Institut für Softwaretechnologie
Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Fachstudie Nr. 181

Optimierung eines Workflows für Softwareentwickler bei der Bearbeitung von Arbeitspaketen

Andreas Rempel, Kai Friedrich und Andreas Gross

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. rer. nat. Stefan Wagner
Betreuer:	Jasmin Ramadani, M.Sc.
begonnen am:	10.06.2013
beendet am:	10.12.2013
CR-Klassifikation:	D.2.9

Zusammenfassung

Bei der Entstehung von Software ist es unabdingbar entstehende Aufgaben über mehrere Arbeitspakete aufzuteilen. Dabei ist es wichtig ein Arbeitspaket während seines gesamten Lebenszyklus zu kontrollieren und dokumentieren. Dieser Lebenszyklus muss auf das Unternehmen zugeschnitten sein, um eine hohe Effizienz und Qualität bei der Bearbeitung des Arbeitspaketes zu gewährleisten. Um die erfolgreiche Durchführung zu unterstützen, will der Industriepartner ein Issue-Tracking-System einsetzen in dem dieser Lebenszyklus abgebildet werden kann. Im Auftrag des Industriepartners wurde dessen bestehender Workflow analysiert und für die Abbildung in einem Issue-Tracking-System optimiert. Hierzu werden in dieser Fachstudie 12 Issue-Tracking-Systeme evaluiert.

Abstract

The development of software requires dividing existing tasks into work packages. It is important to track and document a work package throughout its life cycle. This life cycle has to be tailored to the particular needs of the business to ensure high efficiency and high quality during the process. The industry partner would like to introduce an issue tracking system to support this process which maps this life cycle. The work flow has been analysed and optimized to be mapped to an issue tracking system on behalf of the industry partner. 12 issue tracking systems have been evaluated for this purpose.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In jedem Unternehmen, das heute mit Software arbeitet ist es wichtig die daraus resultierenden Projekte zu verwalten, zu dokumentieren und zu archivieren. Zudem sind diese Methoden ab einer bestimmten Unternehmensgröße bzw. Größe der Softwareabteilung unverzichtbar, ebenso einen definierten Prozess einzuführen und diesen auch anzuwenden. Jedoch müssen Projekte, die nach einem solchen Prozess durchgeführt werden, verwaltet, überwacht und notfalls korrigiert werden. An dieser Stelle werden die sogenannten “Ticketing-Systeme”eingesetzt. Je nach Anforderung des Prozesses und Unternehmens kann entweder eine einfache Ticketliste, also einzelne erteilte Teilaufgaben, oder auch eine Anbindung an bestehende Content-Management-Systeme und Entwicklungsumgebungen reichen. Bei der Ticketerstellung kann man entscheiden, ob man dem Kunden selbst ermöglicht Tickets direkt zu erstellen oder die Erstellung lediglich über zugewiesene Ansprechpartner des eigenen Unternehmens weitergeleitet und in das Ticketsystem eingepflegt werden.

Folglich sind zuerst einige grundlegende Thematiken zu erörtern und zu bewerten, bevor man die Anforderungen des eigenen Unternehmens und die Befriedigung dessen Anforderungen erörtern kann. Der Markt bietet heutzutage eine Vielfallt an solchen ”Ticketing-Systemen”. Einige davon werden als kostenfreie Open-Source Lösung angeboten, die meistens jedoch viel eigenen Aufwand in der Verwaltung und Sicherung fordern, was wiederum hohe indirekte Kosten verursacht. Andererseits gibt es kommerzielle Lösungen, die jedoch schnell einen nicht unerheblichen Kostenaspekt in den Vordergrund rücken lassen, sich aber meist durch die effiziente und qualitative Nutzbarkeit auszeichnen.

Zusammenfassend sollte man bei der Wahl darauf achten, was man tatsächlich benötigt, vor allem im Hinblick auf den Ist-Stand, der Erweiterungsmöglichkeiten des Systems und der allgemeinen Entwicklungsrichtung beziehungsweise des Entwicklungsfortschritts des eigenen Unternehmens.

1.1 Begriffe

Dieses Kapitel dient der Erklärung der Fachterminologie. Die Gültigkeit der Definitionen beschränkt sich nur dieses Dokument. Dieses Kapitel soll vor allem für das einheitliche Verständnis und Interpretation der in dieser Arbeit behandelten Thematik sorgen.

Incident Ein Incident beschreibt ein Anliegen jeglicher Art und ist nicht nur auf Software beschränkt. Es muss jedoch in einer vorher festgelegten Form definiert und eingereicht werden. Im Softwarebereich kann ein Incident einen Change Request, einen Bugreport oder jegliche Anliegen im Zusammenhang mit der verwalteten Software sein.

Software Problem Report Ist ein Bericht, welcher das Auftretten eines Fehlverhaltens bei der Bedienung einer Software beschreibt. Dieser ist jedoch nicht qualifiziert und schränkt die Ursachen für das Auftretten des Fehlers nicht weiter ein. Zudem ist diese auch frei von Vorschlägen der Problemlösung.
Synonyme: *Issue*

Change Request Ein Change Request beinhaltet eine Anfrage einer Systemanpassung. Es ist von einer erheblichen Wichtigkeit im Änderungsmanagementprozess. Ein Change Request ist deklarativ, zum Beispiel beschreibt er was durchgeführt werden soll, jedoch schreibt er nicht vor, wie die Aufgaben durchzuführen sind.
Synonyme: *Änderungsanforderung*

(Sofware-) Issue-Tracking-System Ist ein System, das unter anderem ein Programm zum Verwalten und Überwachen von Aufgaben, die im Rahmen eines Softwarelebenszyklus anfallen, enthält und den Fortschritt dieser Aufgaben dokumentiert.

Ticket Ein Ticket ist die graphische Darstellung und Zuweisung eines Issues an einen Zuständigen. Ferner dient ein Ticket der Strukturierung und Verwaltung eines Softwarelebenszyklus.
Es beginnt bei der Entwicklung, in welcher das Softwareprojekt geplant wird und ist darüber hinaus auch bei der Wartung und Erweiterung von bereits in Betrieb genommener Software hilfreich.
Es kann auch in untergeordnete Tickets aufgeteilt werden, wenn es sich um ein komplexes Problem oder eine langwierige Aufgabe handelt. Ein Ticket ist auch Bestandteil eines Issue-Tracking-Systems.

Workflow Ein Workflow bezeichnet mehrere dynamische, abteilungsübergreifende aber fachlich zusammenhängende, arbeitsteilige Aktivitäten, die in logischer oder zeitlicher Abhängigkeit zueinander stehen. Ein Workflow ist die informationstechnische Realisierung eines Geschäftsprozesses [2].

Produktverantwortlicher Der Produktverantwortliche kümmert sich um die Entwicklung und Wartung einer Software. Ein Produktverantwortlicher muss nicht unbedingt für ein einziges Produkt verantwortlich sein, sondern kann auch mehrere Produkte betreuen.

(Software-) Configuration Management Das Configuration Management beginnt mit der Entwicklung der Software und endet erst, wenn die Software außer Betrieb genommen wird. Es ist ein essentieller Teil eines guten Projektmanagements und solider Anwendung von Software-Engineering-Praktiken. (Software-) Configuration Management bietet einen Mechanismus um Versionen jeder Softwarekomponente zu identifizieren, kontrollieren und zu verfolgen. In vielen Fällen müssen ältere Versionen die noch in Betrieb sind gewartet und überwacht werden.

Incident Management Das Incident Management beschäftigt sich mit der Verwaltung von Incidents. Dazu gehört das Weiterleiten an die richtigen Ansprechpartner und die Verfolgung des Fortschritts jedes einzelnen Incidents, sowie das weiterleiten der Information an das gesamte Team. Sobald ein Incident abgearbeitet bzw. gelöst ist, wird es geschlossen und archiviert.

Dispatcher Der Dispatcher ist eine Rolle im Geschäftsprozess. Er ist die erste Instanz, der gemeldete Incidents bearbeitet. Er leitet sie bei Bedarf an die richtigen Ansprechpartner oder Abteilungen weiter. Werden Vorfälle mehrfach gemeldet, so ist es die Aufgabe des Dispatchers diese miteinander zu verknüpfen, oder auf vorliegende Lösungen zu verweisen.

Task Ein Task ist eine logisch zusammenhängende Aufgabe, die als Ziel einen Bugfix, eine Weiterentwicklung oder ein Softwareartefakt hat. Falls ein Task zu umfangreich wird, sodass dieser sich in mehrere Teilaufgaben gliedert, so werden aus diesen Teilaufgaben Subtasks.

Softwareprodukt Ein Softwareprodukt beschreibt in diesem Fall ein Softwareartefakt mit der dazugehörigen Dokumentation. Ein Softwareprodukt kann für sich vertrieben, jedoch nicht zwangsweise auch eigenständig verwendet, bzw. ausgeführt werden.

Dashboard Das Dashboard ist eine Startseite eines Issue-Tracking-Systems, das Informationen meist in graphischer und kompakter Form Repräsentiert. Meistens ist es in verschieden Bereiche eingeteilt und besteht aus mehreren visuellen Komponenten.

User Story User Storys dienen überwiegend in agilen Softwareprozessen der Anforderungserhebung. Sie umschreiben in kurzer Form die Ansprüche eines Benutzers an das System.

Shortcut Ein Shortcut ist eine Tastenkombination entweder gleichzeitig oder aufeinander folgend gedrückter Tasten, die ein Aktion anstoßen. Aber auch einzelne Tasten, wie z.B. die Funktionstasten, können einen Shortcut bilden, indem sie eine definierte Funktion durchführen oder eine Aktion anstoßen.
Synonyme: Tastaturbefehl, Tastaturkürzel, Tastenkombination

Produkt-Backlog Das Produkt-Backlog ist die Sammlung von Anforderungen an ein Softwareprodukt und den Umfang, den dieses Produkt haben soll. Es kommt überwiegend in agilen Softwareprozessen zum Einsatz. Im Laufe des Produktlebenszyklus wird das Produkt-Backlog ergänzt und erhebt nicht den Anspruch vollständig zu sein.

Burndown-Report Ein Burndown-Report enthält Informationen über den Stand der Entwicklung in einer zeitlich beschränkten Phase. Es wird üblicherweise die Gesamtanzahl der Aufgaben, die Anzahl der Aufgaben, die bereits abgeschlossen, in Bearbeitung, oder noch zu bearbeiten sind. Oft enthält ein Burndown-Report auch ein Diagramm, das den Entwicklungsfortschritt der jeweiligen Phase visuell darstellt.

