verfolgen die Strategie Zeiten direkt in den Fällen (Trouble Ticket) zu speichern. Als Verbesserung dieses Ansatzes sollen die Zeiten nicht mehr in dem Fall gespeichert werden, sondern in den Änderungsnotizen eines Falls. Abbildung 8.2 zeigt das Konzept der Speicherung von Zeiten. Eine

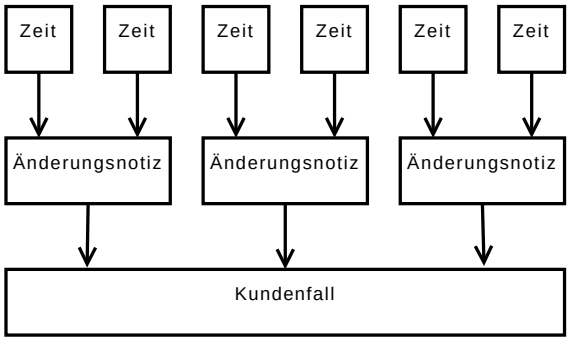

Abbildung 8.2: Speicherung der Zeiten in den Änderungsnotizen

Zuordnung zu dem Fall ist dann indirekt über die Änderungsnotizen eines Falls möglich. Eine derartige Speicherung bietet zwei wesentliche Vorteile:

1. Die Zeitaufwände in einem Fall werden differenzierbar, da über die Änderungsnotiz direkt ersichtlich ist für welche konkrete Aufgabe die Zeit aufgewendet wurde.

2. Durch die Speicherung der Zeiten an den Änderungsnotizen wird die Zeiterfassung unabhängig von einem Kundenfall. Die Zeiterfassung kann so ohne großen Aufwand auf alle Klassen angewendet werden, die ebenfalls Änderungsnotizen einbinden. Hierzu zählt z.B. der Remote Event.

Zur Speicherung von Zeiträumen bietet der Roundup Server als Teil seiner Zeit- und Datums-Funktionen den Datentyp *Interval*. Für diesen Datentyp stehen umfangreiche Funktionen zur Verfügung, über die verschiedene Berechnungen mit den Zeiträumen durchgeführt werden können.

Jeder Zeiteintrag soll für eine spätere Abrechnung bewertet werden können. Dies bedeutet, dass die von einem Benutzer eingetragene Zeit als Brutto-

Zeit zu verstehen ist. Über eine Bewertung lässt sich festlegen wieviel Prozent der Brutto-Zeit in einer Abrechnung auftauchen soll (Nettozeit).

8.3 Erweiterung des Roundup Datenbank Schemas

Zur Umsetzung der Zeiterfassung wird das Datenbank Schema um die Tabellen timelog und timelograte erweitert. Abbildung 8.4 zeigt die relevanten Teile der Erweiterung des Datenbank Schemas in der Datei `<TRACKERINSTANZ>/schema.py`. Die Tabelle timelog dient der Speicherung von Zeiten und ver-

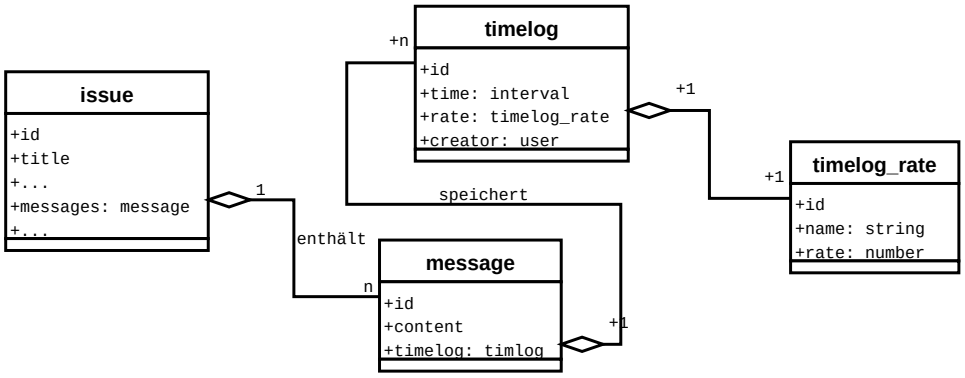

Abbildung 8.3: Erweitertes Datenbank Schema der Zeiterfassung

fügt hierfür über den bereits erwähnten Datentyp Interval. Eine Zuordnung zwischen den Zeiten und der Benutzer, die diese Zeiten eingetragen haben, entsteht beim Anlegen eines neuen Zeiteintrag automatisch über das Feld *creator*.

Zur Bewertung bindet die Tabelle timelog einen Eintrag aus der Tabelle *timelograte* ein. Diese enthält eine Liste prozentualer Angaben, die als Grundlage für eine Berechnung der Nettozeit herangezogen werden.

Die Zuordnung zwischen den Zeiten und dem eigentlichen Fall erfolgt über die Verkettung der Tabellen issue und message durch Multilink Felder.

8.4 Klassendiagramm

Die Implementation der Zeiterfassung erfolgt als Python-Modul, welches als Extension in den Roundup eingebunden wird. Python-Module kapseln ähnlich wie Java-Packages eine komplexe Funktionalität als eine Einheit. Auf diese Weise lässt sich die Zeiterfassung auch in anderen Erweiterungen nutzen. Die Fassaden-Klasse TimelogFacade stellt die Funktionalität des Moduls über eine definierte Schnittstelle bereit. In der Schnittstelle sind Methoden deklariert, die zur Abfrage der Zeitaufwände in einem Fall

genutzt werden. Die Anfragen werden dabei an die Klasse Timelog delegiert. Abbildung 8.4 zeigt ein vereinfachtes Klassendiagramm der Zeiterfassung. Sämtliche Bezüge zu anderen Erweiterungen wie z.B. den Arbeitspaketen wurden aus Gründen der Übersichtlichkeit entfernt. Die Klasse Timelog übernimmt in dieser vereinfachten Darstellung lediglich die Erzeugung der Fälle (Issue). Die Klassen Issue und Message dienen primär der logischen

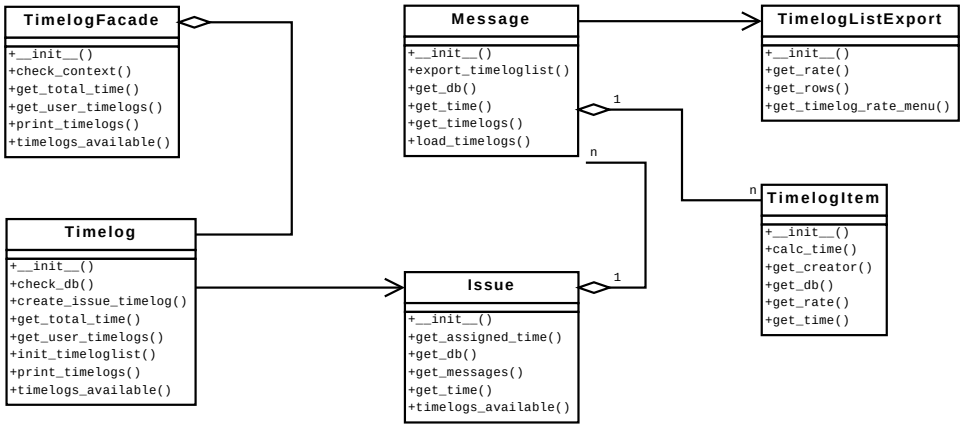

Abbildung 8.4: Klassendiagramm der Zeiterfassung

Strukturierung. Die Funktionalität der beiden Klassen beschränkt sich neben dem Zugriff auf die Daten des Roundup Server auf das Aufsummieren von Zeiten. Die Klasse Message berechnet die Summe der in ihr eingetragenen Zeitaufwände, die wiederum von der Klasse Issue zu einem Gesamtaufwand aufaddiert werden.

Die Klasse TimelogListExport bietet die Möglichkeit eine Auflistung der Zeiten zu exportieren. Derzeit ist nur der Export als HTML für die Weboberfläche vorgesehen. Sind weitere Exportformate gewünscht, ließe sich die Klasse zu einem Strategie-Muster nach (Gamma u. a., 1995) ausweiten.

Die Klasse TimelogItem repräsentiert einen Zeiteintrag. Die Klasse implementiert Methoden zum Laden und Berechnen der Nettozeit.

8.5 Ablaufdiagramm Zeitberechnung

Abschließend soll der Algorithmus bei der Berechnung des Gesamtaufwands zu einem Fall beschrieben werden. Dieser wird in Abbildung 8.5 anschaulich dargestellt. Im ersten Schritt wird der Fall geladen. Dieser Fall enthält eine Reihe von Nachrichten, die jeweils über mehrere Zeiteinträge verfügen können (vgl. 8.4). Die Berechnung erfolgt in zwei ineinander verschachtelten Schleifen. In der inneren Schleife werden die Zeiten einer einzelnen Nachricht aufsummiert. Die äussere Schleife addiert diese Zwischensummen zu dem

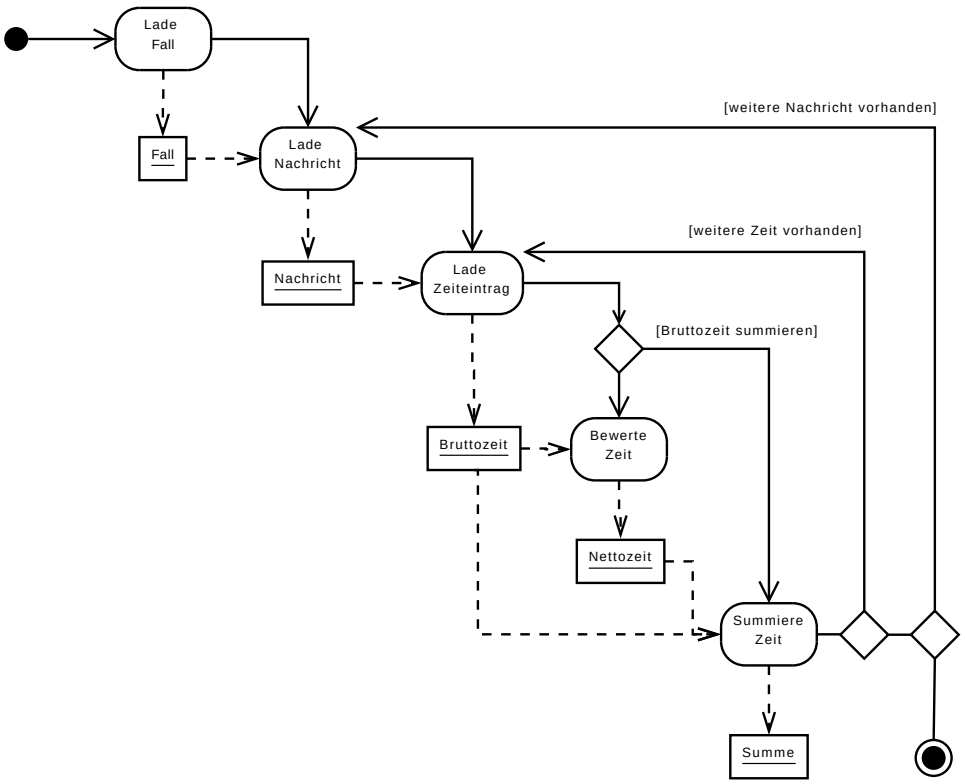

Abbildung 8.5: Ablaufdiagramm Zeiterfassung

Gesamtaufwand eines Falls zusammen. Nicht eingetragen in diesem Diagramm ist die Bewertung der Zeit. Sie findet direkt nach dem Laden der Zeiten statt.

Kapitel 9

Remote Events

Im Rahmen dieses Kapitels soll das Konzept und das Design einer Verbindung zwischen einer lokalen und entfernten Informationsquelle beschrieben werden.

Bislang wurde immer von der Verbindung zweier Tracker Systeme bzw. der darin enthaltenen Fälle gesprochen. Typischerweise ist dies die Verbindung von Fällen zwischen dem internen Tracker und öffentlichen Tracker. Allerdings muss diese Verbindung nicht zwangsläufig zwischen zwei Tracker Systemen bestehen. Grundsätzlich soll ein Zugriff auf eine beliebige Informationsquelle aus einem Fall heraus ermöglicht werden. Als alternative Informationsquellen können hier Webseiten, Mailinglisten oder Versionskontrollsysteme genannt werden.

Für eine weitere Beschreibung werden folgende Begrifflichkeiten verwendet, die kurz erläutert werden:

lokale Informationsquelle Eine lokale Informationsquelle bezeichnet den Ort von dem auf eine andere Informationsquelle zugegriffen werden soll. Im Rahmen dieser Arbeit handelt es sich bei der lokalen Informationsquelle um einen Fall (lokaler Fall) in einem Tracker System (lokaler Tracker).

entfernte Informationsquelle Das Gegenstück der lokalen Informationsquelle ist die entfernte Informationsquelle. Sie bezeichnet den Ort auf den zugegriffen wird. In dieser Arbeit wurde allerdings nur der Zugriff auf einen Fall (entfernter Fall) in einem Tracker (entfernter Tracker) implementiert.

Nachdem diese beiden Begriffe geklärt wurden kann das Konzept zur Realisierung dieser Verbindung erläutert werden.

9.1 Konzept

Der Modellierung einer Verbindung zwischen einem Fall im lokalen Tracker und einer entfernten Informationsquelle liegt folgender Gedanke zu Grunde: Der Roundup Server wird um eine neue Klasse erweitert, die stellvertretend für eine entfernte Informationsquelle steht. Ein solches Stellvertreter-Objekt wird im folgenden *Remote Event* genannt. Lokale Fälle können diese Remote Events einbinden, um so eine Verbindung zwischen dem lokalen Fall und einer entfernten Informationsquelle zu modellieren.

Bei einem Aufruf eines Remote Events wird eine Verbindung mit der entfernten Informationsquelle aufgebaut und die gewünschten Informationen geladen. Abhängig von der Informationsquelle und der Art der Abfrage (HTTP, POP3, RSS) befinden sich die Informationen nach dem Laden zunächst noch in einem "Datencontainer" wie z.B. einem HTML-Dokument oder einer Email. Ein Parser übernimmt das Extrahieren der Informationen aus dem "Datencontainer". Die extrahierten Daten werden dann in dem Remote Event gespeichert und stehen dem Benutzer für eine Anzeige bereit.

Eine Überwachung der entfernten Informationsquellen kann durch eine automatisierte regelmäßige Abfrage der Remote Events erfolgen. Dabei werden die aktuellen Daten der entfernten Informationsquelle mit den zuletzt gespeicherten Daten im Remote Event verglichen. Wird ein Unterschied zwischen diesen Datensätzen festgestellt, so wird eine Benachrichtigung an die Personen verschickt, die im lokalen Fall an der Bearbeitung beteiligt sind (Nosy-List). Remote Events eigenen sich zur Modellierung der in Abschnitt 5.4 genannten N:1 Beziehung zwischen den N Kundenfällen, die alle eine technische Ursache aus dem öffentlichen Tracker haben. Abbildung 9.1 veranschaulicht noch einmal das Konzept der Remote Events grafisch am Beispiel einer Verbindung von Fällen in verschiedenen Trackern.

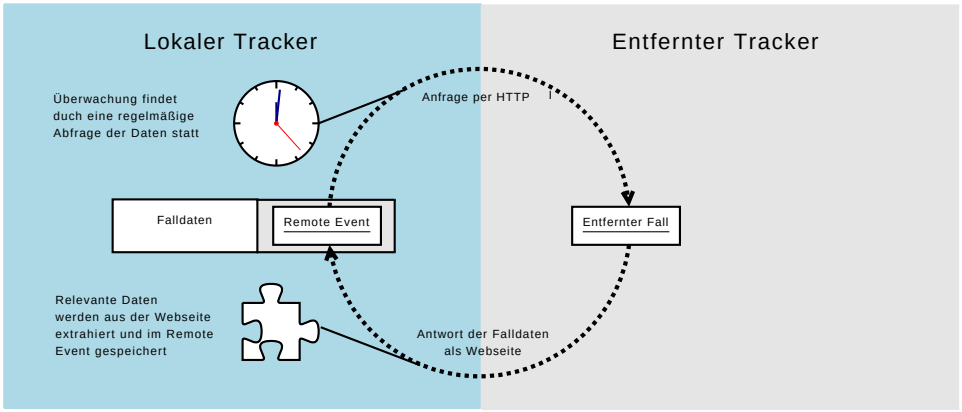

Abbildung 9.1: Konzept der Remote Events

Durch die Schaffung solcher Stellvertreter-Objekte ergibt sich als Nebeneffekt eine Lösung für ein Problem aus dem Datenschutz. Wie in der Analyse gezeigt darf der öffentliche Tracker aus Gründen des Datenschutzes keine kundenbezogenen Daten enthalten. Ein Remote Event im internen Tracker bietet die Möglichkeit kundenspezifische Informationen zu einem öffentlichen Fall im internen Tracker zu speichern. Da der interne Tracker als vertrauenswürdig gilt, können Informationen wie Aufwandsabschätzungen oder Zeitaufwände dort unter Wahrung des Datenschutzes gespeichert werden.

9.2 Anforderungen

Die funktionalen Anforderungen an die Remote Events werden in Abbildung 9.2 als Anwendungsfall Diagramm dargestellt.

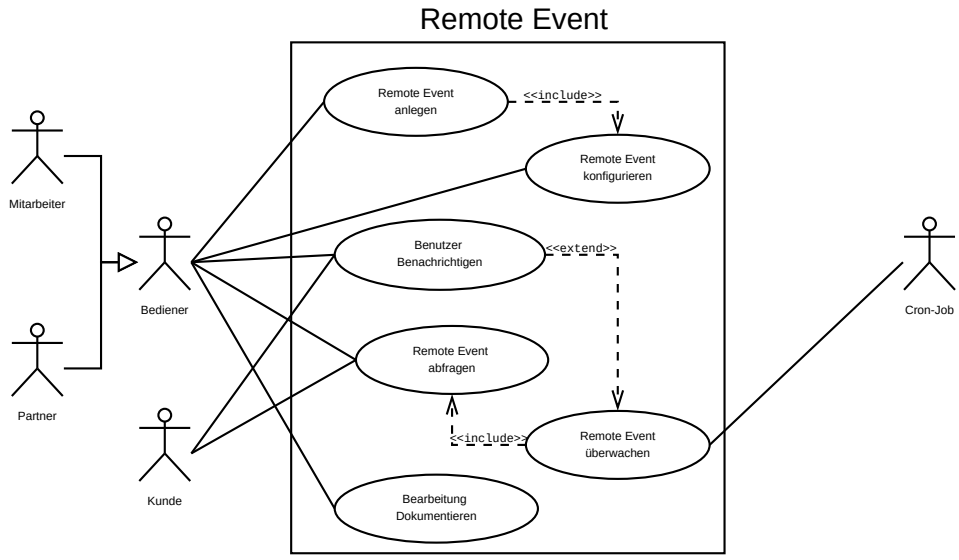

Abbildung 9.2: Anwendungsfalldiagramm der Remote Events

Remote Event anlegen Der Benutzer legt innerhalb eines Falls ein Remote Event an, um über diesen lesenden Zugriff auf eine entfernte Informationsquelle zu erlangen. Bereits bestehende Remote Events können in anderen Fällen wiederverwendet werden. So ist es möglich von mehreren lokalen Fällen auf eine entfernte Informationsquelle zu verweisen.

Remote Event konfigurieren Der Remote Event kann in seinen Eigenschaften konfiguriert werden. Hierzu gehört:

- Aktivierung/Deaktivierung einer automatischen Überwachung des Remote Event
- Abfrageintervall des Remote Event (Überwachung)
- Aktivierung/Deaktivierung von Benutzerbenachrichtigungen
- Konfiguration wie Art der entfernten Informationsquelle und wie diese zu erreichen ist

Dokumentation Der Benutzer kann Informationen zu einer entfernten Informationsquelle als Änderungsnotiz direkt an einem Remote Event speichern, um so z.B. kundenspezifische Details zu einem öffentlichen Fall im lokalen Tracker zu speichern.

Remote Event überwachen Durch einen Cron-Job werden die in der Datenbank gespeicherten Remote Events regelmäßig aufgerufen und eine Überprüfung der Daten in den entfernten Informationsquellen vorgenommen. Dabei wird geprüft, ob sich Daten seit dem letzem Aufruf geändert haben.

Benutzer benachrichtigen Wird bei der Überprüfung der Daten einer entfernten Informationsquelle festgestellt, dass sich die Daten geändert haben, wird eine Benachrichtigung verschickt.

Remote Event abfragen Bei einer Abfrage des Remote Event wird die entfernte Informationsquelle geladen und die relevanten Daten durch einen Parser extrahiert. Die extrahierten Daten werden im Remote Event gespeichert.

9.3 Erweiterung des Roundup Datenbank Schemas

Die Umsetzung der Remote Events machte die Erweiterung des Roundup Datenbank Schemas nötig. Abbildung 9.3 zeigt die relevanten Erweiterungen, die durchgeführt wurden.

Neu in dem Datenbank Schema sind die Tabellen *remote_event*, emphre_connector, *re_connector_spec* und *refresh_prio*.

Von zentraler Bedeutung ist die Tabelle remote_event, die der Speicherung der Remote Events dient. Die Tabelle enthält eine Reihe von Attributen, über die sich das Verhalten der Remote Events festlegen lässt. Hierzu zählt die Festlegung, ob und in welchen Abständen eine regelmäßige Überprüfung der Remote Events stattfinden soll, und ob bei Veränderungen der Remote Events eine Benachrichtigung verschickt wird.

Um welche Art von Remote Event (entfernte Informationsquelle) es sich handelt, wird über die Attribute re_connector und re_connector_spec bestimmt. Sie verlinken mit Einträgen aus den gleichnamigen Tabellen. Das

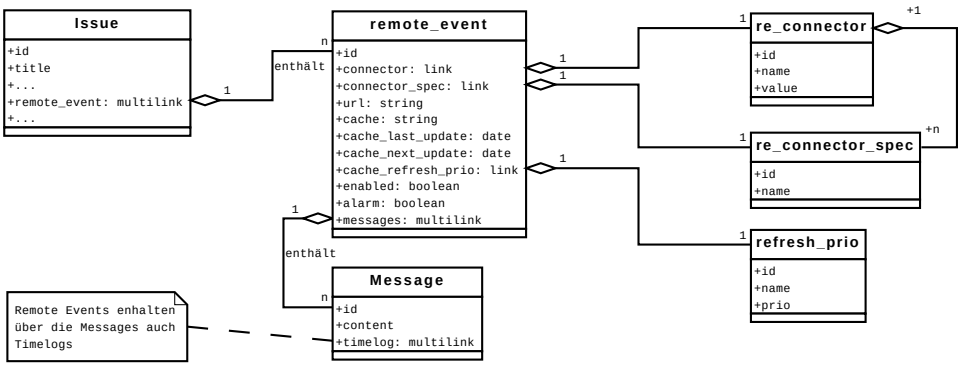

Abbildung 9.3: Erweitertes Datenbank Schema der Remote Events

Attribut re_connector bestimmt die grundsätzliche Art der entfernten Informationsquelle, also ob es sich dabei um ein Tracker-System, eine Webseite, eine Mailingliste oder Sonstiges handelt. Das Attribut re_connector_spec verfeinert die Art der Informationsquelle noch einmal, indem bestimmt wird um welche genaue Ausprägung es sich handelt. Am Beispiel eines Tracker Systems wird so festgelegt um welchen Tracker es sich genau handelt. Diese Festlegung ist wichtig, da die verschiedenen Informationsquellen die Daten in unterschiedlichen Datencontainern ausliefern. Eine solche Angabe legt also fest wie das System die Informationen aus dem Datencontainer extrahieren muss.

Um die Anzahl der Anfragen an die entfernte Informationsquelle so gering wie möglich zu halten, besitzt ein Remote Event einen Zwischenspeicher, in dem die Daten aus der letzten Abfrage gespeichert werden. Der Intervall in denen eine Aktualisierung des Zwischenspeichers erfolgt, wird über das Attribut cache_refresh_prio festgelegt.

Genauso wie normale Fälle können Remote Events Änderungsnotizen (Tabelle messages) enthalten, in denen Informationen zu einem Remote Event gespeichert werden.

Eine Verbindung der Fälle mit einem Remote Event wird über das Multilink Attribut *remote_event* der Tabelle *issue* erreicht.

9.4 Klassendiagram

Abbildung 9.4 zeigt das Klassendiagramm der Remote Event Erweiterung. Die Funktionalität wird innerhalb eines Python-Moduls gekapselt und als Roundup-Extension eingebunden. Die Erweiterung wird dem Benutzer über die Fassaden-Klasse *RemoteEventFacade* zugänglich gemacht. Die darin enthaltenen Methoden sind im Webinterface registriert und können von dort aufgerufen werden.

RemoteEventFacade Die *RemoteEventFacade* bildet die Schnittstelle zu dem Roundup Server. Sie definiert all die Methoden, die von einem Benutzer über das Webinterface aufgerufen werden können. Die Fassade implementiert keine Funktionalität, sondern delegiert die Anfragen an die internen Klassen *RemoteEventTracker* weiter.

RemoteEventTracker Die Klasse *RemoteEventTracker* dient primär der Erstellung eines Remote Events. Sie übernimmt die Erstellung von Connectoren und übergibt diese mit dem Backend bei der Erstellung eines Remote Event.

RemoteEvent Die Klasse *RemoteEvent* bietet eine einheitliche Schnittstelle um Informationen aus einer entfernten Informationsquelle abzufragen. Sie abstrahieren somit den Zugriff auf die unterschiedlichen Informationsquellen. Eine weitere Funktionalität der Klasse ist es die geladenen Daten auf Veränderung zu überprüfen und gegebenenfalls eine Benachrichtigung per Mail zu verschicken.

9.4.1 Connectoren

Connectoren realisieren die Verbindung mit der entfernten Informationsquelle. Sie bauen die Verbindung auf, lesen die Daten aus und extrahieren die gewünschten Informationen aus den Datencontainern.

Connectoren setzen sich in einer Klassenhierachie aus drei Stufen zusammen. Die Klassen in den verschiedenen Stufen übernehmen dabei unterschiedliche Aufgaben.

Stufe 1: Definition der Schnittstelle

Stufe 2: Definition der Daten

Stufe 3: Definition der Algorithmen

Die Klasse *Connector* (1. Stufe) definiert das Interface eines Connectors und implementiert Methoden, die allen abgeleiteten Klassen gemein sind. Die meisten der Methodenaufrufe werden aber an abgeleitete Klassen delegiert.

Wie bereits in dem Konzept der Remote Event erwähnt, soll ein Zugriff auf verschiedenste Informationsquellen möglich sein. Die verschiedenen Arten dieser Quellen unterscheiden sich hauptsächlich durch die Art der Informationen, die durch sie zur Verfügung gestellt werden. Diese grundsätzliche Unterscheidung zwischen den Informationsquellen wird in der zweiten Stufe der Klassenhirarchie erreicht.

Da im Rahmen dieser Arbeit nur der Zugriff auf andere Tracker Systeme umgesetzt wurde, ist die Klasse *Tracker* (Stufe 2) die einzige Ableitung von *Connector*. Die Klasse *Tracker* stellt selbst keine Funktionalität, sondern

definiert lediglich welche Informationen aus einem Fall in einem Tracker extrahiert werden sollen. Des weiteren bindet die Klasse die Klassen zum Export der Daten über ein Strategie-Muster (Gamma u. a., 1995) ein. Da die Daten, die aus den verschiedenen Trackern extrahiert werden gleich sind, bietet sich diese Stelle zur Implementation der Exports an.

In der dritten Stufe finden sich die Klassen, in denen die Algorithmen für den Zugriff und das Extrahieren der Daten implementiert sind. Derzeit ist das Auslesen von Daten aus den folgenden Tracker Systemen unterstützt:

- Debian Bugtracker
- Kolab Bugtracker
- KDE Bugtracker

Die Entscheidung die Klassen in einer Hierarchie von drei Stufen aufzuteilen, wurde aus Gründen der logischen Trennung vor dem Hintergrund einer späteren Erweiterung auf andere Informationsquellen getroffen. Da in der zweiten Schicht nur die gemeinsamen Daten, aber keine Funktionalität definiert werden, ist diese Schicht aus Sicht einer funktionalen Trennung eigentlich überflüssig.

9.4.2 Backends

Das Backend stellt die Verbindung zur lokalen Informationsquelle her, also die Verbindung zu dem Roundup Server. Um die Funktionalität der Remote Events grundsätzlich auch mit anderen Trackersystemen als Roundup betreiben zu können, wurden sämtliche Zugriffe auf die lokale Informationsquelle als Strategie-Muster gekapselt.

Die Klasse *Backend* definiert das Interface für den Zugriff auf die Informationsquelle, während die konkreten Ableitungen die Funktionalität implementieren. Im Rahmen dieser Arbeit wurde jedoch nur der Zugriff auf den Roundup Server realisiert. Desweiteren bindet die Klasse den Zugriff auf ein Mailsystem ein, um in der Lage zu sein im Falle von Veränderungen der Remote Events Benachrichtigungen an die Benutzer zu verschicken.