Helpdesk Ein Helpdesk ist ein Teil eines Issue-Tracking-Systems, der in erster Linie dazu dient Benutzern von Software bei auftretenden Komplikationen und aufkommenden Fragen zu unterstützen. Er dient als erste Anlaufstelle für die Benutzer von Software und deckt in manchen Fällen auch Anfragen für andere Dienstleistungsbereiche ab.

Task-Author Task-Author ist die Rolle, die für die Erstellung von Tasks und die Überwachung des Fortschritts dieser Tasks verantwortlich ist. Zudem sind die Personen, die diese Rolle innehalten Ansprechpartner für die Anforderungen und den Verlauf eines Tasks. Der Task-Author unterstützt die Task-Owner bei Schwierigkeiten oder Unklarheiten in den Beschreibungen der erstellten Tasks, oder bei Entscheidungen zur Umsetzung der im Task beschriebenen Aufgaben.

Task-Owner Task-Owner ist die Rolle, die für die Umsetzung der in einem Task vom Task-Author beschriebenen Aufgaben verantwortlich ist. Zudem gehört zu den Aufgaben eines Task-Owners das Unterstützen eines Task-Authors bei der Anpassung der Beschreibungen von Tasks, die der Task-Owner eigenständig oder im Team bearbeitet. Ferner schätzt der Task-Owner Aufwände und Risiken eines Tasks ein und kommuniziert diese an den Task-Author. Der Task-Owner ist auch dafür verantwortlich in vom Task-Author festgelegten Zyklen den Status des zu bearbeitenden Tasks mit dem Task-Author zu diskutieren.

Deployment "deployment - that is, the activities related to the release, installation, activation, deactivation, update and removal of components, as well as whole systems."[1]
Synonyme: *Software Deployment*

1.2 Ziel

Ziel ist es den Task-Workflow der Aufgabenverteilung an die Software-Entwickler zu optimieren und ein entsprechendes Issue-Tracking-System vorzuschlagen, das diesen Task-Workflow exakt abbilden kann. Dabei werden die Anforderungen des Industriepartners berücksichtigt und dienen als Bewertungskriterien bei der Evaluation der Issue-Tracking-Systeme.

2 Ablauf der Fachstudie

Im Folgenden wird ein Überblick über den zeitlichen Ablauf dieser Arbeit vermittelt. Die Arbeit an der Fachstudie begann am 10. Juni 2013 in einem Büro des Industriepartners. Hierbei wurden uns grobe Zusammenhänge der unternehmensinternen Abteilungen vorgestellt und die Kontaktpersonen der jeweiligen Abteilungen genannt. Es wurden uns Vorstellungen bezüglich des Ablaufs der Fachstudie von den uns zugeteilten Betreuern und Ansprechpartnern des Industriepartners mitgeteilt. Anschließend wurde gemeinsam ein grober Umriss des Vorgehens der Fachstudie erörtert.

2.1 Phasen

Der gesamte Ablauf der Fachstudie ist in kleinere zeitliche Abschnitte unterteilt, die wir als Phasen bezeichnen. Zum Abschluss jeder Phase liegen ein oder mehrere Ergebnisse in Form von Dokumenten oder in einer anderen greifbaren Form vor, z.B. eine Präsentation inklusive Folien und eventuell anderen Unterlagen.

Nachfolgend werden nun die einzelnen Phasen beschrieben. Die Reihenfolge und der zeitliche Rahmen kann dem Gantt-Diagramm am Schluss dieses Abschnittes entnommen werden.

2.1.1 Ausarbeitung des Fragenkatalog

In dieser Phase wurde ein Projektplan angefertigt, der den zeitlichen Ablauf und die Phasen mit den damit verbundenen Aufgaben und Artefakten beschreibt. Zur Ist- und Soll-Analyse wurde von uns ein Fragenkatalog erstellt. Der Fragekatalog sollte uns dabei unterstützen, gezielt Informationen von den Kontaktpersonen, die größtenteils verschiedene Aufgaben- und Zuständigkeitsbereiche haben, zu beschaffen und um die Ergebnisse strukturiert bzw. quantifiziert auswerten zu können. Dieser Fragenkatalog wurde mit dem Betreuer des Industriepartners abgenommen.

2.1.2 Durchführung der Befragung

Diese Phase hatte als Kernaufgabe die Befragungen mit den Kontaktpersonen durchzuführen. Hierbei wurde der Fragenkatalog verwendet, um eine Umfrage mit dem Umfragewerkzeug LimeSurvey zu erstellen. Die genaue Durchführung und die Ergebnisse der Umfrage werden in Kapitel 4 näher erläutert.

Ferner wurden in dieser Phase erste „Best-Practices Ticketing“ recherchiert, die uns bei der Optimierung des Workflows als Referenzen dienen sollten.

2.1.3 Aufstellung der Anforderungen

Um die Anforderungen zu identifizieren, wurden die Ergebnisse der Umfrage von uns ausgewertet. Daraufhin wurden zusätzlich einzelne Kontaktpersonen im Unternehmen in einem Interview befragt. Außerdem wurde in dieser Phase eine Zwischenpräsentation unserer bisherigen Ergebnisse gehalten. Abschließend wurden Kriterienkatalog anhand der identifizierten Anforderungen erstellt.

2.1.4 Workflow Analyse

In dieser Phase wurde der bestehende Workflow des Industriepartners analysiert. Dafür wurden ebenfalls die Ergebnisse der Umfrage herangezogen und Interviews mit einzelnen Kontaktpersonen im Unternehmen durchgeführt. Abschließend wurden eine mögliche Optimierungen von uns ausgearbeitet.

2.1.5 Tool Analyse und Einsatzempfehlung

In dieser Phase wurden Tools zur Ticketverwaltung anhand des Bekanntheitsgrades und Erfahrungen ausgewählt. Zu dem Kriterienkatalog wurde ein Bewertungsschema erstellt. Die ausgewählten Tools zur Ticketverwaltung wurden mithilfe des Kriterienkatalogs evaluiert.

Nach Abschluss der Evaluierung wurde dem Unternehmen die Bewertung der Tools zur Ticketverwaltung und mögliche Optimierungen des aktuellen Workflows mitgeteilt.

2.1.6 Abgabe der Fachstudie

In dieser abschließende Phase wurde ein Bericht der Fachstudie erstellt und diese im Sekreteriat des zuständigen Prüfungsausschusses eingereicht.

2.2 Zeitliche Planung

Das nachfolgende Gantt-Diagramm zeigt den zeitlichen Ablauf der zuvor beschriebenen Phasen.

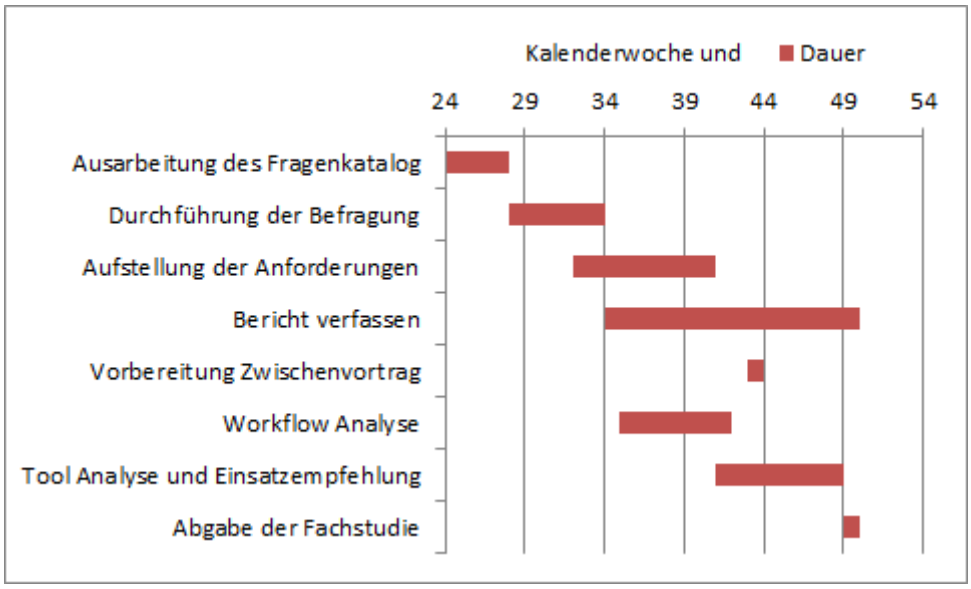

Abbildung 1: Zeitplanung

3 Voraussetzungen für erfolgreiches Änderungsmanagement

3.1 Ein definierter Änderungsmanagement-Prozess

Damit eine effiziente Bearbeitung von Problemmeldungen erfolgen kann, muss ein definierter Prozess existieren. Abblidung 2 auf Seite 16 zeigt einen solchen Prozess auf, der unserer Meinung nach sehr gut widerspiegelt, welche Aspekte jeder effiziente Änderungsmanagement-Prozess mindestens abdecken muss. Diese Aspekte werden im Folgenden kurz beschrieben.

Analyse von Problemmeldungen Problemmeldungen dürfen nicht ohne vorhergehende Überprüfung an Entwickler als Arbeitspakete weitergegeben werden. Problemmeldungen müssen vorgefiltert werden. Unvollständig beschriebene Problemmeldungen müssen an den Ersteller zurückgegeben werden. Zudem muss geprüft werden, ob eine Änderung notwendig ist oder ob es sich etwa nur um einen Bedienfehler seitens eines Benutzers handelt. Zudem muss geprüft werden, ob es sich bei einer Problemmeldung um ein Duplikat handelt, d.h. dass für das selbe Problem bereits eine Software-Änderung in Auftrag gegeben wurde. Ohne eine solche Analyse läuft man Gefahr, dass Änderungen unter Umständen doppelt beauftragt werden oder die Code-Base der Software mit unnötigen oder sogar fehlerhaften Änderungen verunreinigt wird.