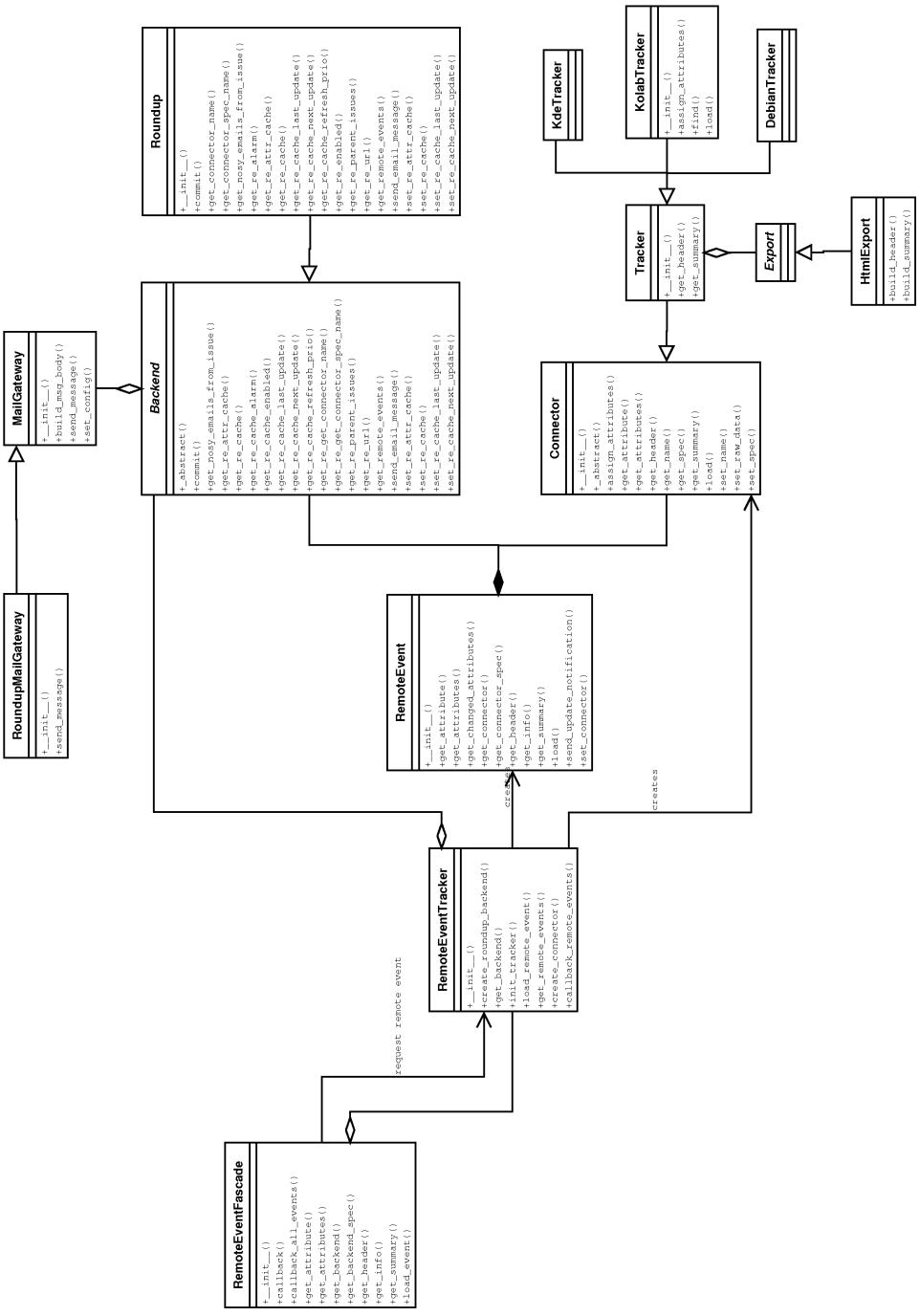

Abbildung 9.4: Klassendiagramm der Remote Events

9.5 Sequenzdiagramm: Remote Event abfragen

Um die Zusammenhänge zwischen den vorgestellten Klassen besser verstehen zu können, wird nun der komplette Ablauf einer Abfrage eines Remote Events anhand des in Abbildung 9.5 dargestellten Sequenzdiagramm erläutert. Als Beispiel wird die Abfrage der Zusammenfassung eines Remote Events verwendet. Das Einbinden der Remote Events erfolgt analog zu dem Einbinden der Zeiterfassung, wie es in Abschnitt 10.4.1 gezeigt wird.

Der Ablauf beginnt mit dem Aufruf der Methode *get_summary(id, err_msg, cached)* in der Fassaden-Klasse. Der Aufruf stammt aus der Weboberfläche. Der Parameter *id* bezeichnet den Remote Event, der geladen werden soll. Über *cached* kann bestimmt werden ob die Zusammenfassung aus dem Cache des Remote Events stammen soll, oder neu erstellt werden soll.

Mit dem Aufruf *load_remote_event(id, cached)* wird die Klasse RemoteEventTracker angewiesen einen RemoteEvent zu erstellen. Der RemoteEventTracker wendet sich dazu zunächst an das Backend (Roundup Server), um in Erfahrung zu bringen, welche Art von Remote Event erstellt werden soll. Hierzu wird zunächst die Art der Informationsquelle (*get_connector_name(id)*) und die genaue Ausprägung dieser Art (*get_connector_spec_name(id)*) abgefragt. Sind diese Informationen bekannt wird das Connector Objekt erstellt.

Die Klasse RemoteEventTracker ruft nun den Konstruktor der Klasse *RemoteEvent* auf. Neben der Id des Remote Events und der Flagge, ob die Inhalte des Events zwischengespeichert sein soll, wird dem Konstruktor auch das Backend und der Connector als Parameter übergeben.

Durch den Aufruf des Konstruktors wird die Initialisierung des Remote Event angestoßen. Aus Gründen der Übersichtlichkeit wurde dieser Prozess in dem Sequenzdiagramm nur schematisch dargestellt. Die Initialisierung beinhaltet zunächst das Laden der Konfiguration des Remote Event aus dem Backend. Dazu gehört die Adresse der entfernten Informationsquelle, zu der mit dem Connector Verbindung aufgenommen werden soll, sowie die Konfiguration zu dem Verhalten des Remote Event (z.B ob eine Benachrichtigung verschickt werden soll). Ebenfalls geladen wird an dieser Stelle der Zwischenspeicher des Remote Event.

Im Anschluss wird die Verbindung mir der entfernten Informationsquelle aufgebaut. Die geladenen Daten werden durch den Connector geladen und die gewünschten Informationen aus dem Datencontainer extrahiert. Diese können nun mit dem zuvor geladenen Zwischenspeichern verglichen werden um eine Veränderung des Datenbestandes festzustellen und gegebenenfalls eine Benachrichtigung zu verschicken. Wurde eine Veränderung festgestellt, so werden die aktualisierten Daten in den Zwischenspeicher geschrieben.

Im Anschluss an die Initialisierung steht der Remote Event in der Fassade zur Verfügung und die Zusammenfassung kann durch den Aufruf der Methode *get_summary()* wieder zurück an die Weboberfläche gegeben werden.

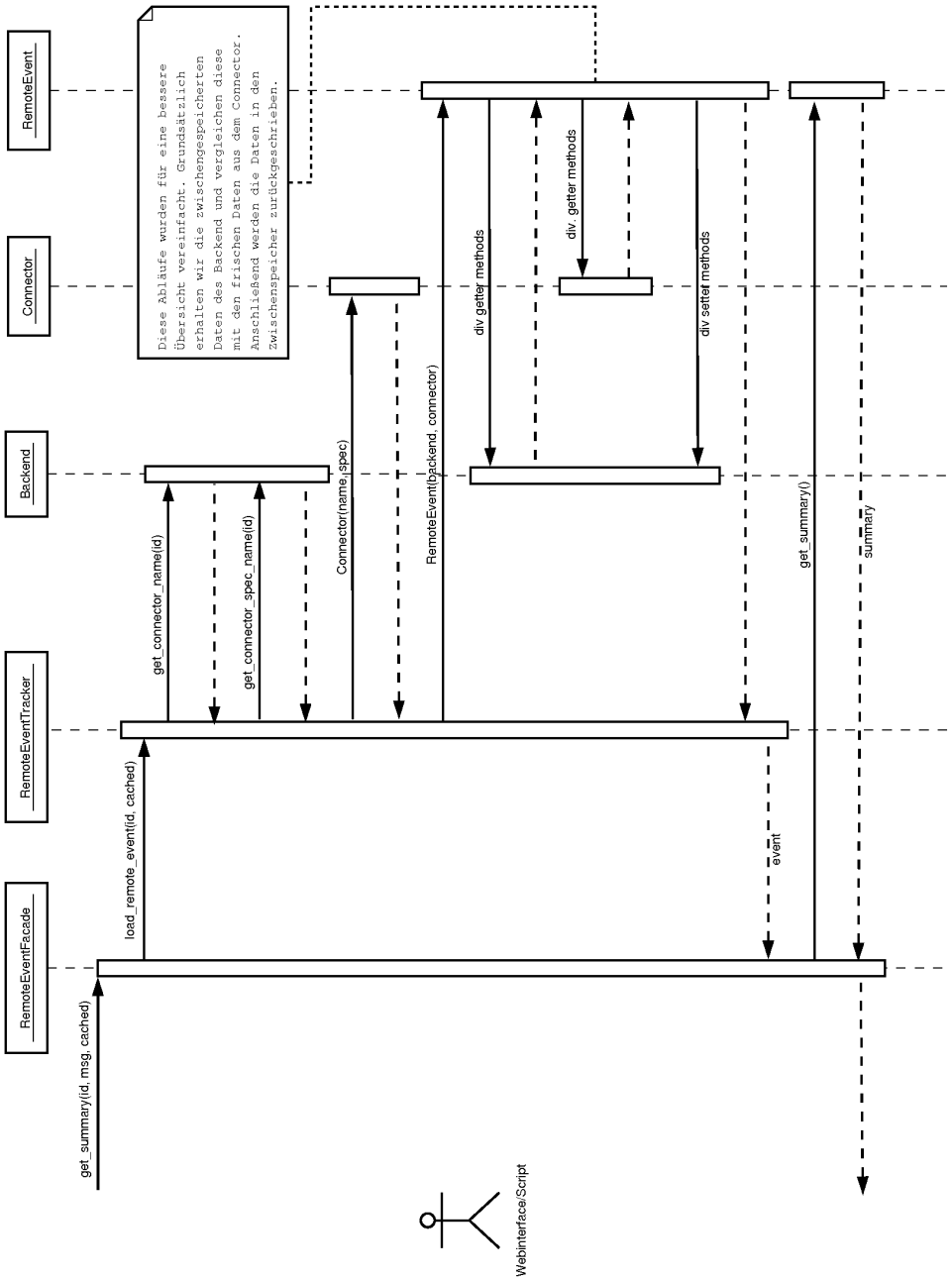

Abbildung 9.5: Seqeunzdiagramm Remote Event

Teil IV

Implementation und Abschluss

Kapitel 10

Implementierung

Im Rahmen dieses Kapitels sollen die zur Implementation der Erweiterung vorgenommenen Schritte dargelegt werden. Zunächst wird die allgemeine Vorgehensweise erklärt, um dann einige konkrete Implementationsschritte anhand von Beispielen zu erläutern.

10.1 Rahmenbedingungen

Die Gestaltung und Implementation der Erweiterungen erfolgen unter den aufgeführten Einschränkungen.

1. Sämtliche Modifikationen und Erweiterungen des Roundup sollen nach Möglichkeit innerhalb einer Trackerinstanz (siehe Abschnitt 3.5) durchgeführt werden.

2. Eine Veränderung der Bibliothek des Roundup soll nur vorgenommen werden, wenn sich die gewünschte Erweiterung nicht in der Instanz realisieren lässt.

3. In der Gesamtlösung sollen die einzelnen Erweiterungen unabhängig voneinander einsetzbar sein.

10.2 Grundsätzliches Vorgehen bei der Implementation

Die Realisierung der einzelnen Erweiterungen erfolgt nach dem Vorbild eines iterativen Entwicklungsmodell, in dem sich die Phasen Analyse, Gestaltung und Implementation in Zyklen solange wiederholen, bis die Anforderungen zur Zufriedenheit erfüllt sind.

Die in diesen Phasen angewandten Methodiken orientieren sich an den Verfahren wie sie aus der objektorientierten Analyse und Design (OOAD)

bekannt sind. Für eine weitergehende Beschreibung der OOAD wird auf (Biermann, 2004) und (Balzert, 1997) verwiesen.

Bei der Erweiterung der Datenbank wurde darauf geachtet, dass die Struktur weitgehend bis zur 3. Normalform konform ist. Die Überführung der Datenstrukturen in die Normalformen dienen der Vermeidung von redundanten Informationen und somit zur Verbesserung der Datenintegrität (Balzert, 1997). Abweichungen davon wurden nur an den Stellen vorgenommen an denen die Datenstruktur durch den Roundup Server vorgegeben wird. Als Beispiel ist hier die Speicherung von Rollen und Verweisen als Multilink zu nennen. In beiden Fällen entspricht die vorgegebene Datenstruktur nicht der 1. Normalform, da die Daten nicht atomar, sondern als kommaseparierte Liste von Werten gespeichert werden.

10.2.1 Chronologie der Entwicklung

Die Entwicklung der einzelnen Erweiterungen erfolgte zeitlich nacheinander in der Reihenfolge, wie sich die Anforderungen aus der Analyse der Geschäftsprozesse ergaben. Abbildung 10.1 zeigt die zeitliche Aufeinanderfolge der Implementierung der einzelnen Erweiterungen.

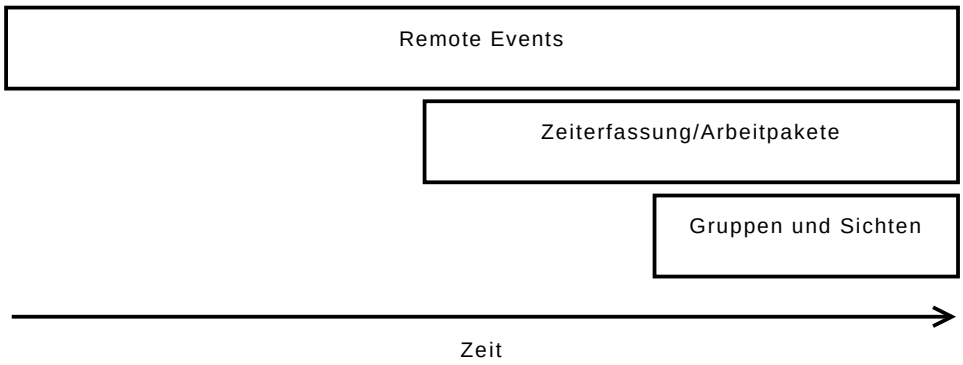

Abbildung 10.1: Beginn der Implementation der Erweiterung

Die Implementation der Remote Events hatte eine relativ lange Vorlaufzeit, da diese Erweiterung gleichzeitig für eine grundsätzliche Einarbeitung in den Roundup Server diente.