Kontrollierte Durchführung von Änderungen an Software Wenn eine Problemmeldung ein echtes Problem repräsentiert, d.h. die Lösung erfordert eine Fehlerkorrektur oder Erweiterung der Software, muss der Aufwand für die Durchführung der Änderung, sowie Folgen und Risiken der Änderung abgeschätzt werden und anschließend muss eine Instanz (CCB) die Durchführung der Änderung freigeben. Falls sich herausstellt, dass eine Änderung unverhältnismäßig hohe Kosten oder Risiken nach sich zieht, kann das CCB die Änderung auch ablehnen, sofern keine Rechtsbindung durch Wartungsverträge besteht. In letzterem Fall kann man dem Kunden eventuell auch einen Workaround anbieten. Falls die Änderung nur geringen Aufwand benötigt, kann nach einer Einschätzung die Änderung direkt ohne Umwege an einen Entwickler beauftragt werden, um in solchen Fällen keinen unverhältnismäßigen Verwaltungsaufwand zu erzeugen.

Fazit Ein definierter Änderungsmanagement-Prozess leistet einen wesentlichen Beitrag dazu, die Lebensdauer eines Software-Produktes zu verlängern, da das Produkt länger wartbar bleibt. Zudem werden Kosten eingespart, da unnötige Änderungen und die Mehrfachbeauftragung von Duplikaten eingedämmt werden.

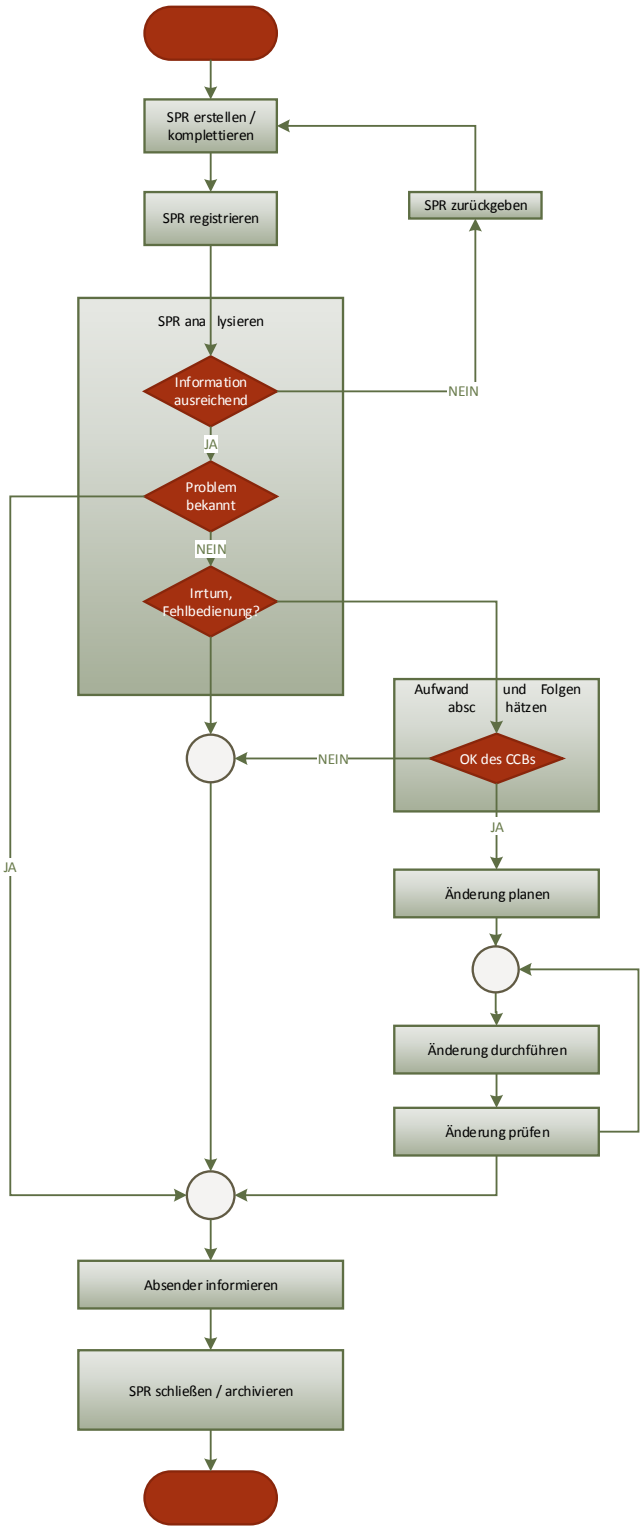

Abbildung 2: Änderungsmanagement-Prozess [3].

3.2 Einsatz eines Issue-Tracking-Systems

Prinzipiell müsste für das Änderungsmanagement kein spezielles Werkzeug (Issue-Tracking-System) eingesetzt werden. Problemmeldungen für ein Software-Proukt könnten per E-Mail oder Telefon übermittelt werden. Ein solches Vorgehen bringt jedoch etliche gravierende Nachteile mit sich, die im Folgenden ausgeführt werden.

Keine zentrale Verwaltung von Problemmeldungen Problemmeldungen werden nicht zentral dokumentiert und Änderungen an Problemmeldungen werden nicht protokolliert. Das kann dazu führen, das Problemmeldungen übersehen werden oder Problemmeldungen unter der Annahme veralteter Informationen bearbeitet werden.

Isolation der Entwickler Es fehlt das gemeinsame Ziel vor Augen. Jeder Entwickler weiß nur, was er selbst zu erledigen hat, er kann nicht auf einen Blick einsehen, worauf das Team momentan hinarbeitet, d.h. wie der Fortschritt für das nächste Release ist

Keine Erkennung von Duplikaten Wenn Problemmeldungen nicht zentral verwaltet werden, ist es fast unmöglich Duplikate gezielt herauszufiltern. Dadurch besteht die Gefahr, dass ein und dasselbe Arbeitspaket mehreren Entwicklern zugewiesen wird.

Durchsetzen eines Workflows ist schwierig Bei der Übermittlung von Problemmeldungen bestehen für den Ersteller zu viele Freiheiten bezüglich der Form der Meldung. Ein definierter Workflow ist viel einfacher zu umgehen, da er sich nicht explizit erzwingen lässt.

Fehlende Integration mit der Konfigurationsverwaltung Die Zuordnung, welche Problemmeldungen zu welchen Änderungen an Software-Artefakten geführt haben, lässt sich praktisch nicht herstellen.

Keine effiziente Releaseplanung Die Zuordnung von Problemmeldungen zu Releases bzw. Versionen eines Softwareproduktes ist ohne ein solches System praktisch nicht sauber herzustellen

Keine Protokollierung von Lösungswegen Der Lösungsweg einer Problembehebung bleibt weitestgehend undokumentiert. Entwickler haben später keine Möglichkeit, nach Lösungswegen von ähnlichen Problemen zu suchen und auf einen Blick zu sehen, welche Änderungen an Software-Artefakten für die Lösung nötig waren.

Diese Nachteile können durch die Einführung eines speziellen Issue-Tracking-Systems vermieden werden. Daher ist für effizientes Issue-Tracking der Einsatz eines solchen Systems unerlässlich.

4 Ist-Analyse

In diesem Kapitel geht es um die Beschaffung von Informationen zur aktuellen Situation des Industriepartners.

Dazu mussten hauptsächlich folgende Fragen geklärt werden:

1. Wie ist der aktuelle Workflow bzgl. der Aufgabenverteilung an die Softwareentwicklung definiert? Ist dies dokumentiert?
2. Welche Issue-Tracking-Systeme werden aktuell eingesetzt?
3. Sind Systeme im Einsatz, die Teilaufgaben eines Issue-Tracking-Systems übernehmen?

Bei einem Kickoff Meeting mit unseren Betreuern des Industriepartners haben wir erste Antworten auf diese Fragen bekommen. Dies beschränkte sich jedoch auf eine grobe Beschreibung der aktuellen Aufgabenverteilung und einem Dokument in dem der Softwareprozess des Industriepartners beschrieben ist. Die Softwareentwicklung bekommt Aufgaben von internen Abteilungen und direkt von Kunden. Der Kontakt findet zum Teil direkt mit den Entwicklern statt. Bei wenigen Produkten werden auch Aufgaben über das bei Synatec eingesetzte Issue-Tracking-System Trac vergeben. In der uns übergebenen Prozessbeschreibung wird kein direkter Aufgabenfluss definiert. Aus dem Projektablauf mit einzeln definierten Phasen und den zugehörigen Rollen ist jedoch eine abstrakte Aufgabenentstehung ersichtlich.

Um Informationen über die tatsächlichen Abläufe und Lebenszeiten von Aufgaben für Softwareentwickler zu erlangen, wurde eine Mitarbeiterbefragung durchgeführt.

Im Anschluss zur Mitarbeiterbefragung wurden vereinzelte Interviews mit leitenden Rollen im Unternehmen durchgeführt. Erst bei diesen Interviews wurde uns mitgeteilt, dass sich das Unternehmen aktuell in einer Umstellungsphase im Auftrags- und Projektmanagement zu SAP Business ByDesign befindet. SAP Business ByDesign ist als direkte Schnittstelle zwischen den Kunden und dem Unternehmen geplant. Dies wird in den nachfolgenden Bewertungskriterien berücksichtigt.

4.1 Mitarbeiterbefragung

Um eine erfolgreiche Befragung durchführen zu können, wurde ein Fragenkatalog erstellt. Zur Durchführung der Befragung wurde das Website-basiertes Umfragewerkzeug Limesurvey[1] für die Fachstudie eingerichtet. Anhand des Fragenkataloges wurden die Fragen im Limesurvey eingepflegt. Limesurvey bietet die Möglichkeit Abhängigkeiten zwischen den Fragen herzustellen. In Limesurvey wurde eine private Umfrage erstellt, die nur für ausgewählte Teilnehmer zugängig ist. Für beendete Umfragen werden zur Auswertung automatisch Statistiken erzeugt. Hierbei ist zu beachten, dass bei den Umfragesessions der Teilnehmer abgespeichert werden, um an einem späteren Zeitpunkt eine Umfrage fortzusetzen. Dies wird in den Auswertungen mit dem Hinweis "Nicht beendet oder nicht gezeigt bzw. Serie6" angegeben.

Ausgewählte Fragen aus dem Fragenkatalog:

1. Haben Ihre Anfragen an SW-Entwickler eine Erweiterung der Funktionalität der Software als Ziel?
2. Haben Ihre Anfragen an SW-Entwickler die Behebung von Fehlern in der Software als Ziel?
3. Auf welchem Wege kommunizieren Sie Ihre Anliegen an Software-Entwickler?
4. Gibt es zusätzliche Anliegen die Sie an Software-Entwickler richten?
5. Gibt es einen definierten Prozess für Wartungsprojekte und an welcher Stelle lässt sich die Definition nachschauen?
6. Welche Tools werden zum Erfassen von Anforderungen verwendet?