Als nächstes begann die Entwicklung der Zeiterfassung und Arbeitspakete. Die Entwicklung verlief mit den Remote Events parallel, da sich in einem der Analyse Zyklen herausstellte, dass sich die Remote Events in Verbindung mit der Zeiterfassung dazu eignen den Aufwand bei der Bearbeitung eines Falls im öffentlichen Tracker über mehrere Kundenfälle zu verteilen.

In der letzten Phase wurde mit der Implementation der Gruppen und Sichten begonnen. Auch diese Phase verlief aufgrund der engen Verzahnung

der Erweiterungen parallel zu den beiden anderen Phasen. Zur Verwaltung und Verfolgungen der Entwicklung der verschiedenen Erweiterungen wurde das Versionskontrollsystem Mercurial[1] verwendet.

Die Implementation wurde über die gesamte Zeit hinweg von Funktionstests begleitet. Die Tests beschränkten sich auf die Überprüfung der Bedienung und der korrekten Funktionsweise anhand der Ausgaben und werden daher als *Black-Box*-Tests bezeichnet (Ernst, 2003). Die Überprüfung erfolgte dabei nicht automatisiert, sondern erforderte das manuelle Durcharbeiten eines Testplans. Der Testplan orientiert sich an den Anwendungsfällen der Erweiterungen und definiert das gewünschte Verhalten bzw. die erwartete Ausgabe bei ihrer Ausführung.

Zum Ende der Implementation der drei Erweiterungen begann die Entwicklung eines Migrationspfads zur Übernahme der Bestandsdaten aus den alten Trackersystemen. Desweiteren wurde die Internationalisierung der Erweiterungen der Weboberfläche durchgeführt. Die Migration der Daten sowie die Internationalisierung werden in den Abschnitten 10.6 (Migration) und 10.7 (Internationalisierung) behandelt. Den Abschluss der Implementation bildet die Inbetriebnahme der Erweiterung. Wie auch die Entwicklung erfolgte die Inbetriebnahme komponentenweise.

10.3 Installation des Roundup Servers

Zu Beginn der Implementierungsphase wurde ein Roundup Server installiert, der als Grundlage für zukünftigen Entwicklungen dienen sollte.

Dieser Abschnitt soll die Installation eines Roundup Server in der Version 1.3.3 aus den Quelldateien erläutern. Die Installation orientiert sich an der in Jones (2006b) beschriebenen Vorgehensweise.

Zunächst werden die Quellen des Roundup Server in das Installationsverzeichnis herunter geladen und dort entpackt. Da das Paket für die Entwicklung nicht systemweit installiert werden soll, wird ein Unterverzeichnis angelegt, in das der Server installiert werden soll.

```
ti@euklid: wget http://cheeseshop.python.org/packages/source/r/
roundup/roundup-1.3.3.tar.gz
ti@euklid: tar xzfv roundup-1.3.3.tar.gz
ti@euklid: mkdir roundup-install
```

Nach einem Wechsel in das Roundup Verzeichnis kann die Installation gestartet werden. Über den Paramter *prefix* lässt sich bestimmen, dass die Installation vollständig in einem vorgegebenen Verzeichnis erfolgen soll. Es wird hier das zuvor angelegte Verzeichnis angegeben.

```
ti@euklid: cd roundup-1.3.3
ti@euklid: python setup.py install prefix=../roundup_install
```

[1] http://www.selenic.com/mercurial/

In einem nächsten Schritt wird der frisch installierte Roundup für die Aufnahme verschiedener Tracker Instanzen vorbereitet. Dazu wird das Verzeichnis "trackers" angelegt. Nun kann ein erster Tracker installiert werden.

Bei der Installation wird auf der Basis einer Vorlage ein neues Verzeichnis mit allen benötigten Dateien erstellt, die für einen Tracker benötigt werden. Dieses Verzeichnis wird *Instanz* genannt. Jede *Instanz* hat eine eigene Datenbank, eigene Konfigurationsdateien und einen eigenen Satz von HTML-Dateien, aus denen die Web-Schnittstelle aufgebaut ist

Anpassungen an einer Tracker Installation werden innerhalb dieser Instanz vorgenommen und haben auch nur auf diese Auswirkung. Andere Tracker Instanzen bleiben davon unberührt.

```
ti@euklid: cd ..
ti@euklid: mkdir roundup-install/trackers
ti@euklid: cd roundup-install/bin
ti@euklid: ./roundup-admin install

Enter tracker home: ../trackers/vanilla
Templates: minimal, classic
Select template [classic]: classic
Back ends: anydbm, mysql, sqlite
Select backend [anydbm]: sqlite
```

In einem letzen Schritt muss die gerade erzeugte Instanz eines Trackers noch konfiguriert werden. Diese Konfiguration umfasst mindestens die Angabe der Adresse unter der der Tracker erreichbar sein soll, sowie die Konfiguration des Email-Interface. Die Konfigurationsdatei befindet sich in der Instanz des Trackers in der Datei *config.py* und kann mit einem beliebigen Editor bearbeitet werden. Für die genauen Schritte der Konfiguration wird an dieser Stelle auf die Dokumentation der Roundup Projekt-Webseite verwiesen. Nachdem die Konfiguration erfolgt ist, wird der Tracker initialisiert und kann gestartet werden.

```
ti@euklid: vim ../trackers/vanilla/config.ini
ti@euklid: ./roundup-admin initialise -i ../trackers/vanilla
ti@euklid: ./roundup-server fast=../trackers/vanilla
```

Nachdem Start des Servers ist dieser unter der konfigurierten Adresse erreichbar und kann genutzt werden.

10.3.1 Installation der Erweiterungen dieser Arbeit

Die Erweiterungen dieser Arbeit wurden so implementiert, dass sie als Vorlage für neue Tracker-Installationen dienen kann. Es wird in diesem Abschnitt davon ausgegangen, dass der Roundup Server nach dem in Abschnitt

10.3 beschriebenen Verfahren installiert wurde. Die Installation der Erweiterungen beginnt in dem Vorlagen Verzeichnis der Roundup Installation. In diesem Verzeichnis befinden sich die Vorlagen für die Installationen der Tracker. Die Erweiterungen dieser Arbeit werden in einem Mercurial Repository verwaltet. Von dort werden die Erweiterungen nun geladen und dann in dem Verzeichnis abgelegt.

```
ti@euklid: cd INSTALLDIR/
ti@euklid: cd share/roundup/templates/
ti@euklid: hg clone http://hg.intevation.org/roundup/fast
```

Die Vorlage steht nun für Installationen bereit. Trotz großer Bemühen sämtliche Änderungen innerhalb der Trackerinstaz durchzuführen, mussten für einige Erweiterungen kleinere Modifizierungen an den Roundup Bibliotheken vorgenommen werden. Die Modifikationen sind in dem Verzeichnis `patches` abgelegt in das als nächstes gewechselt wird. Innerhalb des Verzeichnisses befinden sich fünf Patch-Dateien die nun nacheinander angewendet werden, um die Roundup Bibliotheken anzupassen.

```
ti@euklid: cd fast/patches/
ti@euklid: patch
../../../../../lib/python2.5/site-packages/roundup/security.py
< security.py_usergroup.patch
patching file
../../../../../lib/python2.5/site-packages/roundup/security.py
...repeat for other patches ...
```

Der Roundup Server ist nun vollständig auf die Installation eines um die Erweiterungen modifizierten Tracker vorbereitet.

Die Installation erfolgt exakt nach den gleichen Schritten wie sie in Abschnitt 10.3 beschrieben sind, nur mit dem Unterschied, dass bei der Installation des Trackers eine weitere Vorlage eines Trackers zu Auswahl angeboten wird (FaSt).

```
Enter tracker home: ../trackers/fast
Templates: FaSt, minimal, classic
Select template [classic]: FaSt
Back ends: anydbm, mysql, sqlite
Select backend [anydbm]: sqlite
```

Die Konfiguration und der Start des Trackers erfolgt wieder wie bereits beschrieben.

Aktivierung der einzelnen Komponenten der Erweiterung

In dem Verzeichnis der Instanz des gerade installierten Trackers befindet sich mit `extensions/config.py` eine Konfigurationsdatei, über die sich einstellen

lässt welche Komponenten aktiviert werden sollen. Die Konfigurationsdatei hat den folgenden Aufbau und kann mit einem beliebigen Editor modifiziert werden.

```
[main]
enable_remoteevents = False
enable_timelogs = True
enable_workpackages = True
enable_views = True
```

Die Änderungen an der Datei werden mit einem Neustart der Trackerinstanz übernommen.

10.4 Implementation Zeiterfassung

10.4.1 Einbinden der Zeiterfassung als Extension

Bei Erweiterungen handelt es sich um Python-Scripte, die in der Trackerinstanz in dem Ordner `TRACKERINSTANZ/extensions` abgelegt werden. Diese Scripte werden bei dem Start des Roundup automatisch geladen und initialisiert. Dabei werden die Funktionen der Erweiterung in dem Roundup registriert, so dass diese in der Weboberfläche bekannt sind und genutzt werden können.

Die folgenden zwei Codebeispiele aus der Datei `extensions/timelog_extension.py` sollen zeigen, wie die Erweiterung der Zeiterfassung als Extension in den Roundup eingebunden wird.

```
1  import os
2  import sys
3
4  def init(instance):
5      tracker_home = instance.tracker_home
6      new_import = os.path.join(tracker_home, 'extensions')
7      sys.path.append(new_import)
8      from timelogs import timelog
9      tl = timelog.Timelog(tracker_home)
10     tlf = tl.init_timelog()
```

Die Funktion *init* wird bei dem Start des Roundup automatisch aufgerufen und dient der bereits erwähnten Initialisierung. Als Parameter wird dieser Funktion ein "Instanz-Objekt" übergeben, welches die aktuelle Trackerinstanz repräsentiert, in der die Erweiterung registriert werden soll.

Da die Funktionalität der Zeiterfassung in einem Modul implementiert wurde, gilt es dieses zunächst in der Extension verfügbar zu machen. Dies geschieht durch ein Importieren des Moduls. Bei einem Import sucht Python nach dem zu importierenden Modul an verschiedenen Stellen im System. Die

genauen Verzeichnissen, in denen gesucht wird, sind in der Umgebungsvariablen `PYTHONPATH` angegeben. Diese ist vergleichbar mit der Classpath-Variable aus der Java-Programmierung. Da sich das Modul der Zeiterfassung nicht in einem dieser Verzeichnisse befindet, muss der Pfad temporär zu Laufzeit des Roundup erweitert werden. Dies geschieht in den Zeilen 5 bis 7. Das Modul kann nun in Zeile 8 importiert werden. In der nächsten Zeile wird ein Objekt der Klasse *TimeLog* erzeugt und anschließend initialisiert. Die Funktion *init_timelog* erzeugt ein Fassadenklassen-Objekt, welches die Funktionalität des Moduls der Zeiterfassung über eine definierte Schnittstelle verfügbar macht.

In den folgenden Zeilen werden die Methoden der Fassadenklasse in dem Roundup Server registriert, so dass sie in der Weboberfläche verfügbar sind und genutzt werden können.

```
1 #Methods used for timelog listing on issues
2 instance.registerUtil('tl_print_timelogs', tlf.↘
    →print_timelogs)
3 instance.registerUtil('tl_get_total', tlf.get_total_time)
4 instance.registerUtil('tl_get_total_assigned', tlf.↘
    →get_total_assigned_time)
5 instance.registerUtil('tl_get_total_unassigned', tlf.↘
    →get_total_unassigned_time)
6 instance.registerUtil('tl_get_user_timelogs', tlf.↘
    →get_user_timelogs)
7 instance.registerUtil('tl_timelogs_available', tlf.↘
    →timelogs_available)
```

Die Registrierung erfolgt durch den Aufruf der Funktion *instance.registerUtil* die zwei Parameter entgegen nimmt. Der zweite Parameter ist dabei die Methode der Fassadenklasse, die registriert werden soll. Der erste Parameter ist der Name unter welchem die Funktion in der Weboberfläche bekannt sein soll.

Eine Verwendung dieser Erweiterung in der Weboberfläche erfolgt dann beispielhaft über den Einbau folgender Anweisung in einer der HTML-Dateien:

```
1 <span tal:content="python:utils.tll_print_timelogs(context↘
    →)"/>
```

Dabei wird der gesamte “” *Tag* durch den Rückgabewert der Erweiterungsfunktion ersetzt. In diesem Fall wird eine Auflistung aller Zeiten ausgegeben. Der Paramenter *context* ist eine Variable aus der Weboberfläche, die festlegt für welches Objekt (Remote Event, Kundenfall) eine Auflistung der Zeiten ausgegeben werden soll. Weitere Informationen zum Einbinden der Erweiterung in die Weboberfläche findet sich in Abschnitt 10.4.2.

10.4.2 Anpassungen der Weboberfläche

Die Bedienung der Erweiterungen erfolgt über die Weboberfläche. Diese setzt sich aus einer Reihe von HTML-Dateien zusammen, die von dem Benutzer verändert und angepasst werden können. Die HTML-Dateien enthalten Elemente der Vorlagensprache TAL, die dazu dient die Weboberfläche dynamisch auf Grundlage der Daten des Roundup zu erstellen. TAL-Elemente bieten die Möglichkeit sowohl lesend als auch schreibend auf die Daten zuzugreifen.

In den folgenden Beispielen wird sowohl die Realisierung der Zeiteingabe als auch die Auflistung der Zeiten beschrieben.

Eingabe von Zeiten

Die Eingabe der Zeiten erfolgt durch das Eintragen eines Zeitwerts in einem Textfeld über die Weboberfläche. Um einen Zeiteintrag speichern zu können, muss eine Änderungsnotiz erstellt werden. Der Grund hierfür liegt in der Datenstruktur die vorgibt, dass Zeiteinträge an Änderungsnotizen gespeichert werden (vgl. Abschnitt 8.3).

Abbildung 10.2 zeigt den relevanten Ausschnitt der Weboberfläche während der Bearbeitung bzw. der Erstellung eines Falls. Das Textfeld mit der Bezeichnung Timelog ermöglicht einen Zeiteintrag nach dem angegebenen Muster einzutragen. Versucht der Benutzer den Eintrag ohne eine Änderungsnotiz oder mit fehlerhaft eingegebener Zeit zu speichern, wird eine Fehlermeldung angezeigt. Die Fehlermeldung wird von Roundup automatisch generiert und bedarf keiner weiteren Implementierung.

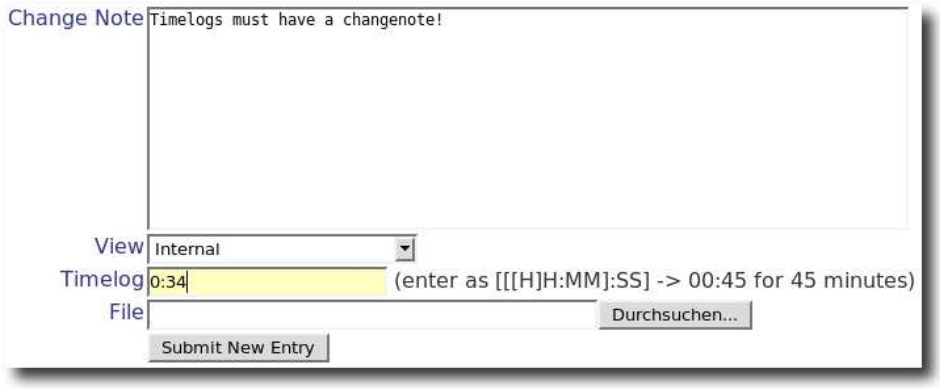

Abbildung 10.2: Textfeld zur Eingabe von Zeiten

Das Auswahlfeld oberhalb des Textfeldes zur Eingabe der Zeit dient zur Bestimmung in welcher *Sicht* die Änderungsnotiz nach dem Abspeichern zu

sehen sein wird (siehe Abschnitt 7.2.1.2), hat für das Eintragen der Zeiten keine Bedeutung.

Es sollen nun die zugehörigen Zeilen HTML zur Anzeige dieses Textfeldes näher erläutert werden.

```
1 <tr tal:condition="python:request.user.hasPermission('↘
    →Create', 'timelog') and db.config.ext['ENABLE_TIMELOGS↘
    →']=='True'">
2  <th i18n:translate="">Timelog</th>
3  <td colspan=3 tal:define="empty_time string:">
4     <input type="text" name="timelog-1@time" tal:↘
        →attributes="value string:${request/form/timelog-1↘
        →@time/value | empty_time}"/>
5   <span i18n:translate="">timelog_example</span>
6   <input type="hidden" name="msg-1@link@timelog" value="↘
      →timelog-1" />
7  </td>
8 </tr>
```

Die erste Zeile des Beispiels zeigt eine bedingte Anweisung aus der Vorlagensprache TAL. Sie besagt, dass das Eingabefeld nur angezeigt wird, wenn

1. der Benutzer die Befugnis hat neue Zeiteinträge zu erstellen und
2. die Erweiterung der Zeiterfassung aktiviert wurde.

Ruft der Benutzer diese Seite auf, so findet zunächst eine Auswertung dieser Anweisung durch das Rechtesystem statt. Trifft eine der beiden Bedingungen nicht zu, so wird die gesamte Zeile der Tabelle (<tr>) und ihrem Inhalt in der generierten Webseite nicht angezeigt.

Das eigentliche Eingabefeld wird in Zeile 3 und 4 definiert. In Zeile 3 wird eine Variable empty_time definiert, die eine leere Zeichenkette enthält und nur lokal innerhalb der Tabellenspalte (<td>) sichtbar ist.

Zeile 4 enthält nun das Eingabefeld. Die Parameter des Eingabefeld sollen hier kurz erläutert werden.

name gibt an wo der eingetragene Zeitwert gespeichert werden soll. In diesem Fall soll die Zeit in dem Attribut *time* des zu erstellenden Zeiteintrags in der Tabelle timelog gespeichert werden. Die Kennzeichnung timelog-1 ist für die interne Verarbeitung in Roundup wichtig und gibt an, dass es sich hier um den Eintrag handelt, der als nächstes erstellt wird.

value gibt an was gespeichert werden soll. Die Anweisung führt dazu, dass der Zeitwert nicht in der vom Benutzer eingegebenen Form, sondern als Datentyp Interval gespeichert wird, so wie es die Datenbank fordert. Da es natürlich auch möglich sein soll Nachrichten ohne einen Zeiteintrag zu erstellen, wird dann an Stelle der Zeit die zuvor definierte Variable empty_time ausgewertet, wodurch eine Fehlermeldung des Roundup

vermieden werden kann. Für eine genaue Erklärung der Syntax wird auf die Dokumentation des Roundup verwiesen (Jones, 2006a).

Zeile 6 enthält ein verstecktes Formularfeld, welches dafür sorgt, dass der neu erstellte Zeiteintrag mit der Nachricht verknüpft wird.

Die Anweisungen werden nach dem Absenden serverseitig ausgewertet. Das Erstellen der Einträge, sowie die Verlinkung der Elemente, geschieht automatisch und erfordert keine weitere Implementierung.

Auflistung der Zeiten

Als zweites Beispiel wird die Auflistung der Zeiten eines Falls beschrieben. Die in Abbildung 10.3 dargestellte Auflistung zeigt eine detailierte Übersicht aller Zeiten zu einem Fall. Aus Gründen des Datenschutzes ist die Übersicht nur für Benutzer sichtbar, die die Rolle Manager besitzen. Die Bedeutung

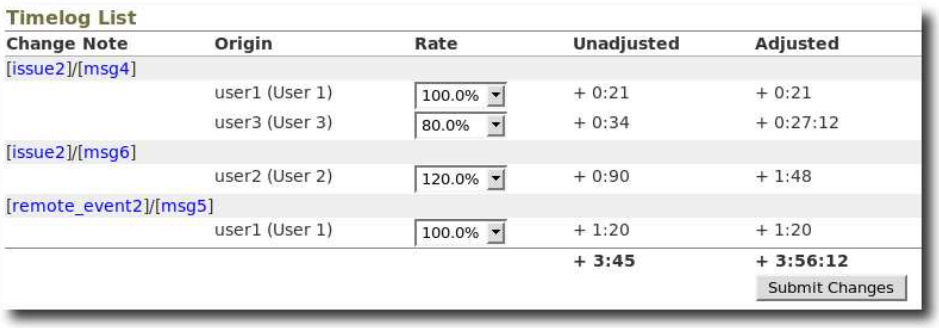

Abbildung 10.3: Auflistung der Zeiten in einem Fall

der einzelnen Spalten wird nun kurz erklärt.

Change None gibt den "Pfad" zu der Änderungsnotiz an, in der die Zeit eingetragen wurde. Die Einträge sind verlinkt und ermöglichen so ein schnelles Anspringen der Einträge, um z.B. zu überprüfen für welche Aufgabe die Zeit eingetragen wurde. In der Abbildung 10.3 ist z.B. zu sehen, dass in der Änderungsnotiz 4 des Fall 2 zwei Benutzer Zeiten eingetragen haben.

Origin zeigt an, von welchem Benutzer diese Zeit eingetragen wurde.

Rate legt die Bewertung der Zeit fest. Der Inhalt des Auswahlfeldes wird aus den Einträgen der Tabelle timelog_rate erstellt (siehe 8.3).

Unadjusted enthält die Bruttozeit, so wie sie von dem Benutzer eingetragen wurde.

Adjusted enthält die Nettozeit. (bewertete Bruttozeit).

Unterhalb der Auflistung ist noch eine Aufsummierung der Zeiten zu sehen.

Die konkrete Implementierung der Übersicht soll anhand des nachfolgenden Codebeispiel gezeigt werden. Zeile 1 leitet das Beispiel mit einer Tabelle ein, in der die Auflistung der Zeiten realisiert ist. Die geforderte Beschränkung des Zugriffs auf die Auflistung wird erneut über eine bedingte TAL-Anweisung realisiert. Die Zeitauflistung wird somit nur angezeigt, wenn

1. der Benutzer die Rolle Manager hat und
2. die Erweiterung der Zeiterfassung aktiviert wurde.

In den Zeilen 3 bis 9 wird die Kopfzeile der Tabelle erstellt. Sie enthält die fett gedruckten Überschriften. Zeile 10 enthält einen Aufruf der Funktion tll_print_timelogs die Teil der Zeiterfassungs-Erweiterung ist. Die Funktion hat als Rückgabewert die Auflistung der eigentlichen Zeiten inklusive aller zusätzlichen Felder wie z.B. die Auswahlfelder zur Bewertung. In der endgültigen Ausgabe wird die Zeile 10 durch den Rückgabewert der Funktion ersetzt.

```
1  <table class="timelog" tal:condition="python:request.user.↘
       →hasRole('Manager') and db.config.ext['ENABLE_TIMELOGS↘
       →']=='True'">
2   <tr><th colspan="6" class="header" i18n:translate="">↘
        →Timelog List</th></tr>
3   <tr>
4    <th i18n:translate="">Change Note</th>
5    <th i18n:translate="">Origin</th>
6    <th i18n:translate="">Rate</th>
7    <th i18n:translate="">Unadjusted</th>
8    <th i18n:translate="">Adjusted</th>
9   </tr>
10  <tr tal:replace="structure python:utils.↘
       →tll_print_timelogs(context)"></tr>
11  <tr>
12   <td class="sum"></td>
13   <td class="sum"></td>
14   <td class="sum"></td>
15   <td class="sum"><span tal:content="python:utils.↘
       →tll_get_total(context)"/></td>
16   <td class="sum"><span tal:content="python:utils.↘
       →tll_get_total(context,Rate=True)"/></td>
17  </tr>
18  <tr>
19   <td colspan="4"></td>
20   <td><span tal:replace="structure context/submit">submit ↘
       →button</span></td>
21  </tr>
22 </table>
```

In den Zeilen 11 bis 17 wird die Tabellenreihe mit der Summierung erstellt. Die summierten Ergebnisse stammen aus einem Aufruf der Funktionen ttl_get_total. Bei dem zweiten Aufruf wird der Funktion zusätzlich der Parameter Rate mitgegeben, der die Zeiterfassungs-Erweiterung anweist, die bewertete Zeit als Rückgabewert zu liefern.

10.5 Erweiterung des Rechtesystems

In dem folgenden Abschnitt soll exemplarisch dargestellt werden welche Erweiterungen an dem Rechtesystem vorgenommen wurden, um die für die Sichten nötige Funktionalität zu implementieren. Für eine ausführliche Beschreibung zur Modifizierung des Rechtesystem in Roundup wird auf Jones (2006a) verwiesen.

Das folgende Beispiel zeigt eine Zusammenstellung einiger Funktionen aus der Datei `TRACKERINSTANZ/schema.py`, die einen lesenden Zugriff eines Kunden auf eine Änderungsnotiz reglementieren. Der Beispielcode teilt sich dabei in zwei Teile auf.

1. Die Erstellung einer Permission und die Zuweisung dieser Permission an eine Rolle.

2. Die Erweiterung des Rechtesystem durch das Erstellen benutzerdefinierter Funktionen, in denen die für die Sicht nötige Funktionalität implementiert werden.

```
1  # 1. Part
2  # Creating permissions and assigning them to user roles
3  vm = db.security.addPermission(name='View',
4       description='Customer is allowed to view msgs which ↘
           →are public',
5       properties=('attribute1',
6                   'attribute2',
7                   'attributeN')
8       klass='msg',
9       check=view_msg)
10 db.security.addPermissionToRole('Customer', vm, 'msg')
```

Im ersten Teil wird eine neue Permission für einen lesenden Zugriff (View) erstellt. Dazu wird die Funktion *addPermission* des Rechtesystem aufgerufen. Sie nimmt eine Reihe von Parametern entgegen:

name Diese Parameter legt fest für welche Art von Zugriff diese Permission erstellt wird. Der Name *View* ist vom System vorbelegt und legt fest, dass es sich um einen lesenden Zugriff handelt. Alternativ können auch *Create* und *Edit* angegeben werden.

description gibt der Permission eine kurze Beschreibung.

klass legt fest für welche Klasse diese Permission gilt.

properties legt fest für welche Attribute der Klasse diese Persmission gilt.

check Durch den Benutzer lässt sich an dieser Stelle eine eigene Funktion einbinden, mit der sich spezielle Tests im Rahmen der Rechteüberprüfung realisieren lassen. Bei einer Überprüfung der Zugriffsrechte wird als erstes diese Funktion überprüft. Nur wenn diese Überprüfung positiv verläuft, wird mit der übrigen Rechteüberprüfung fortgefahren.

Nachdem eine solche Permission erstellt wurde, wird sie über die Funktion *addPermissionToRole* einer Rolle zugewiesen.

```
1  # 2. Part
2  # Enhancing the permission system with user definded ↘
       →functions
3  def view_msg(db, userid, itemid):
4      '''Checks if the user is allowed to see the msg. ↘
           →Returns False if the user
5      is not permitted to see it else True'''
6      view_id = db.msg.get(itemid, 'view')
7      return check_permission(db, userid, view_id)
8
9  def check_permission(db, userid, view_id):
10     if db.config.ext['ENABLE_VIEWS']=='False':
11         return True
12     allowed_views = []
13     groups = db.usergroup.filter(None, {'member': userid})
14     for g in groups:
15         allowed_views.extend(db.usergroup.get(g, 'show_msg↘
               →'))
16     if view_id in allowed_views:
17             return True
18     return False
```

Der zweite Teil zeigt die benutzerdefinierten Funktionen. Die Funktion *view_msg* wurde durch die "View-Permission" aus dem ersten Teil eingebunden und wird aufgerufen sobald ein lesender Zugriff durch den Kunden auf Änderungsnotizen stattfindet. Bei ihrem Aufruf wird die ID des zugreifenden Benutzers (user_id) sowie die ID der Änderungsnotiz (item_id), auf die zugegriffen werden soll übergeben. Eine Verbindung zur Datenbank ist über das übergebene Datenbank-Objekt möglich (db).

In Zeile 6 wird zunächst geladen, in welcher Sicht die Nachricht zu sehen sein soll. Dazu wird die get-Methode der msg-Klasse des Datenbank-Objekts aufgerufen. Diese Sicht wird der Funktion *check_permission* zu einer weiteren Überprüfung übergeben.

Diese Funktion übernimmt die Überprüfung, ob der Benutzer berechtigt ist auf die Änderungsnotiz zuzugreifen. Dazu wird in Zeile 13 eine Filterfunk-

tion des Benutzergruppen-Objekts aufgerufen, die als Rückgabewert eine Liste von Benutzergruppen zurück gibt, in der der Benutzer Mitglied ist.

Sind die Gruppen geladen, so können von diesen die *Views* geladen werden, in denen die Mitglieder auf die Nachrichten zugreifen dürfen. Zuletzt findet in Zeile 16 eine Überprüfung statt, ob es eine Übereinstimmung zwischen den Sichten gibt, in der der Benutzer auf Nachrichten zugreifen darf und der Sicht die der Nachricht zugewiesen wurde. Das Ergebnis dieser Überprüfung wird als boolscher Wert an das Rechtesystem zurückgegeben. Ist das Ergebnis dieser Überprüfung negativ, wird der Zugriff für den Benutzer auf die Änderungsnotiz verweigert. Bei einem positiven Ergebnis wird mit der Auswertung der Zugriffsrechte auf die einzelnen Attribute der Änderungsnotiz fortgefahren.