Diese Fragen wurden als Vorlage für die Umfrage ausgearbeitet.

[1] https://www.limesurvey.org/de

An der Umfrage haben 20 von 49 eingeladenen Teilnehmern aktiv teilgenommen. 11 der aktiven Teilnehmer haben die Umfrage vollständig ausgefüllt. Folgend werden ausgewählte Fragen der Umfrage angegeben und deren Ziele für die Fachstudie erläutert. Ausgewählte Fragen aus der Umfrage:

1. Frage: Gibt es Erweiterungsverträge?
   Ziel: Ist es möglich, dass Änderungsanfragen vom Kunden kommen

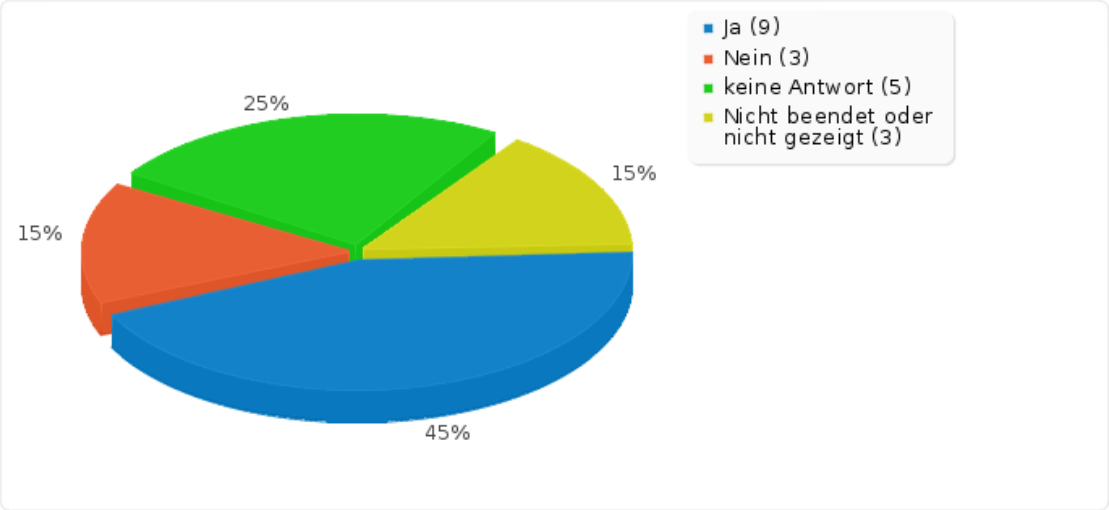

Abbildung 3: Auswertungsstatistik Frage 1

2. Frage: Gibt es vorgegebene Arten/Stufen von Erweiterungsverträgen?
   Ziel: Differenzierte Workflows bezüglich Änderungsanfragen zu identifizieren

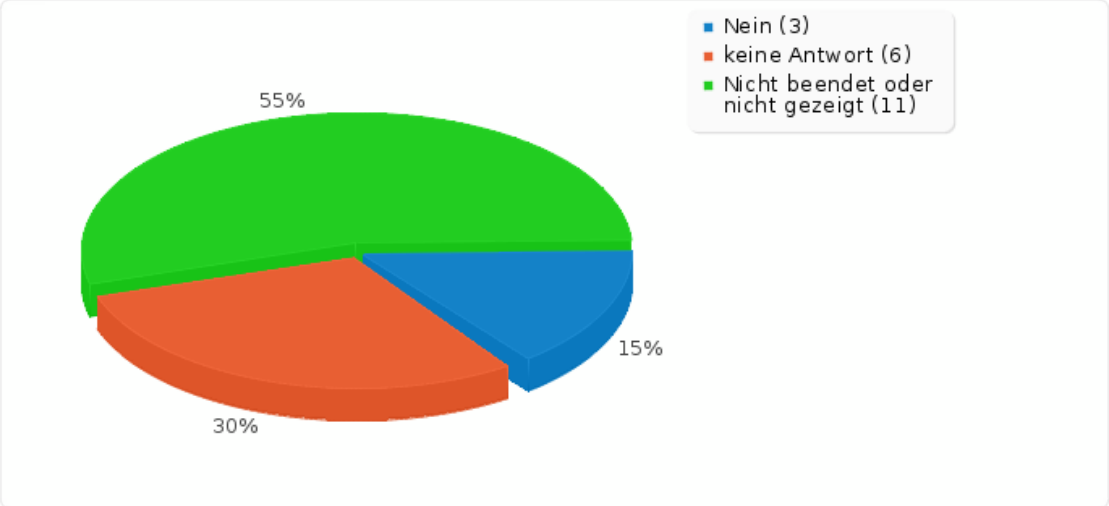

Abbildung 4: Auswertungsstatistik Frage 2

3. Frage: Mithilfe welcher Werkzeuge werden Projekte gesteuert?
   Ziel: Eingesetzte Werkzeuge aus denen Tickets entstehen könnten zu identifizieren

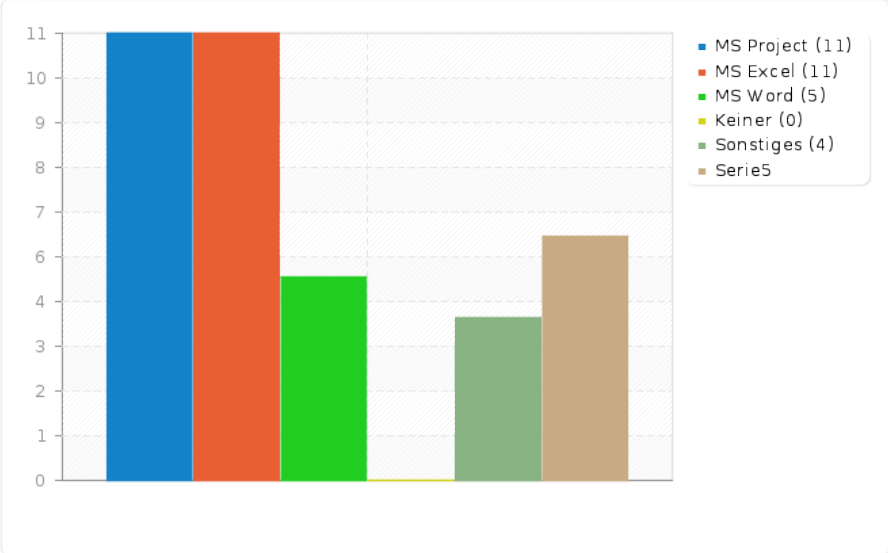

Abbildung 5: Auswertungsstatistik Frage 3

4. Frage: Werden alle Projekte der Firma Synatec zentral in einem System verwaltet?
   Ziel: Identifizierung der möglichen Anbindungskomplexität

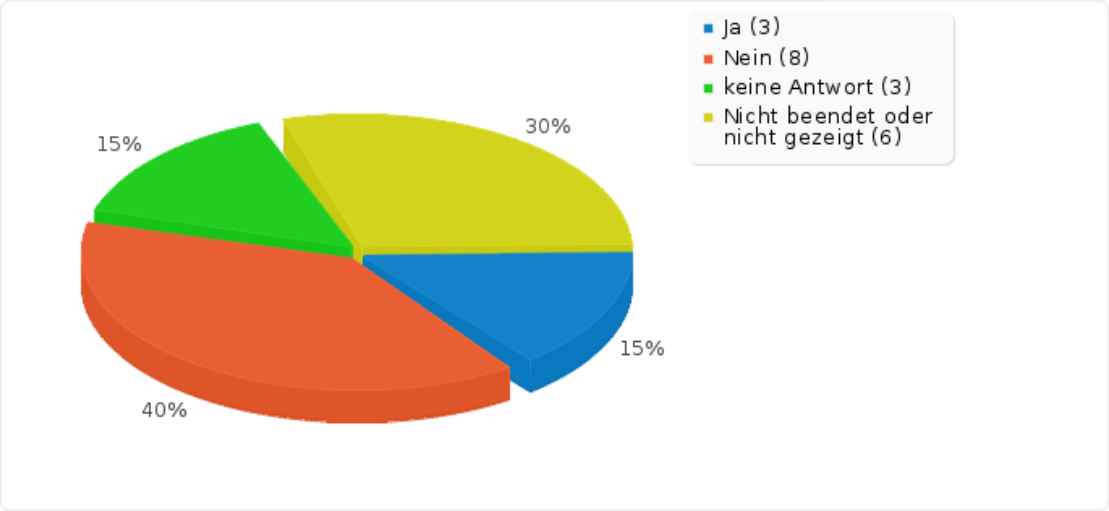

Abbildung 6: Auswertungsstatistik Frage 4

5. Frage: Wird die Zuteilung von Software-Entwicklern zu Projekten an einer zentralen Stelle verwaltet?
   Ziel: Zuständige Rollen der Aufgabenverteilung identifizieren

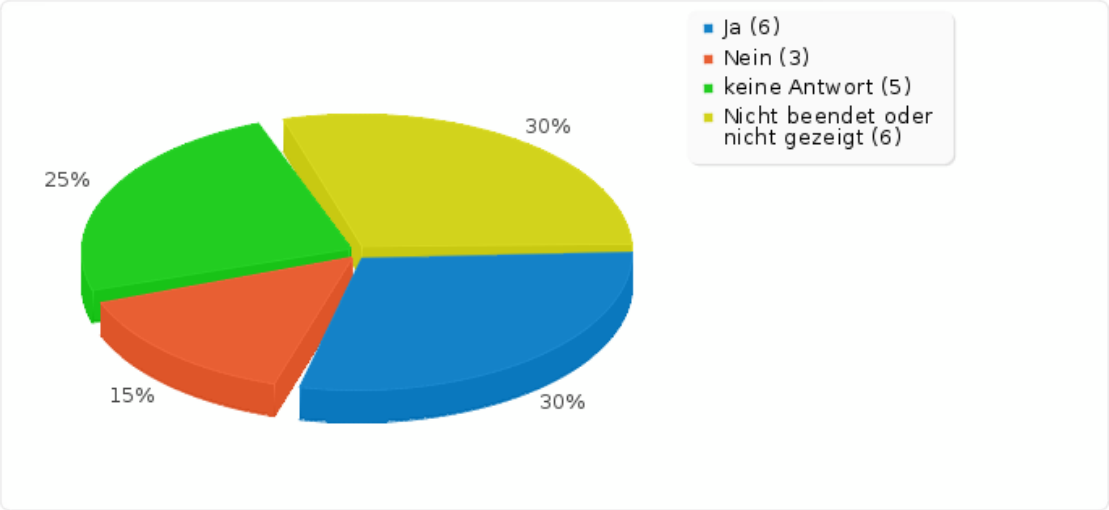

Abbildung 7: Auswertungsstatistik Frage 5

6. Frage: Wer darf der Qualitätssicherungsabteilung Aufgaben erteilen?
   Ziel: Identifizierung des Standpunktes der Qualitätssicherung im aktuellen Workflow

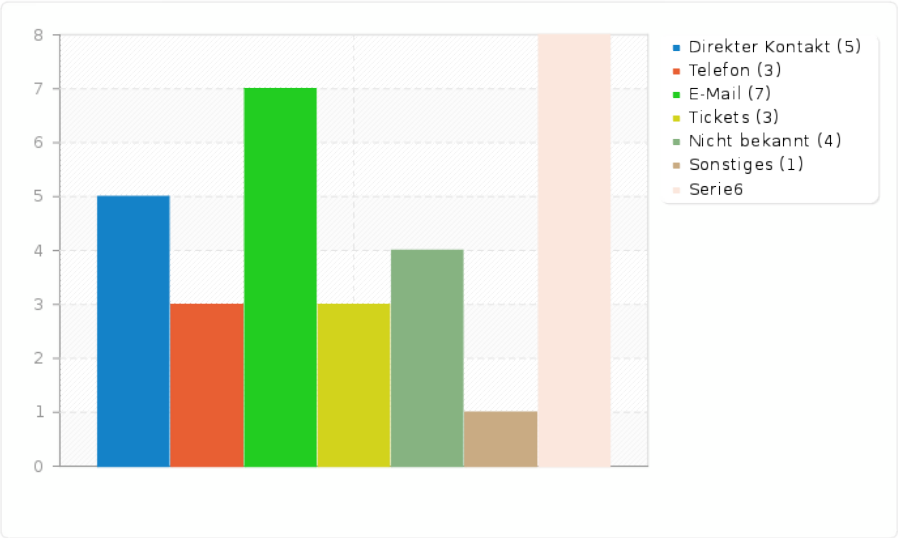

Abbildung 8: Auswertungsstatistik Frage 6

7. Frage: Wie bekommt die Qualitätssicherungsabteilung Ihre Aufgaben?
   Ziel: Identifizierung der möglichen Anbindung der Qualitätssicherung an die Ticketverwaltung.

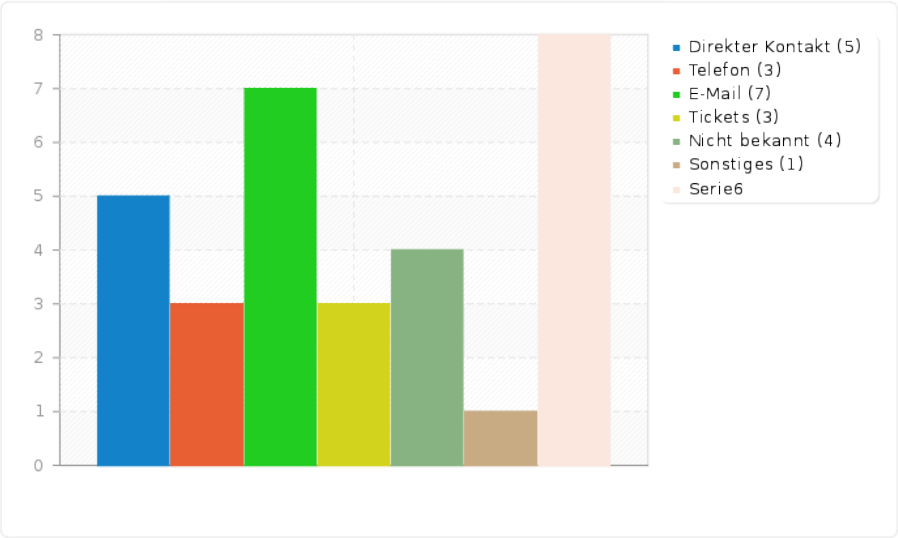

Abbildung 9: Auswertungsstatistik Frage 7

8. Frage: Auf welchem Wege kommunizieren Sie Ihre Anliegen an Software-Entwickler?
   Ziel: Welche Tools aktuell zur Ticketerstellung oder ähnlichem exisiteren

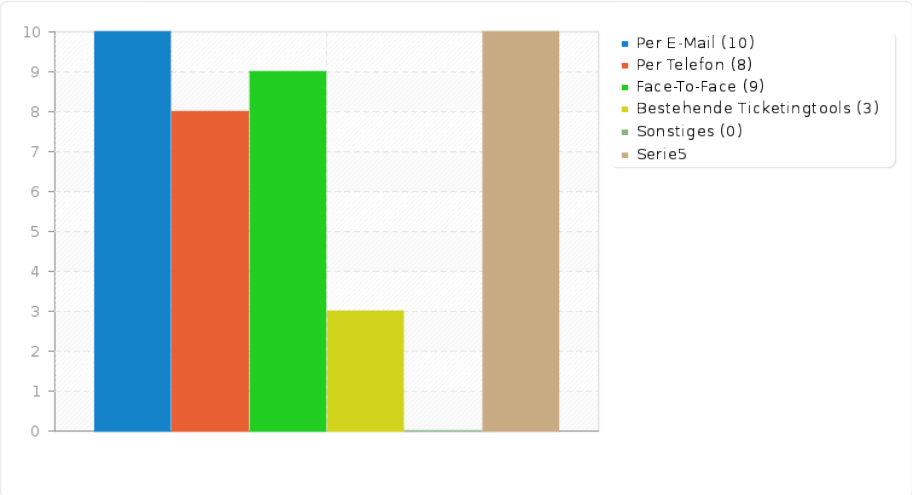

Abbildung 10: Auswertungsstatistik Frage 8

9. Frage: Welche Rollen sind befugt Software-Entwickler Projekten zuzuteilen?
   Ziel: Identifizierung der Rollen, die Aufgaben an Softwareentwickler vergeben dürfen

Abbildung 11: Auswertungsstatistik Frage 9

Aufgrund der gezeigten und weiteren Ergebnissen kamen wir zu dem Entschluss, dass diese Ergebnisse von uns nicht sinnvoll auswertbar sind. Die Antworten zielten meist nicht auf einen gemeinsamen Konsens aus. Daraufhin wurden von uns einzelne Interviews, überwiegend mit Schlüsselpersonen des Industriepartners, durchgeführt, wodurch wir die Durchführung von Softwareprojekten und die Aufgabenverteilung analysieren konnten.

4.2 Durchführung von Softwareprojekten

Der Industriepartner hat mehrere Softwareprodukte, die Prozessdaten verarbeiten. Diese Software ist im ständigen Kontakt mit den Kunden und der Serviceabteilung. Deshalb sind die Softwareprojekte des Industriepartners überwiegend kundengetrieben. Softwareprojekte werden nach einer Prozessbeschreibung mit sieben Phasen und sieben Gateways durchgeführt.

1. Projektdefinition
2. Analyse
3. Design
4. Implementation
5. Integration / Test
6. Systemtest
7. Inbetriebnahme

Das Gateway nach einer jeden Phase repräsentiert ein Review in dem die erhobenen Anforderungen an die Phase verifiziert werden. Die Phasen Design, Implementation und Integration/Test können iteriert werden. Die anderen Phasen sind abschließend. Für das Projektmanagement werden MS Project und MS Excel eingesetzt.

4.3 Aufgabenverteilung

In diesem Abschnitt soll geklärt werden, welche Instanzen befugt sind an die Softwareentwickler Aufgaben zu vergeben. Bei dem Industriepartner wird eine Linienorganisation als Organisationsstruktur angewendet. Die Softwareentwickler sind in dieser Linienorganisation auf drei Abteilungen aufgeteilt: Server Systems, Line Browser und Software Quality Assurance. Die Softwareentwickler unterliegen Ihren jeweiligen Abteilungsleitern. Die Softwareentwickler und die Abteilungsleiter unterliegen dem Leiter des Software Departments. Das Software Department und dessen untergeordneten Instanzen unterliegt dem Technologie Leiter. Damit keine Informationen vorenthalten werden, ist eine direkte Kommunikation und direkte Aufgabenverteilung erwünscht. Das bedeutet, dass möglichst keine Instanzen übersprungen werden sollen. Dies ermöglicht Informationsverluste zu vermeiden und eine gleichmäßige Auslastung bzw. eine optimale Verteilung der Mitarbeiter zu garantieren.

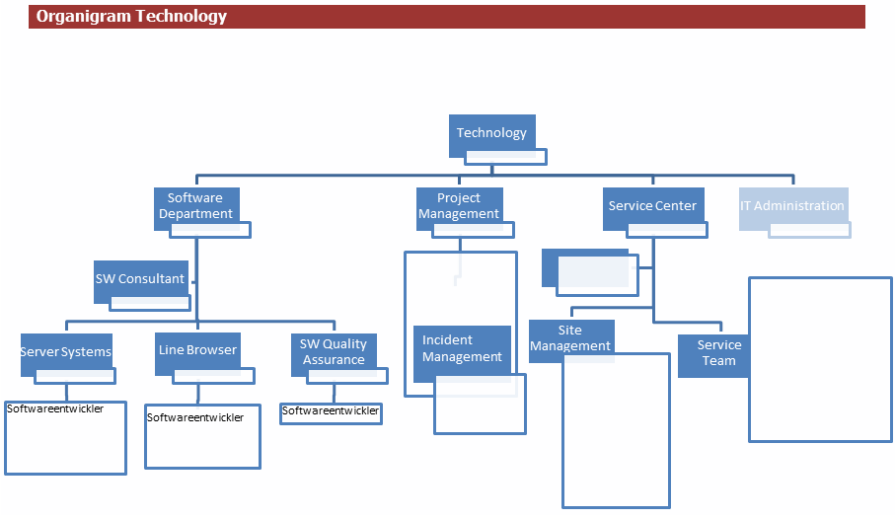

Abbildung 12: Organigram der technischen Abteilung

In der Aufgabenverteilung ist vereinzelt das Issue-Tracking-System Trac im Einsatz. Aufgaben werden zusätzlich von den zuständigen Abteilungsleitern per Email (MS Word), Telefon oder ein persönlichem Treffen kommuniziert. Im Form einer Email besteht die Möglichkeit ein Formular zu verwenden. Die Softwareentwickler werden zusätzlich von Servicemitarbeitern direkt kontaktiert und beauftragt, was eine nötige Resourcenübersicht der Abteilungsleiter erschwert. Telefonkontakte bestehen außerdem zwischen Softwareentwicklern und Servicetechnikern ebenso wie mit Kundenrepräsentanten.