10.5.1 Vererbung von Zugriffsrechten

Um das Konzept der Benutzerrollen auch auf die Benutzergruppen übertragen zu können, musste eine Änderung an einer der Roundup Bibliotheken vorgenommen werden. Das nachfolgende Codebeispiel zeigt den Patch für die Datei `INSTALLDIR/lib/python2.5/site-packages/roundup/security.py`. Die Funktionen innerhalb dieser Datei übernehmen einen Großteil der Aufgaben des Rechtesystems in Roundup. Zeile 15 und 16 stammen aus der Funktion *has_permision*, die im Rahmen der Rechteüberprüfung aufgerufen wird und testet ob der Benutzer die nötigen Zugriffsrechte hat. Die Überprüfung erfolgt auf Grundlage der Rollen, die dem Benutzer zugewiesen wurden. Um nun eine "Vererbung" der Rollen aus dem Benutzergruppen zu erreichen, wurde der ursprüngliche Aufruf in Zeile 15 durch einen Aufruf einer neu hinzugefügten Funktion ausgetauscht. Die Zeilen 24 bis 40 zeigen diese neue Funktion.

```
1  --- security_old.py       2007-02-04 12:09:16.000000000 ↘
       →+0100
2  +++ security.py 2007-02-06 18:06:10.000000000 +0100
3  @@ -3,6 +3,7 @@
4    __docformat__ = 'restructuredtext'
5
6    import weakref
7   +from sets import Set
8
9    from roundup import hyperdb, support
10
11 @@ -166,7 +167,7 @@
12               Note that this functionality is actually ↘
                     →implemented by the
13               Permission.test() method.
14           '''
15  -        roles = self.db.user.get(userid, 'roles')
16  +        roles = self.getUserRoles(userid)
```

```
17            if roles is None:
18                return 0
19            if itemid and classname is None:
20 @@ -181,6 +182,23 @@
21                                userid, itemid):
22                            return 1
23            return 0
24 +
25 +    def getUserRoles(self, userid):
26 +        ''' Returns a combined list of user roles and ↘
   →roles inherited from the usergroups '''
27 +        roles = Set()
28 +        userroles = self.db.user.get(userid, 'roles')
29 +        if userroles is not None:
30 +            roles.union_update(Set(userroles.split(',')))
31 +        # Add roles inherited from the usergroups
32 +        groups = self.db.usergroup.filter(None, {'member'↘
   →: userid})
33 +        for gid in groups:
34 +            r = self.db.usergroup.get(gid, 'roles')
35 +            if r is None:
36 +                continue
37 +            roles.union_update(Set(r.split(',')))
38 +        if len(roles) == 0:
39 +            return None
40 +        return ",".join(list(roles))
41
42      def addPermission(self, **propspec):
43          ''' Create a new Permission with the properties ↘
               →defined in
```

Die Funktion lädt zunächst eine Liste der Rollen des Benutzers. Diese Liste wird aber dann um die Rollen der Benutzergruppen erweitert, in denen der Benutzer Mitglied ist. Zum Schluss wird eine kombinierte Liste aus Benutzer- und Gruppenrollen zurück gegeben.

10.6 Migration der bestehenden Daten

Die bestehenden Daten sollen aus dem veralteten, öffentlichen Tracker in eine aktuelle Version des Trackers übernommen werden. Der aktualisierte Tracker enthält die in dieser Arbeit vorgestellten Erweiterungen, die aber noch nicht aktiviert sind.

Aufgrund einiger Veränderungen in der Datenstruktur des Roundup Server zwischen der Version 0.6.2 und Version 1.3.3 ist eine direkte Übernahme der Daten nicht ohne Weiteres möglich. Die Migration der Daten verläuft daher in zwei Schritten:

1. Aktualisierung der vorhanden Tracker Installationen auf Version 1.3.3.

Eine Aktualisierung muss pro Trackerinstanz durchgeführt werden

2. Übernahme der Daten in den erweiterten Tracker

Die Dokumentation des Roundup Servers (Jones, 2006d) sieht eine schrittweise Aktualisierung vor, da eine korrekte Konvertierung der Datenstruktur bei der Aktualisierung nur zwischen direkt aufeinander folgenden Versionen gewährleistet ist. Eine Aktualisierung auf die nächst höhere Version erfolgt dabei in zwei Schritten. In einem ersten Schritt werden die Weboberfläche und die Konfigurationsdateien (z.B. `schema.py`) aktualisiert. Die Änderung an den Dateien erfolgen nicht automatisch, sondern müssen manuell ausgeführt werden. Eine Anleitung an welchen Stellen die Dateien geändert werden müssen, wird auf den Projektseiten des Roundup bereitgestellt. In dem zweiten Schritt muss die Konvertierung der Datenstruktur durchgeführt werden. Dazu verfügt die jeweils aktuelle Version des Roundup Server über einen Mechanismus, der die Datenstruktur der Daten aus der Vorgängerversion bei einem Zugriff automatisch aktualisiert. Der Zugriff auf die Daten kann am einfachsten durch das Kommandozeilen-Werkzeug *roundup-admin* durchgeführt werden. Nach dem Zugriff ist die Aktualisierung auf eine neue Version abgeschlossen.

Um den Aufwand für künftige Aktualisierungen zu verringern, wurde wie folgt vorgegangen: Es wurde eine Aktualisierung eines unmodifizierten Tracker von der Version 0.6.2 auf die Version 1.3.3 nach den Vorgaben der Anleitung vorgenommen. Dabei wurden von den Differenzen zwischen den verschiedenen Versionen entlang des Aktualisierungspfades Patches erstellt, die in künftigen Aktualisierungen wiederverwendet werden können. Diese Patches enthalten die Änderungen in der Weboberfläche und den Konfigurationsdateien.

Der Aufwand zur Überführung eines weiteren Tracker in eine neue Version reduziert sich somit auf das Anwenden der Patches, und einer anschließenden Aktualisierung der Datenstruktur durch einen kurzen einmaligen Zugriff auf die Daten. Der Aktualisierungspfad stellt sich wie in Abbildung 10.4 dar.

Nachdem der Tracker vollständig in die Version 1.3.3 überführt wurde, können die Daten durch eine Kopie in den erweiterten Tracker übernommen werden.

```
$ cp -a OLD_TRACKERINSTANZ/db NEW_TRACKERINSTANZ/db
```

Zusätzliche Änderungen an dem Datenbank Schema, die sich durch die Erweiterungen ergeben, werden automatisch bei dem nächsten Zugriff auf die Daten hinzugefügt. Die Migration ist mit diesem Punkt abgeschlossen.

der

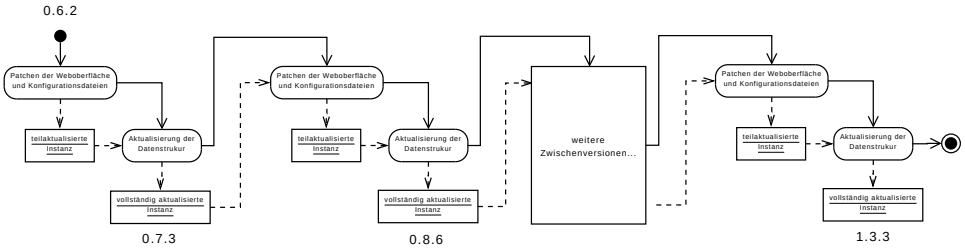

Abbildung 10.4: Upgrade einer Instanz über mehrere Zwischenversionen

10.7 Internationalisierung

Zur Übersetzung der Textausgaben in der Weboberfläche wird die GNU Internationalisierungs-Bibliothek *gettext*[2] verwendet. Die Übersetzung erfolgt dabei lokal für eine Trackerinstanz nach den folgenden Schritten:

1. Kennzeichnung der zu übersetzenden Zeichenketten
2. Erzeugen einer POT (Portable Object Template) Datei
3. Erzeugen einer PO (Portable Object) Datei
4. Übersetzung der PO Datei
5. Kompilieren der PO Datei in eine MO Datei (Machine Object)

Die zu übersetzenden Zeichenketten müssen in den HTML-Dateien der Weboberfläche speziell gekennzeichnet werden. Die Vorlagensprache TAL ist für eine solche Kennzeichnung vorbereitet.

```
<th i18n:translate="">Change Note</th>
```

Das Beispiel zeigt eine von vielen solcher gekennzeichneten HTML-Tags.

Nachdem sämtliche zu übersetzenden Zeichenketten markiert wurden, wird mit Hilfe des Programms *roundup-gettext* die markierten Strings aus der HTML-Dateien extrahiert und in die POT Datei `TRACKERINSTANZ/locales/messages.pot` gespeichert. Diese Datei enthält eine Liste aller zu übersetzenden Zeichenketten. Ein Beispiel einer solchen POT Datei ist nachfolgend dargestellt

```
... [weitere Einträge] ...

#: ../html/issue.item.html:140
```

[2] http://www.gnu.org/software/gettext/

```
#: ../html/page.html:276
#: ../html/workpackage.item.html:99
#: ../html/remote_event.item.html:88
msgid "Change Note"
msgstr ""

... [weitere Einträge] ...
```

Diese Datei dient als Vorlage für die Internationalisierung in andere Sprachen. Um eine Übersetzung in die deutsche Sprache vorzubereiten, wird mit dem Befehl *msginit* eine länderspezifische PO Datei erzeugt.

```
$ msginit -locale=de -input=messages.pot
```

Diese Datei heißt entsprechend der Internationalisierung `TRACKERINSTANZ/locales/de.po`. Die Übersetzung erfolgt durch das Eintragen der übersetzten Zeichenkette in die Variable *msgstr*.

```
... [weitere Einträge] ...

#: ../html/issue.item.html:140
#: ../html/page.html:276
#: ../html/workpackage.item.html:99
#: ../html/remote_event.item.html:88
msgid "Change Note"
msgstr "Änderungsnotiz"

... [weitere Einträge] ...
```

Zuletzt muss die PO Datei noch mit dem Befehl *msgfmt* kompiliert werden, damit der Roundup Server diese Datei zur Internationalisierung verwenden kann.

```
ti@euklid: msgfmt -statistics -o de.po de.mo
```

Bei der Übersetzung versucht gettext zunächst eine passende Übersetzung in einer der MO-Dateien aus dem `TRACKERINSTANZ/locales` Verzeichnis zu finden. Wird dort keine passende Übersetzung gefunden, so werden die systemweiten MO-Dateien für den Roundup Server konsultiert. Die Auswahl der Sprache in die übersetzt werden soll, geschieht über die Auswertung des HTTP-Headers, der von dem Browser bei dem Aufruf der Webseite übersendet wird.

Kapitel 11

Diskussion

11.1 Überprüfung der Ziele

Zur Bewertung des Projekterfolgs wird eine Überprüfung der Ziele anhand der zu Beginn dieser Arbeit gestellten Forderung vorgenommen. Die anfangs gestellte Forderung einer Verbesserung der Unterstützung von Arbeitsabläufen lässt sich zum derzeitigen Zeitpunkt nicht überprüfen, da sich die Erweiterungen noch nicht vollständig und lang genug im produktiven Einsatz befinden. Eine Überprüfung dieses Punktes reduziert sich somit auf eine Betrachtung der Vollständigkeit der Musskriterien. Sie können als Voraussetzung für die geforderte Verbesserung der Abläufe gesehen werden.

Vollständigkeit der Musskriterien Die Vollständigkeit der Musskriterien wurde erreicht. Die Musskriterien wurden vollständig nach den gestellten Forderungen implementiert. Neben den Musskriterien wurde mit den Arbeitspaketen auch ein Wunschkriterium implementiert.

Mit den Benutzergruppen und Sichten wurden die Voraussetzungen für eine Verbesserung der Kommunikation zwischen Personen im Rahmen des Kundendienstes geschaffen. Die bisher zwingende Schnittstelle durch den Mitarbeiter, zwischen Kunden und Partner, kann durch die Öffnung des internen Trackers für einen gemeinsamen Zugriff entfallen.

Die Zeiterfassung bietet die Möglichkeit Zeitaufwände während der Fallbearbeitung nun viel effektiver verfolgen zu können. In Verbindung mit den Arbeitspaketen und den Remote Events entsteht so ein Werkzeug, was die Erstellung von Rechnungen im Vergleich zu Ausgangssituation stark vereinfachen sollte.

Remote Events bieten durch die Modellierung der Verbindung von verschiedenen Tracker Systemen das Potential zur Verbesserung des Arbeitsfluss im Kundendienst.

Eine erneute Auflistung dieser Forderungen entfällt an dieser Stelle mit dem Hinweis auf Kapitel 6.

Produktivbetrieb der Erweiterungen Zum Zeitpunkt des Schreibens dieser Arbeit befindet sich mit der Zeiterfassung eine von drei Erweiterungen im produktiven Einsatz. Ein mit den Erweiterungen modifizierter Tracker dient aber bereits als Grundlage für neu aufgesetzte Trackerinstanzen bei der Intevation GmbH. Es ist geplant weitere Erweiterung in der kommenden Zeit zu aktivieren und in den Produktivbetrieb zu übernehmen.

Veröffentlichung und Rückführung der Arbeitsergebnisse Eine vollständige Integration der Ergebnisse in den Hauptstrom der Roundup Entwicklung konnte nicht erreicht werden. Stattdessen wurden die Ergebnisse dieser Arbeit auf einer eigens dafür eingerichteten Projekt Webseite[1] veröffentlicht und sind frei verfügbar. Auf der Webseite des Roundup Projekts wird auf diese verlinkt. Dies entspricht nach den Beobachtungen auch der gängigen Praxis in der Gemeinschaft des Roundup Projekts. Erweiterungen des Servers werden nur sehr konservativ in den Hauptstrom integriert, sondern vielmehr mit Anleitungen zu deren Benutzung in dem Wiki des Roundup Projekts veröffentlicht.

Die Ziele der Arbeit wurden somit nicht vollständig erreicht. Speziell der Projektabschluss mit der Übernahme der Erweiterungen in den Produktivbetrieb und die Integration der Arbeitsergebnisse in den Hauptstrom ist nicht wie gefordert erreicht worden.

11.2 Bewertung der Vorgehensweise

Eine Bewertung der Vorgehensweise findet aufgrund der Struktur dieser Arbeit getrennt über ihre beiden Teilbereich statt.

1. Einer Analyse von Geschäftsprozessen im Kolab-Kundendienst mit dem Ziel dortige Abläufe zu identifizieren, die sich durch ein IT-System besser unterstützen lassen.

2. Der Realisierung dieses IT-Systems zur Unterstützung der identifizierten Abläufe.

Für die konkrete Realisierung des IT-Systems konnten im zweiten Teil der Arbeit, die im Rahmen des Studiums an der Fachhochschule Osnabrück erlernten Methoden erfolgreich angewendet werden. Die Realisierung der einzelnen Erweiterungen erfolgte entlang der Verfahren aus der Vorlesung zur objektorientierten Analyse und Design. Allerdings muss festgestellt werden, dass die Reihenfolge in der die Erweiterungen realisiert wurden bei nachträglicher Betrachtung nicht optimal war. Das Design der einzelnen Erweiterungen hätte sich bei veränderter Reihenfolge sicher verbessern lassen.