5 Marktübersicht

Dieses Kapitel stellt einige auf dem Markt verfügbare Issue-Tracking-Systeme stichpunktartig vor. Die Angaben, sowie das Bildmaterial stammen von den jeweiligen Webseiten der Anbieter, weswegen manche erwähnten Punkte nicht zwangsweise auch erfüllt sein können.

Bugzilla

Hersteller:	bugzilla.org
Version:	4.5.1
Lizenz:	Creative Commons License
Preis:	Kostenfrei
Link:	`http://www.bugzilla.org/`

Bugzilla ist ein Fehler-Verfolgungs-System oder Bug-Tracking System. Bug-Tracking Systeme erlauben es einzelnen Entwicklern oder auch Gruppen, noch zu behebendes Fehlverhalten in ihrem Produkt effektiv zu verfolgen. Obwohl Bugzilla "umsonst" ist, hat es dennoch viele Funktionen, die der kommerziellen Konkurrenz fehlen. Aus diesen Gründen wurde Bugzialla der Favorit von tausenden Organisationen weltweit.

Trac

Hersteller:	Edwall Software
Version:	1.0
Lizenz:	BSD-Lizenz
Preis:	Kostenfrei
Link:	`http://www.edgewall.org/`

Trac ist ein minimalistischer Ansatz einer web-basierenden Verwaltung von Softwareprojekten. Das Ziel von Trac ist es eine effektive Verfolgung und Handhabung von Anliegen, die die Software betreffen zu vereinfachen. Alle Aspekte von Trac wurden mit der einzigen Zielsetzung entworfen Entwicklern zu helfen tolle Software zu schreiben, ohne den im Team etablierten Prozessen im Weg zu stehen oder diese zu beeinflussen.

Redmine

Hersteller:	Redmine
Version:	2.2.0
Lizenz:	GNU General Public License v2
Preis:	Kostenfrei
Link:	`http://www.redmine.org/`

Redmine ist eine flexible Web-Anwendung zum Verwalten von Projekten. Es verfügt über eine umfangreiche Grundausstattung und erlaubt es die Funktionalität durch das Einbinden von zusätzlichen Plug-Ins zu erweitern.
So kann zum Beispiel schon in der Grundausstattung ein Gantt-Diagramm anhand der Projektplanung angefertigt werden.
Weitere Stärken sind das Verwalten von mehreren parallelen Projekten und die Möglichkeit untergeordnete Projekte anzulegen.

Apache Bloodhound

Hersteller:	Apache Bloodhound
Version:	0.7
Lizenz:	Apache License
Preis:	Kostenfrei
Link:	`http://bloodhound.apache.org/`

Apache Bloodhound orientiert sich stark an der Umsetzung von Trac, jedoch bietet es auch die Möglichkeit mehrere Produkte gleichzeitig zu verwalten. Funktionen wie ein Wiki oder ein Source-Code-Browser sind bereits integriert.
Es hat einen einfachen Installationsvorgang, der das Verwenden des gewünschten Webservers oder Datenbank anbietet.
Für die komfortable Benutzung wird ein Dashboard angeboten, welches es den Benutzern ermöglicht die Repräsentation selbst anzupassen, um die wichtigsten Information gleich auf einen Blick zu erfassen.

Clarizen

Hersteller:	Clarizen
Version:	V6
Lizenz:	SaaS
Preis:	$29,95 - $54,95 pro Monat
Link:	`http://www.clarizen.com/`

Clearizen's IT Project Management Software ist entworfen, um den Entwickler in seiner Arbeitsweise zu unterstützen und nicht einzuschränken. Es bietet ein frei konfigurierbare Dashboards, Workflows, Geschäftsregeln und bewährte Prozesse, genauso wie angepasste Aktionen und Felder. Es wird den Entwicklern nicht aufgezwungen ihre Arbeitsweise neu zu erfinden, sodass die Adaptionsrate hoch gehalten ist.
Es ermöglicht eine zentrale Verwaltung aller Dokumente geordnet nach Projekten oder Arbeitspaketen. Clarizen integriert auch Werkzeuge wie Box und Googl Drive, um die Strategie der Dokumentenverwaltung eines Unternehmens zu Unterstützen.

Gemini

Hersteller:	Countersoft
Version:	5.4.2
Lizenz:	Proprietär, SaaS
Preis:	kostenfrei[1] oder ab $120,00 pro Monat
Link:	`http://www.countersoft.com/`

Durch das Einsetzen von Projektvorlagen gibt Gemini jedem Team und Projekt eine Projektverfolgung in ihrer eigenen Sprache und ihren eigenen Prozessen. Der kompromisslose Ansatz im Bezug auf Projekte, der von Gemini verfolgt wird, hat einen großen Vorrang.
Diese Umsetzung befolgt die Richtlinie, dass alles entwickelt wurde um die Sachen, die am wichtigsten sind zu respektieren - Menschen und Prozesse.
Es erlaubt Querbeziehungen zwischen Projekten, die praktisch keine Einschränkungen haben, was das Schachteln dieser anbelangt.

[1] Bei bis zu zehn Benutzern, aber ohne Hosting-Service

OnTime

Hersteller:	Axosoft
Version:	13,3
Lizenz:	Proprietär, SaaS
Preis:	$3995,00, oder ab $35,00 im Monat
Link:	`http://www.axosoft.com/`

Mit vollständig anpassbaren Workflows und Vorlagen für die einzelnen Felder lässt sich OnTime Scrum den Prozessen, die das Entwicklungsteam braucht, anpassen. Von da an kann man den Fortschritt mit dynamischen und visuellen Tools, wie zum Beispiel OnTime Dashboard, mitverfolgen. Das Hinzufügen von User Stories wird zu den Benutzern durch die Drag and Drop Schnittstelle vereinfacht.
Filter, Tastatur-Shortcuts und Gruppen ermöglichen es den Produkt-Backlog mühelos zu organisieren und zu verwalten. OnTime Scrum ist erstellt um die Komponenten für Bugtracking, Helpdesk und Wiki von OnTime zu unterstützen.

DevSuite

Hersteller:	TechExcel
Version:	9.2.1
Lizenz:	Proprietär, SaaS
Preis:	$0,00, oder ab $180,00 im Monat
Link:	`http://techexcel.com/`

Sowohl agile als auch traditionelle Teams sollen gleichermaßen wirksam bei der Benutzung von DevSuite unterstützt und verwaltet werden. Agile Entwicklungsmerkmale, wie vorgegebene Iterationsdauer und Burndown-Reports sind nahtlos in den QS Testzyklusverwalltung integriert. Das vollständig unterstützte Produkt-Backlog ersetzt die konventionellen Anforderungserhebung nicht, vielmehr integriert es mit dem einzigen Anforderungsverwaltungswerkzeug, dass eine wirkliche zweiwege MS-Word-Integration unterstützt um ergänzende Funktionalität hinzuzufügen, falls es gewollt ist. DevSuite erlaubt es dem Benutzer Entwicklung auf seine Art zu verwalten, man schafft sich somit Lösungen, die die eigenen Bedürfnisse erfüllen.

JIRA

Hersteller:	Attlassian
Version:	6.0.4
Lizenz:	Proprietär, SaaS
Preis:	ab $10,00, oder ab $10,00 im Monat
Link:	`http://www.axosoft.com/`

JIRA ist ein Projektverfolgungstool für Teams, die großartige Software erstellen wollen. Tausende Teams haben sich für JIRA entschieden, um Ihre Aufgaben besser zu koordinieren. Mit JIRA ist man immer informiert, woran das Team gerade arbeitet.
Der Prozess eines Entwicklungsteams ist der zentrale Punkt, um den sich alles dreht, deswegen bietet JIRA Workflows die vielen ähnlich sind und darüber hinaus die Möglichkeit bestehende Prozesse vollständig an die eigenen Bedürfnisse anzupassen und nicht umgekehrt. Der gut ausgebaute Marketplace von Atlassian bietet sehr viele Plug-Ins um JIRA und andere Produkte von Atlassian zu erweitern und anzupassen. Der Marketplace besitzt alle Informationen, die man für die Entscheidung zum Test einer Erweiterung benötigen, wie Kompatible Version, Preise, Rezensionen.

Planbox

Hersteller:	Planbox
Version:	–
Lizenz:	SaaS
Preis:	ab $10,00, oder ab $10,00 im Monat
Link:	`https://www.planbox.com/`

Planbox bietet eine einzigartige vier Ebenen Struktur, mit deren Hilfe sich Planbox an beliebigen Geschäftsfeldern oder Umgebungen anpassen lässt. Als solches ist es ein gutes Werkzeug für alle Unternehmen und Organisationen, die nach einem einfachen und mächtigen agilen Projektverwaltungswerkzeug suchen. Es bietet Echtzeitbenachrichtigungen über den Stand einzelner Aufgaben, die von Teammitgliedern bearbeitet werden. Durch die Möglichkeit Dateien direkt an die einzelnen Task anzuhängen wird das Versenden von arbeitsrelevanten Dokumenten überflüssig. Durch Abschätzen der erwarteten Zeitaufwandes und dem Vergleich mit der für eine Aufgabe tatsächlich benötigten Zeit wird schnell sichtbar, welche Teammitglieder mit Arbeit überladen sind.

MantisBT

Hersteller:	MantisBT
Version:	–
Lizenz:	GPL
Preis:	kostenfrei
Link:	`http://www.mantisbt.org/`

MantisBT ist ein kostenfreies web-basierendes Bugtracking-System. Es ist in der PHP Skriptsprache geschrieben und unterstützt die Datenbanken MySQL, MS SQL und PostgreSQL und zusätzlich einen Webserver. Es wurde mittlerweile unter den Betriebssystemen wie Windows, Mac OS, OS/2 und auch weiteren installiert.
Nahezu jeder Web-Browser sollte in der Lage sein als Client zu dienen. Zusätzlich steht mit MantisTouch eine App für iPhone, Android und Windows Phone zur Verfügung.

Pivotal Tracker

PIVOTAL TRACKER

Hersteller:	Pivotal Labs
Version:	–
Lizenz:	SaaS
Preis:	ab $7,00 im Monat
Link:	`http://www.pivotaltracker.com/`

Pivotal Tracker ist ein einfach zu bedienendes Werkzeug zum Verwalten von agilen Projekten mit dem Fokus auf der Zusammenarbeit von Software-Entwicklungsteams. Von Pivotal Labs entwickelt verkörpert es bewährte agile Methoden, die auf der Erfahrung hunderter erfolgreicher großer Projekte beruhen.
Es ist einfach zu erweitern und lässt sich mit bereits verwendeten Werkzeugen, wie unter Anderem Bugzilla und Zendesk integrieren lassen und verfügt über eine flexible Schnittstelle zu anderen Werkzeugen. Pivotal Tracker verfügt über die selbe Darstellung für alle, vom CEO bis zum Entwickler. Zusammenarbeit auf einer feingranularen Ebene, ohne dass man den Gesamtüberblick verliert.

6 Anwendungsbereich und Nutzungsszenarien

Die Fachstudie wurde in Kooperation mit einem Industriepartner durchgeführt. Dieser Industriepartner stellt Software im Bereich der Prozessdaten her. Der Kundenkreis beläuft sich aktuell überwiegend auf die Automobilindustrie, jedoch wächst die Nachfrage und der Absatz in der allgemeinen Industrie stetig. Die daraus resultierenden Anforderungen an die Software wachsen verhältnismäßig schneller als der Kundenzuwachs. Diesem Wachstum steht der Personalaufwand gegenüber, der die Aufgabenverteilung und den Überblick über die Software zunehmend erschwert. Dieser Problematik soll unter anderem durch das auszuwählende Issue-Tracking-System gegengewirkt werden.

6.1 Mengengerüst

Es soll ein Issue-Tracking-System für 200 Nutzer ausgewählt werden. Die jährlichen Kosten dürfen sich auf bis 12000 EUR belaufen. Die Projektanzahl als auch die Ticketanzahl darf nicht beschränkt sein.

6.2 Mögliche Issue-Tracking-Workflows

Im Verlauf unserer Fachstudie wurde beim Industriepartner eine "Work Guideline" verfasst, die unter anderem Richtlinien und einen "Ticket-Lifecycle" für die Entwicklung beinhaltet. Zudem liegt eine Vorlage für einen "Task Workflow" vor, die das Durchlaufen der einzelnen Phasen und den Umfang eines Tasks simulieren soll. Der "Task Workflow" tritt in Kraft, sobald die Zuständigkeit für einen Task an die Softwareabteilung übergeben wurde.

Ursprünglicher Task-Ablauf

Dieser ursprüngliche Entwurf beschreibt die Status und deren Abhängigkeiten für einen Task. Nach der "Task Description" kann derjenige, der die Aufgabe bearbeiten soll entscheiden, ob ihm die Beschreibung genügt, oder ob diese gegebenenfalls ergänzt werden soll. Wenn die Beschreibung genügt dann erstellt der "Task Ownerïn dem Schritt *Detailed Design* einen Entwurf, der auf seine Plausibilität vom *Task Author* entweder akzeptiert oder mit Feedback an den *Task Owner* zurück gegeben wird. Sobald das Design akzeptiert wurde, wird die *Qualitätssicherung* informiert und aktualisiert gegebenenfalls die Testfälle und den Integrationsplan. Parallel und unabhängig dazu wird mit der Implementierung begonnen. Der *Task Owner* prüft, ob er den Task vollständig abgeschlossen hat. Sobald dies der Fall ist wird das Ergebnis auf dem lokalen Entwicklungsbranch integriert, die Dokumentation wird abgeschlossen und die Ergebnisse vom *Task Author* geprüft. Wenn diese Ergebnisse seinen Anforderungen genügen, dann ist der Task abgeschlossen und die *Qualitätssicherung* wird über die Fertigstellung informiert, sodass Integrations- und Testpläne eventuell angepasst werden können. Sollte das Resultat nicht

zufriedenstellen sein, dann geht es an den *Task Owner* zurück, damit er die Implementierung überarbeiten kann.

Angepasster Task-Ablauf

Die nachfolgende Grafik bildet den allgemeinen Task-Ablauf, der von uns vorgeschlagen wird, mit allen darin beinhalteten Sub-Tasks ab.

Im wesentlichen unterscheidet dieser sich von dem oben beschriebenen Ablauf darin, dass die Qualitätssicherung einen stärkeren Stellenwert bekommt.
So wird zum Beispiel die Implementierung und die Unit-Tests, Prüfling genannt, nach ihrem Abschluss und bevor sie integriert werden von *Qualitätssicherung* begutachtet. Bei Inkonsistenz und/oder Fehlern muss der Prüfling mit einer Fehlerbeschreibung an den *Task-Owner* zurückgegeben werden. Im Anschluss prüft der *Task-Owner*, ob der Fehler in der Implementierung oder bereits im Design vorhanden ist. Sollte dieser schon im Design liegen, so wird das Design gegen die Aufgabenbeschreibung geprüft.

Liegt der Fehler am Design, so wird dieser direkt vom *Task-Owner* behoben und der *Task-Author*, wird über die Änderung in Kenntnis gesetzt. Ist der Fehler jedoch auf eine fehlerbehaftete Beschreibung der Aufgabe zurückzuführen, so muss der *Task-Author* in Kooperation des *Task-Owners* die Beschreibung ergänzen oder ggf. diese überarbeiten.

Zudem ist das Vorbereiten des Deployments, das auch von der *Qualitätssicherung* durchgeführt wird, ein Bestandteil eines Tasks.

Dieser Workflow sollte jedoch gerade auf die *Task-Typen* wie z.B *Servicerequest* oder *Emergency-Fix* angepasst werden. In manchen Fällen kann das Durchlaufen des gesamten Ablaufs nicht zwangsweise erforderlich sein, gerade wenn es sich um unerhebliche und kleine Aufgaben handelt. Für diese Aufgaben soll während des Einsatzes eine gekürzte Version auf der Basis von Erfahrungen der Entwickler und Team-Leiter des Industriepartners erstellt werden.