[1] http://trac.irlaender.de/fast

Die Analyse der Geschäftsprozesse wurde anfänglich unterschätzt und stellte sich als komplizierter und aufwendiger heraus als zunächst erwartet.

Es ergaben sich hierdurch Verzögerungen, die letztlich dazu führten, dass mit der Erarbeitung der Erweiterungen begonnen wurde, bevor ein Gesamtbild der Geschäftsprozesse vollständig erfasst war.

11.3 Offene Probleme

Zum Zeitpunkt des Schreibens dieser Arbeit befinden sich 6 offene Fehler und 9 Erweiterungswünsche bzw. Anpassungen des derzeitigen Verhaltens der Software in dem Bugtracker auf der Projektwebseite[2] der Erweiterungen.

Die Fehler sind aber unkritisch, und führen nicht zu fehlerhaften Daten oder Progammabstürzen. Als Beispiel für einen solchen Fehler soll hier die Problematik in der Zeiterfassung gezeigt werden.

Wie in Abschnitt 8 beschrieben, bietet die Zeiterfassung die Möglichkeit der Bewertung einer eingetragenen Zeit. Diese Bewertung wird in der Tabelle timelog über einen Link mit der Tabelle timelog_rate gespeichert. Auf diese Weise wird die 2. Normalform (Balzert, 1997, S.769) des Datenbankschemas erreicht, die zu einer Vermeidung der Redundanz der Bewertungsinformation (Prozentangabe) in der Tabelle timelog führt. Die Vermeidung von Redundanzen dient im Allgemeinen zur Verbesserung der Datenintigrität. Da es sich bei den Bewertungen der Zeiten aber um individuelle Angaben pro Zeit handelt, ist eine Redundanz an dieser Stelle ausdrücklich erwünscht. Die derzeitige Datenstruktur wird an dieser Stelle als fehlerhaft betrachtet.

Die Funktionstests der Software wurden aus Mangel an zeitlichen Ressourcen der Mitarbeiter weitgehend durch den Entwickler selbst durchgeführt. Daher ist es sehr wahrscheinlich, dass vor allem Fehler in der Bedienung der Software verborgen geblieben sind und in zukünftigen Tests der Mitarbeiter entdeckt werden.

Weniger ein direktes Problem als vielmehr eine noch ausstehende Arbeit ist die Überarbeitung des Quellcodes. Der Quellcode ist noch nicht ausreichend dokumentiert, und eine Überarbeitung der Programmierung erscheint bei nachträglicher Betrachtung an einigen Stellen sinnvoll. Vor allem vor dem Hintergrund der Freien Software Entwicklung ist es wichtig die Funktionsweise der einzelnen Programmteile besser zu dokumentieren und die Programmstruktur so deutlicher herauszuarbeiten, um die Verständlichkeit des Quellcodes für andere Entwickler zu verbessern. Idealerweise erfolgt dies durch eine schrittweise Refaktorisierung der Programmstruktur. Die Refaktorisierung beschreibt einen Prozess, in dem bestehender Code eines Software Systems modifiziert wird, ohne das externe Verhalten der Software zu verändern. Bei der Veränderung des Quellcodes findet eine Vereinfachung und Verbesserung der internen Datenstruktur statt(Fowler u. a., 1999).

[2] http://trac.irlaender.de/fast/

11.4 Ausblick

Als Ausblick soll hier ein Punkt angeführt werden, der aus zeitlichen Gründen leider nicht mehr den Weg in dieser Arbeit gefunden hat.

Mit der *Information Technology Infrastructure Library* (ITIL)(OGC, 2007) besteht eine Sammlung von bewährten Verfahren zur Einführung, Betrieb und Management einer IT-Infrastruktur und zugehöriger Dienstleistungen, die heute als defakto-Standard zur Abbildung von IT-Prozessen gilt. Mit diesen Leitfäden sollen die IT-Abteilungen in den Firmen in die Lage versetzt werden den stetig steigenden Anforderungen an die Qualität und Verfügbarkeit von IT-Diensten und Dienstleistungen besser gerecht zu werden. Für weitergehende Informationen zur ITIL wird stellvertretend für das reichhaltige Angebot an Literatur zur ITIL auf (Manfred Brandstätter, 2006) verwiesen. Ein großer Teilbereich der ITIL beschäftigt sich mit der Erbringung von Kundendienstleistungen, und ist somit auch für die Intevation GmbH potentiell interessant.

Trotz der noch relativ geringen Verbreitung in Deutschland gewinnt die ITIL an Bedeutung (Holzmüller u. a., 2003). Für Unternehmen besteht die Möglichkeit durch eine Zertifizierung zu belegen, das ihr Unternehmen ITIL-konform ist. Eine solche Zertifizierung ist oft die Voraussetzung bei der Vergabe von IT-Projekten durch öffentliche Stellen (Weiß, 2006). Vor dem Hintergrund, dass ein großer Teil der Kunden der Intevation GmbH aus dem öffentlicher Bereich stammen, stellt sich eine genauere Betrachtung der ITIL als lohnend dar.

In ersten Untersuchungen in dieser Arbeit deutete sich an, dass es offensichtliche Parallelen zwischen den empfohlenen Praktiken der ITIL und den Abläufen bei der Intevation GmbH gibt. Eine weitergehende Untersuchung dieser Parallelität mit dem Ziel einer Abschätzung des Aufwandes, der betrieben werden müsste um die Prozesse bei der Intevation GmbH ITIL-konform zu gestalten, erscheint daher sehr interessant.

Abbildungsverzeichnis

Anhang A

Trackervergleich

A.1 Bugzilla

Allgemeine Informationen	
Internet	www.bugzilla.org
Lizenz	MPL/GPL/LGPL
Version	2.23
Sprache	Perl
Dokumentation	++++
Fallfunktionen	
Suchfunktionen	✓
Verlaufsübersicht	✓
Emailbenachrichtigungen	✓
öffentliche und private Einträge	✓
Dateianhänge	✓
Fehlereskalation	✗
Alarmfunktionen	✗
Priorisieren	✓
Zuweisen von Bearbeitern	✓
Fehlerkategorien	✓
Priorisieren	✓
Fehlerstatus	✓
Verbindung zu anderen Fällen herstellen	✓
Erstellung von Abhängigkeiten	✓
Sonstige Funktionen	
Statistiken/Reports	✓
Export-Funktionen	csv, rdf
Wissensdatenbank	✗
Email-Verschlüsselung	✗
Meilensteine	✓
RSS-Feed	✓

Abspeichern von Abfragen	✓
zusätzl. Benutzerauthentifizierung	
Benutzerverwaltung	✓
Zeiterfassung	✓
Gruppenaktionen	✗
Erweiterbarkeit/Konfigurierbarkeit	
Anpassbar über Templates	✓
Neue Felder hinzufügbar	✓
Verhaltensänderung (Workflow)	✓
Erweiterbarkeit durch plugins	✓
Einbinden eines Wikis	✗
Einbinden anderer Tracker	✗
Einbinden von SCM-Systemen	über Erweiterung
Schnittstellen	
Webinterface	✓
Emailinterface	✓
Kommandozeile	✗
Datenspeicher	
textfiles	✗
anydb	✗
sqlite	✗
mysql	✓
postgressql	experimentell
mssql	✗
oracle	✗
sybase	✗
Bemerkung	
Keine	

Tabelle A.1: Funktionsübersicht des Bugzilla Bug-Trackers

A.2 OTRS

Allgemeine Informationen	
Internet	www.otrs.org
Lizenz	GPL
Version	2.1
Sprache	Perl
Dokumentation	+++
Fallfunktionen	
Suchfunktionen	✓
Verlaufsübersicht	✓
Emailbenachrichtigungen	✓
öffentliche und private Einträge	✓ (acl pro ticket)
Dateianhänge	✓
Fehlereskalation	✓
Alarmfunktionen	✓
Priorisieren	✓
Zuweisen von Bearbeitern	✓
Fehlerkategorien	✓
Priorisieren	✓
Fehlerstatus	✓
Verbindung zu anderen Fällen herstellen	✓
Erstellung von Abhängigkeiten	✓
Sonstige Funktionen	
Statistiken/Reports	✓
Export-Funktionen	pdf
Wissensdatenbank	✓
Email-Verschlüsselung	gpg, smime
Meilensteine	✗
RSS-Feed	✗
Abspeichern von Abfragen	✓
zusätzl. Benutzerauthentifizierung	LDAP, Datenbank, HTTPAuth, Radius
Benutzerverwaltung	Rollen, Benutzergruppen
Zeiterfassung	✓
Gruppenaktionen	✓
Erweiterbarkeit/Konfigurierbarkeit	
Anpassbar über Templates	✓
Neue Felder hinzufügbar	✓
Verhaltensänderung (Workflow)	✓
Erweiterbarkeit durch plugins	✗
Einbinden eines Wikis	✗
Einbinden anderer Tracker	✗

Einbinden von SCM-Systemen	✗
Schnittstellen	
Webinterface	✓
Emailinterface	✓
Kommandozeile	✗
Datenspeicher	
textfiles	✗
anydb	✗
sqlite	✗
mysql	✓
postgressql	✓
mssql	✓
oracle	✓
sybase	✗
Bemerkung	
ITIL-konform, Erweiterungen wie oben angegeben möglich, aber nicht trivial	

Tabelle A.2: Funktionsübersicht des OTRS Support-Trackers

A.3 Request Tracker 3

Allgemeine Informationen	
Internet	bestpractical.com/rt/
Lizenz	GPL
Version	3
Sprache	Perl
Dokumentation	++
Fallfunktionen	
Suchfunktionen	✓
Verlaufsübersicht	✓
Emailbenachrichtigungen	✓
öffentliche und private Einträge	✓
Dateianhänge	✓
Fehlereskalation	✓
Alarmfunktionen	über Eweiterung
Priorisieren	✓
Zuweisen von Bearbeitern	✓
Fehlerkategorien	✓
Priorisieren	✓
Fehlerstatus	✓
Verbindung zu anderen Fällen herstellen	✓
Erstellung von Abhängigkeiten	✓
Sonstige Funktionen	
Statistiken/Reports	über Eweiterung
Export-Funktionen	✗
Wissensdatenbank	über Erweiterung
Email-Verschlüsselung	✗
Meilensteine	✗
RSS-Feed	✗
Abspeichern von Abfragen	✓
zusätzl. Benutzerauthentifizierung	HTTPAuth
Benutzerverwaltung	✓
Zeiterfassung	✓
Gruppenaktionen	✗
Erweiterbarkeit/Konfigurierbarkeit	
Anpassbar über Templates	✓
Neue Felder hinzufügbar	✓
Verhaltensänderung (Workflow)	✓
Erweiterbarkeit durch plugins	✓
Einbinden eines Wikis	✗
Einbinden anderer Tracker	✗
Einbinden von SCM-Systemen	✗

Schnittstellen	
Webinterface	✓
Emailinterface	✓
Kommandozeile	✗
Datenspeicher	
textfiles	✗
anydb	✗
sqlite	✗
mysql	✓
postgressql	✓
mssql	✗
oracle	✓
sybase	in Arbeit
Bemerkung	
Keine	

Tabelle A.3: Funktionsübersicht des RT3 Support-Trackers

A.4 Roundup

Allgemeine Informationen	
Internet	roundup.sourceforge.net
Lizenz	GPL
Version	1.3.1
Sprache	Python
Dokumentation	+++++
Fallfunktionen	
Suchfunktionen	✓
Verlaufsübersicht	✓
Emailbenachrichtigungen	✓
öffentliche und private Einträge	✗
Dateianhänge	✓
Fehlereskalation	✗
Alarmfunktionen	✗
Priorisieren	✓
Zuweisen von Bearbeitern	✓
Fehlerkategorien	✓
Priorisieren	✓
Fehlerstatus	✓
Verbindung zu anderen Fällen herstellen	✓
Erstellung von Abhängigkeiten	✗
Sonstige Funktionen	
Statistiken/Reports	über Erweiterung
Export-Funktionen	roundup-spezifisch
Wissensdatenbank	✗
Email-Verschlüsselung	✗
Meilensteine	✗
RSS-Feed	über Erweiterung
Abspeichern von Abfragen	✓
zusätzl. Benutzerauthentifizierung	über Erweiterung: LDAP, e-directory, NT-Login
Benutzerverwaltung	Rollenbasiert
Zeiterfassung	über Erweiterung
Gruppenaktionen	über Erweiterung
Erweiterbarkeit/Konfigurierbarkeit	
Anpassbar über Templates	✓
Neue Felder hinzufügbar	✓
Verhaltensänderung (Workflow)	✓
Erweiterbarkeit durch plugins	✓
Einbinden eines Wikis	✗
Einbinden anderer Tracker	✗

Einbinden von SCM-Systemen	über plugin
Schnittstellen	
Webinterface	✓
Emailinterface	✓
Kommandozeile	✓
Datenspeicher	
textfiles	✓
anydb	✓
sqlite	✓
mysql	✓
postgressql	✓
mssql	✗
oracle	✗
sybase	✗
Bemerkung	
Sehr flexibel und erweiterbar	

Tabelle A.4: Funktionsübersicht des Roundup Issue-Trackers

A.5 Mantis

Allgemeine Informationen	
Internet	www.mantisbugtracker.com
Lizenz	GPL
Version	1.0.6
Sprache	PHP
Dokumentation	+++++
Fallfunktionen	
Suchfunktionen	✓
Verlaufsübersicht	✓
Emailbenachrichtigungen	✓
öffentliche und private Einträge	✗
Dateianhänge	✓
Fehlereskalation	✗
Alarmfunktionen	✗
Priorisieren	✓
Zuweisen von Bearbeitern	✓
Fehlerkategorien	✓
Priorisieren	✓
Fehlerstatus	✓
Verbindung zu anderen Fällen herstellen	✓
Erstellung von Abhängigkeiten	✓
Sonstige Funktionen	
Statistiken/Reports	✓
Export-Funktionen	MS Word, MS Excel, csv
Wissensdatenbank	✗
Email-Verschlüsselung	✗
Meilensteine	✓
RSS-Feed	✓
Abspeichern von Abfragen	✗
zusätzl. Benutzerauthentifizierung	LDAP, e-directory, HTTPAuth
Benutzerverwaltung	✓
Zeiterfassung	✗
Gruppenaktionen	✓
Erweiterbarkeit/Konfigurierbarkeit	
Anpassbar über Templates	✗
Neue Felder hinzufügbar	✓
Verhaltensänderung (Workflow)	✓
Erweiterbarkeit durch plugins	✓ (hooks)
Einbinden eines Wikis	optional
Einbinden anderer Tracker	✗

Einbinden von SCM-Systemen	cvs,svn
Schnittstellen	
Webinterface	✓
Emailinterface	✓
Kommandozeile	✗
Datenspeicher	
textfiles	✗
anydb	✗
sqlite	✗
mysql	✓
postgressql	✓
mssql	✓
oracle	experimentell
sybase	in Arbeit
Bemerkung	
Ausschreibungsfunktion verfügbar	

Tabelle A.5: Funktionsübersicht des Bug-Trackers

Anhang B

Rollenbasierte Befugnisse

B.1 Kunde

Issue	Lesen	Bearbeiten	Erstellen
Titel	✓	✗	✓
Kundengruppe	✓	✗	✗
Kundenkontakt	✓	✓	✗
Partnergruppe	✓	✗	✗
Partnerkontakt	✓	✗	✗
Interngruppe	✓	✗	✗
Internkontakt	✓	✗	✗
Priorität	✓	✓	✓
Art	✓	✗	✓
Deadline	✓	✗	✗
Status	✓	✓	✗
Ersetzt durch	✓	✗	✗
Interessenten	✓	✗	✗
Zugewiesen	✓	✓	✗
Schlagwörter	✓	✗	✗
Arbeitspakete	✗	✗	✗
Zeiten bewerten	✗	✗	✗
Remote Event	✓	✗	✗
Nachricht	Lesen	Bearbeiten	Erstellen
Text	✓	✗	✓
Timelog	✗	✗	✗
View	✓	✓	✗
Dateianhänge	✓	✓	✓
Remote Event	Lesen	Bearbeiten	Erstellen
Backend	✓	✗	✗
Connector	✓	✗	✗
Url	✓	✗	✗

Anklopfen Intervall	✓	✗	✗
Nächstes Anklopfen	✓	✗	✗
Alarm	✓	✗	✗
Enabled	✓	✗	✗
Arbeitspakete	Lesen	Bearbeiten	Erstellen
Titel	✗	✗	✗
Aufgaben	✗	✗	✗
Zusammenfassung	✗	✗	✗
Notizen	✗	✗	✗
Empfänger	✗	✗	✗
Zeiten zuweisen	✗	✗	✗

Tabelle B.1: Befugnisse der “Customer” Rolle

B.2 Partner

Issue	Lesen	Bearbeiten	Erstellen
Titel	✓	✗	✗
Kundengruppe	✓	✗	✗
Kundenkontakt	✓	✗	✗
Partnergruppe	✓	✗	✗
Partnerkontakt	✓	✓	✗
Interngruppe	✓	✗	✗
Internkontakt	✓	✗	✗
Priorität	✓	✗	✗
Art	✓	✗	✗
Deadline	✓	✗	✗
Status	✓	✓	✗
Ersetzt durch	✓	✓	✗
Interessenten	✓	✓	✗
Zugewiesen	✓	✓	✗
Schlagwörter	✓	✓	✗
Arbeitspakete	✗	✗	✗
Zeiten bewerten	✗	✗	✗
Remote Event	✓	✓	✓
Nachricht	Lesen	Bearbeiten	Erstellen
Text	✓	✗	✓
Timelog	✓	✓	✓
View	✓	✓	✗
Dateianhänge	✓	✓	✓
Remote Event	Lesen	Bearbeiten	Erstellen
Backend	✓	✗	✓
Connector	✓	✗	✓
Url	✓	✗	✓
Anklopfen Intervall	✓	✓	✓
Nächstes Anklopfen	✓	✓	✓
Alarm	✓	✓	✓
Enabled	✓	✓	✓
Arbeitspakete	Lesen	Bearbeiten	Erstellen
Titel	✗	✗	✗
Aufgaben	✗	✗	✗
Zusammenfassung	✗	✗	✗
Notizen	✗	✗	✗
Empfänger	✗	✗	✗
Zeiten zuweisen	✗	✗	✗

Tabelle B.2: Befugnisse der “Partner” Rolle

B.3 Mitarbeiter

Issue	Lesen	Bearbeiten	Erstellen
Titel	✓	✓	✓
Kundengruppe	✓	✓	✓
Kundenkontakt	✓	✓	✓
Partnergruppe	✓	✓	✓
Partnerkontakt	✓	✓	✓
Interngruppe	✓	✓	✓
Internkontakt	✓	✓	✓
Priorität	✓	✓	✓
Art	✓	✓	✓
Deadline	✓	✓	✓
Status	✓	✓	✓
Ersetzt durch	✓	✓	✓
Interessenten	✓	✓	✓
Zugewiesen	✓	✓	✓
Schlagwörter	✓	✓	✓
Arbeitspakete	✓	✓	✓
Zeiten bewerten	✓	✓	✓
Remote Event	✓	✓	✓
Nachricht	Lesen	Bearbeiten	Erstellen
Text	✓	✓	✓
Timelog	✓	✓	✓
View	✓	✓	✗
Dateianhänge	✓	✓	✓
Remote Event	Lesen	Bearbeiten	Erstellen
Backend	✓	✓	✓
Connector	✓	✓	✓
Url	✓	✓	✓
Anklopfen Intervall	✓	✓	✓
Nächstes Anklopfen	✓	✓	✓
Alarm	✓	✓	✓
Enabled	✓	✓	✓
Arbeitspakete	Lesen	Bearbeiten	Erstellen
Titel	✗	✗	✗
Aufgaben	✗	✗	✗
Zusammenfassung	✗	✗	✗
Notizen	✗	✗	✗
Empfänger	✗	✗	✗
Zeiten zuweisen	✗	✗	✗

Tabelle B.3: Befugnisse der “Internal” Rolle

B.4 Manager

Issue	Lesen	Bearbeiten	Erstellen
Titel	✗	✗	✗
Kundengruppe	✗	✗	✗
Kundenkontakt	✗	✗	✗
Partnergruppe	✗	✗	✗
Partnerkontakt	✗	✗	✗
Interngruppe	✗	✗	✗
Internkontakt	✗	✗	✗
Priorität	✗	✗	✗
Art	✗	✗	✗
Deadline	✗	✗	✗
Status	✗	✗	✗
Ersetzt durch	✗	✗	✗
Interessenten	✗	✗	✗
Zugewiesen	✗	✗	✗
Schlagwörter	✗	✗	✗
Arbeitspakete	✓	✓	✗
Zeiten bewerten	✓	✓	✗
Remote Event	✗	✗	✗
Nachricht	Lesen	Bearbeiten	Erstellen
Text	✗	✗	✗
Timelog	✗	✗	✗
View	✗	✗	✗
Dateianhänge	✗	✗	✗
Remote Event	Lesen	Bearbeiten	Erstellen
Backend	✗	✗	✗
Connector	✗	✗	✗
Url	✗	✗	✗
Anklopfen Intervall	✗	✗	✗
Nächstes Anklopfen	✗	✗	✗
Alarm	✗	✗	✗
Enabled	✗	✗	✗
Arbeitspakete	Lesen	Bearbeiten	Erstellen
Titel	✓	✓	✓
Aufgaben	✓	✓	✓
Zusammenfassung	✓	✓	✓
Notizen	✓	✓	✓
Empfänger	✓	✓	✓
Zeiten zuweisen	✓	✓	✓

Tabelle B.4: Befugnisse der “Manager” Rolle

Literaturverzeichnis

[Balzert 1997] Balzert, Helmut: *Lehrbuch der Software-Technik, Bd.2, Software-Management, Software-Qualitätssicherung und Unternehmensmodellierung, m. CD-ROM.* Spektrum Akademischer Verlag, 1997. – ISBN 3827400651

[Biermann 2004] Biermann, Jürgen: Objektorientierte Analyse und Design. In: *Vorlesungsscript WS04 Fachhochschule Osnabrück* (2004)

[BPS 2007] BPS, Best Practical S.: Best Practical Solutions, LLC: RT: Request Tracker. (2007). `http://www.bestpractical.com/rt/praise.html`, Abruf: 16. März. 2006

[Ernst 2003] Ernst, Hartmut: *Grundkurs Informatik. Grundlagen und Konzepte für die erfolgreiche IT-Praxis.* Vieweg, 2003. – ISBN 3528257172

[Fowler u. a. 1999] Fowler, Martin ; Beck, Kent ; Brant, John ; Opdyke, William ; Roberts, Don: *Refactoring: Improving the Design of Existing Code.* Addison-Wesley Professional, 1999. – ISBN 0201485672

[FSF 1992] FSF, Free Software F.: GNU GENERAL PUBLIC LICENSE Version 2. In: *GNU Project Webseite* (1992). `http://www.gnu.org/licenses/gpl.txt`, Abruf: 07. März. 2007

[Gamma u. a. 1995] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series).* Addison-Wesley Professional, 1995. – ISBN 0201633612

[Grassmuck 2002] Grassmuck, Volker: *Freie Software.* Bundeszentrale für politische Bildung, 2002. – ISBN 3893314326

[Holzmüller u. a. 2003] Holzmüller, Hartmut H. ; Lammerts, Arno ; Stolper, Markus: *ITIL - Status und Trends in Deutschland.* `http://www.it-surveys.de/itsurvey/pages/studie4/studie4_01.html`. Version: 2003, Abruf: 07. März. 2006

[Jones 2006a] Jones, Richard: Customising Roundup. In: *Roundup Online Documentation* (2006). `http://roundup.sourceforge.net/doc-1.0/customizing.html`, Abruf: 19. März. 2007

[Jones 2006b] Jones, Richard: Installing Roundup. In: *Roundup Online Documentation* (2006). `http://roundup.sourceforge.net/doc-1.0/installation.html#installation`, Abruf: 24.April. 2007

[Jones 2006c] Jones, Richard: Roundup Documentation. In: *Roundup Online Documentation* (2006). `http://roundup.sourceforge.net/doc-1.0/`, Abruf: 19. März. 2007

[Jones 2006d] Jones, Richard: Upgrading Roundup. In: *Roundup Online Documentation* (2006). `http://roundup.sourceforge.net/doc-1.0/upgrading.html`, Abruf: 25.April. 2007

[Lutz u. Ascher 2003] Lutz, Mark ; Ascher, David: *Learning Python, Second Edition*. O'Reilly Media, 2003. – ISBN 0596002815

[Manfred Brandstätter 2006] Manfred Brandstätter, Thomas P.: Open-ITIL - ein Ansatz zur Akzeptanz- Verstärkung für den Einsatz von IT-Service Management nach ITIL in Klein- und Mittelunternehmen. (2006). `http://www.lisog.org/`, Abruf: 28. März. 2007

[OGC 2007] OGC: OGC - IT Infrastructure Library (ITIL). (2007). `http://www.itil.co.uk/`, Abruf: 24. März. 2007

[OSI 2000] OSI, Open Soure I.: How is "open source" related to "free software"? (2000). `http://www.opensource.org/advocacy/faq.php`, Abruf: 11. März. 2007

[Prechelt 2000] Prechelt, Lutz: An Empirical Comparison of Seven Programming Languages. In: *IEEE Computer* (2000). `http://page.mi.fu-berlin.de/~prechelt/Biblio/jccpprt_computer2000.pdf`, Abruf: 25. April. 2006

[Probst 2004] Probst, Erwin: *Entwurf einer Trouble-Ticket-Struktur als Grundlage zur Durchführung des Problemmanagements eines verteilten Systems durch einen externen Service-Anbieter*. 2004

[Raymond 2000] Raymond, Eric S.: Why Python? In: *Linux Journal* (2000). `http://www.linuxjournal.com/article/3882`, Abruf: 06. März. 2006

[Raymond 2001] Raymond, Eric S.: *The Cathedral and the Bazaar*. O'Reilly, 2001. – ISBN 0596001088

[rfc1297 1992] rfc1297, Network Working G.: NOC Internal Integrated Trouble Ticket System Functional Specification Wishlist. In: *Request for*

Comments RfC1297 (1992). `http://ietf.org/rfc/rfc1297.txt`, Abruf: 11. März. 2007

[van Rossum 1991] Rossum, Guido van: The Python Programming Language. (1991). `http://www.python.org`, Abruf: 06. März. 2007

[Stallman 1983] Stallman, Richard: The GNU Manifesto. In: *GNU Project Webseite* (1983). `http://www.gnu.org/gnu/manifesto.html`, Abruf: 04. Mai. 2006

[Stallman 1985] Stallman, Richard: The Free Software Definition. In: *GNU Project Webseite* (1985). `http://www.gnu.org/philosophy/free-sw.html`, Abruf: 04. Mai. 2006

[Trummer 2004] Trummer, Christian: *Adaptierbarkeit der Hyperwave Information Server Applikation "Support Tracking" System.* 2004

[Users 2007] Users, Wikipedia: Open Ticket Requiest System. (2007). `http://de.wikipedia.org/wiki/Open_Ticket_Request_System`, Abruf: 16. März. 2006

[Weiß 2006] Weiss, Oliver: ITIL-Zertifizierung für Open Source-Software. In: *Nachrichtenbeitrag der ComputerweltAT* (2006). `http://www.computerwelt.at/detailArticle.asp?a=101379`, Abruf: 06. Mai. 2007

[Yee 2004] Yee, Ka-Ping: Roundup An Issue-Tracking System for Knowledge Workers. An Implementation Guide. (2004). `http://zesty.ca/sc-roundup.html`, Abruf: 16. März. 2007

[Zeller 2005] Zeller, Andreas: *Why Programs Fail: A Guide to Systematic Debugging.* Morgan Kaufmann, 2005. – ISBN 1558608664

Commons Deed

Namensnennung-Weitergabe unter gleichen Bedingungen 2.0 Deutschland

Sie dürfen:


den Inhalt vervielfältigen, verbreiten und öffentlich aufführen


Bearbeitungen anfertigen

Zu den folgenden Bedingungen:


Namensnennung. Sie müssen den Namen des Autors/Rechtsinhabers nennen.


Weitergabe unter gleichen Bedingungen. Wenn Sie diesen Inhalt bearbeiten oder in anderer Weise umgestalten, verändern oder als Grundlage für einen anderen Inhalt verwenden, dann dürfen Sie den neu entstandenen Inhalt nur unter Verwendung identischer Lizenzbedingungen weitergeben.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter die dieser Inhalt fällt, mitteilen.
- Jede dieser Bedingungen kann nach schriftlicher Einwilligung des Rechtsinhabers aufgehoben werden.

Die gesetzlichen Schranken des Urheberrechts bleiben hiervon unberührt. Das Commons Deed ist eine Zusammenfassung des Lizenzvertrags in allgemeinverständlicher Sprache. Um den Lizenzvertrag einzusehen, besuchen Sie die Seite

`http://creativecommons.org/licenses/by-sa/2.0/de`

oder senden Sie einen Brief an Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Osnabrück, 10. Mai 2007

_____

Unterschrift