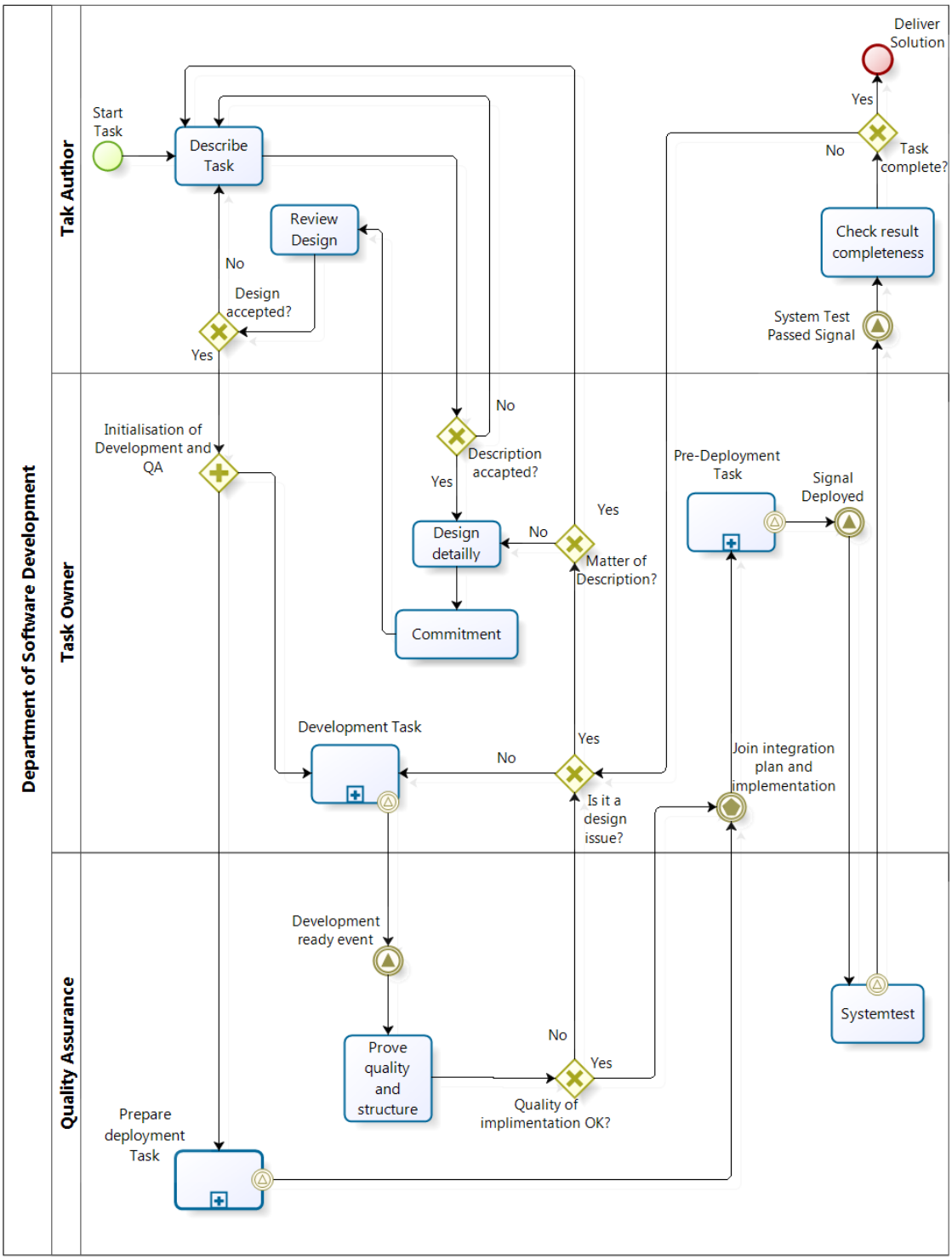

Abbildung 13: Angepasster Task-Ablauf

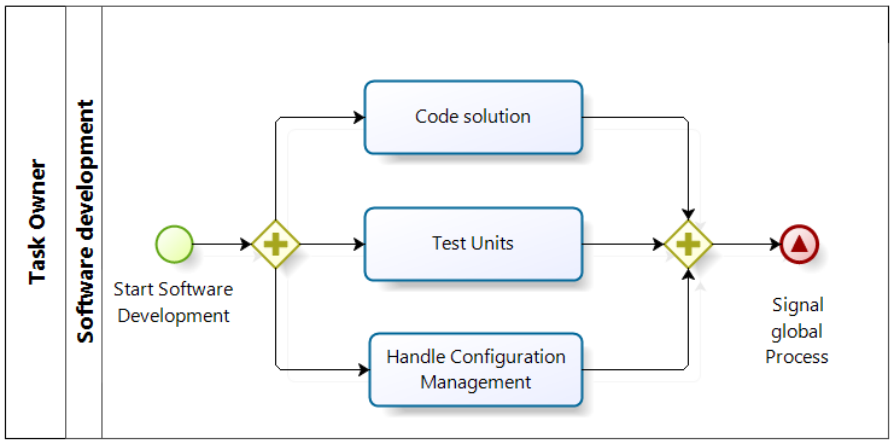

Abbildung 14: Development Task

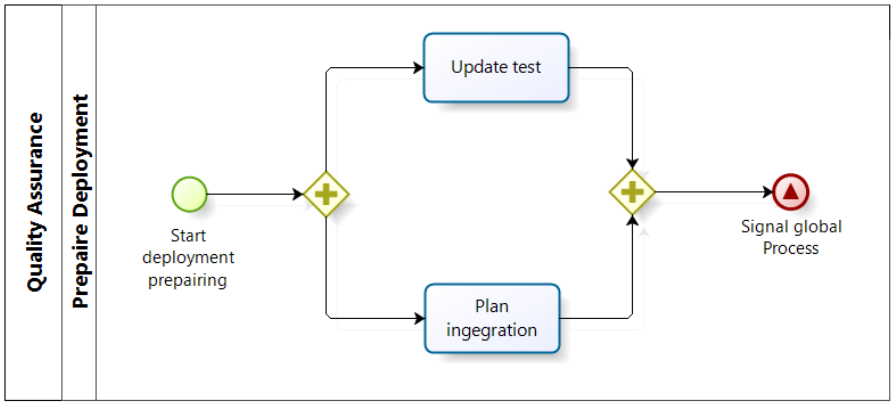

Abbildung 15: Prepair deployment Task

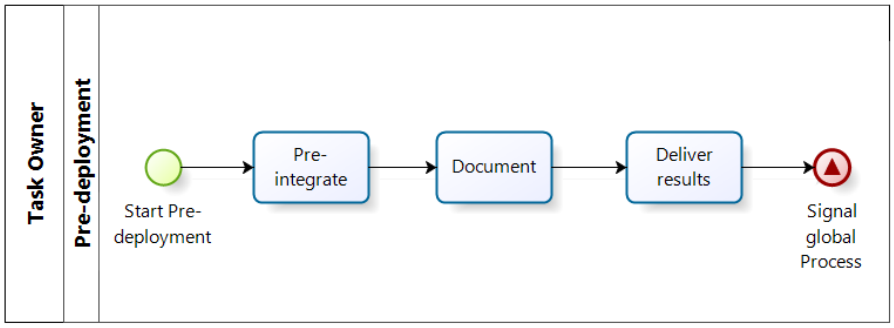

Abbildung 16: Pre-deployment Task

6.3 Mögliche Ticketstruktur

In diesem Kapitel wird beschrieben welche Angaben für ein allgemeines Ticket zur Beschreibung eines Tasks notwendig ist. Außerdem werden zulässige Statusänderungen eines allgemeinen Tickets aufgezeigt mit den notwendigen Voraussetzungen um einen Status anzunehmen.

Für einzelne Tickettypen muss der verlangte Umfang an Informationen gegebenenfalls angepasst bzw. gekürzt werden. Die Anpassung erfolgt jedoch erst im Einsatz der Ticketstruktur, da diese anhand von Erfahrungswerten durchgeführt wird. Außerdem wird erst im Einsatz ermittelt welche *Task-Spezifikationen* für die Analyse von durchgeführten Aufgaben notwendig sind, da noch keine nähere Einschränkungen an Statistiken, die ermittelt werden können, feststehen.

Ergänzend müssen auch die zulässigen Transitionen der Ticketstatus angepasst werden, da gerade in dringenden Fällen häufig nicht der gesamte Ablauf befolgt werden kann, diese eine äußerst kurze Reaktionszeit verlangen und auch außerhalb der regulären Geschäftszeiten auftreten können, wenn nicht alle Rollen vertreten sind.

Parameter eines Tickets

Die folgende Tabelle beinhaltet empfohlene Parameter mit möglichen Werten und einer kurzen Beschreibung, die ein Ticket kennzeichnen.

Parameter	Mögliche Einträge	Kurzbeschreibung
Task-Author	Name eines Mitarbeiters	Mitarbeiter, der den Task erstellt.
Task-Owner	Name eines Mitarbeiters	Mitarbeiter, der den Task bearbeitet.
Task-Type	• Bug • Emergency-Fix • Servicerequest • Enhancement • Extention • Changerequest • Re-Implementation • New Implementation	In diesem Feld wird beschrieben um welche art von Task es sich handelt. Dies hilft einerseits bei der Integrationsplanung, als auch bei der arbeitsverteilung.
Complexity	• Simple • Normal • Difficult • Intense	Geschätzter Schwierigkeitsgrad der Aufgabe, aus einer der vier Stufen ausgewählt wird.
Priority	• Minor • Normal • Major • Critical	Priorität, die der rechzeitzeitige Abschluss des Tasks hat, diese kann zwischen konkurierenden Tasks abgewogen werden.
Risk-Level	• Low • Increase • High	Die Wichtigkeit, die ein Task im globalen Kontext, zB wichtigkeit für den vollschtändigen Umfang einer Auslieferung hat, und das damit verbundene Risiko. Dieses kann anhand der entstehenden Kosten bei nicht rechtzeitiger Fertigstellung ermittelt werden.
Estimated time	Stundenzahl	Geschätzter Aufwand um die im Task beschriebene Aufgabe fertig zu stellen in Stunden

Links	Hyperlink	Verlinkung auf Artikel, deren Inhalt relevant für die Task-Bearbeitung. Außerdem werden hier Links zu Tickets hinterlassen, die Quer- oder Längstbeziehungen haben. ist, und/oder die Taskbschreibung beinhaltet
Deadline	Datum	Das Datum, bis zu welchem die Aufgabe fertig bearbeitet sein soll.
State	• Assigned • Designed • Design Accepted • Design Rejected • Implementation • Quality Assurance • Implementation Accepted • Implementation Rejected • Integrated • Systemtest Passed • Systemtest Failed • Closed	Der Eintrag beschreibt in welchem der prädefinierten Status ein Task sich befindet.
Feedback note	Fließtext	Anmerkung des Task-Owners nach dem Abschluss des Tasks.
Delivery Content	Liste mit wählbaren Elementen	Liste mit Artefakten, von denen beliebig viele, auch keine, als obligatorisch im laufe eines Tasks anzufertigen sind. Der Status in dem das jeweilige Artefakt vorliegen muss, wird beim Industriepartner intern beschlossen.
Task-Description	Fließtext	Kurze beschreibung des zubearbeiteden Tasks, Hinweise und sonstige Anmerkungen des Task-Authors.

Tabelle 1: Parameter für die Spezifikation von Tasks

Ticketstatus
Die unten abgebildete Tabelle enthält alle Status, in denen sich ein allgemeines Ticket befinden kann, in der Reihenfolge, wie sie in einem optimalem Ablauf durchschritten werden. Der Verantwortliche ist die Person, welche im Ticket unter der angegebenen Rolle eingetragen ist, und ist für dem zugeordneten Status für den Task verantwortlich. Der Umfang beschreibt Artefakte, die innerhalb des zugeordnetem und vor dem Wechsel in den nächsten Status verlangt sind. Die Kriterien, deren Erfüllung vor einem Statuswechsel verlangt werden, müssen auch in allen folgenden Status erfüllt sein.

Status	**Verantwortlicher**	**Umfang**
Released	Task Author	• Des Task-Owner muss über den Inhalt des Task informiert sein • Versionierung und Integrationsplanung • Beschreibung des Tasks • Priorität des Tasks • Release-Plan • Task-Type
Assigned	Task Owner	• Eine begründete Aufwandsschätzung • Risiko gewichtung • Komplexitätsgewichtung • Detaillierter Entwurf der Aufgabe
Designed	Task Author	• Prüfung der eingereichten Vorschläge durch den Task-Author
Design Accepted	Task Owner	• Das Einverständniss mit den eingereichten Rahmenbedingungen und des Designs. • Der Task Owner kann mit der Bearbeitung fortfahren.
Design Rejected	Task Owner	• Begründung warum das Design und die eingereichten Rahmenbedingungen Abgelehnt wurden. • Der Taskautor muss Korekturen an seinem Vorschlag vornehmen
Implementation	Task Owner	• Quellcode • Spezifizierte Unittests • Dokumentation ausgeführter Unitttests • Aktueller Stand des Taskfortschritts im Repository

Quality Assurance	Qualitätssicherung	• Prüfen der Einhaltung firmeninterner Entwicklungsrichtlinien • Prüfen der Testabdeckung • Prüfen ob der Entwurf eingehalten und umgesetzt wurde • Integrationsplanung
Implementation Accepted	Task Owner	• Alle bisherigen Kriterien, die für den Task verlangt sind, wurden erfüllt • Der Task-Owner kann mit der Integration in den Development-Branch fortfahren
Implementation Rejected	Task Owner	• Liste mit Mängeln, die von der Qualitätssicherung erstellt wird • Der Task-Owner muss den Status, in dem Fehler hätten entstehen können, einstellen • Der Task-Owner behebt die Mängel
Integrated	Qualitätssicherung	• Testfälle für die im Task entstandenen Softwarekomponenten • Ausführung des Systemtest • Systemtest Protokoll
Systemtest Passed	Task Author	• Vollständige dokumentation des Tasks • Einpflegen der Entstandenen Artefakte ins Repository
Systemtest Failed	Task Owner	• Systemtest mit Fehlgeschlagenen Systemtestfällen • Liste mit sonstigen Mängel
Closed	Task Author	• Der Task ist abgeschlossen und die enstandenen Softwarekomponenten können im spezifiertem Umfang ausgeliefert und eingesetzt werden

Table 2: Beschreibung der Ticketstatus

Ticketstatusüberführungen

Die folgende Abbildung zeigt die validen Zustände in denen sich ein Ticket befinden kann. Zudem sind auch die zulässigen Übergänge zwischen den jeweiligen Zuständen. Die Vorgaben für einen Statuswechsel sind der vorangegangenen Tabelle zu entnehmen.

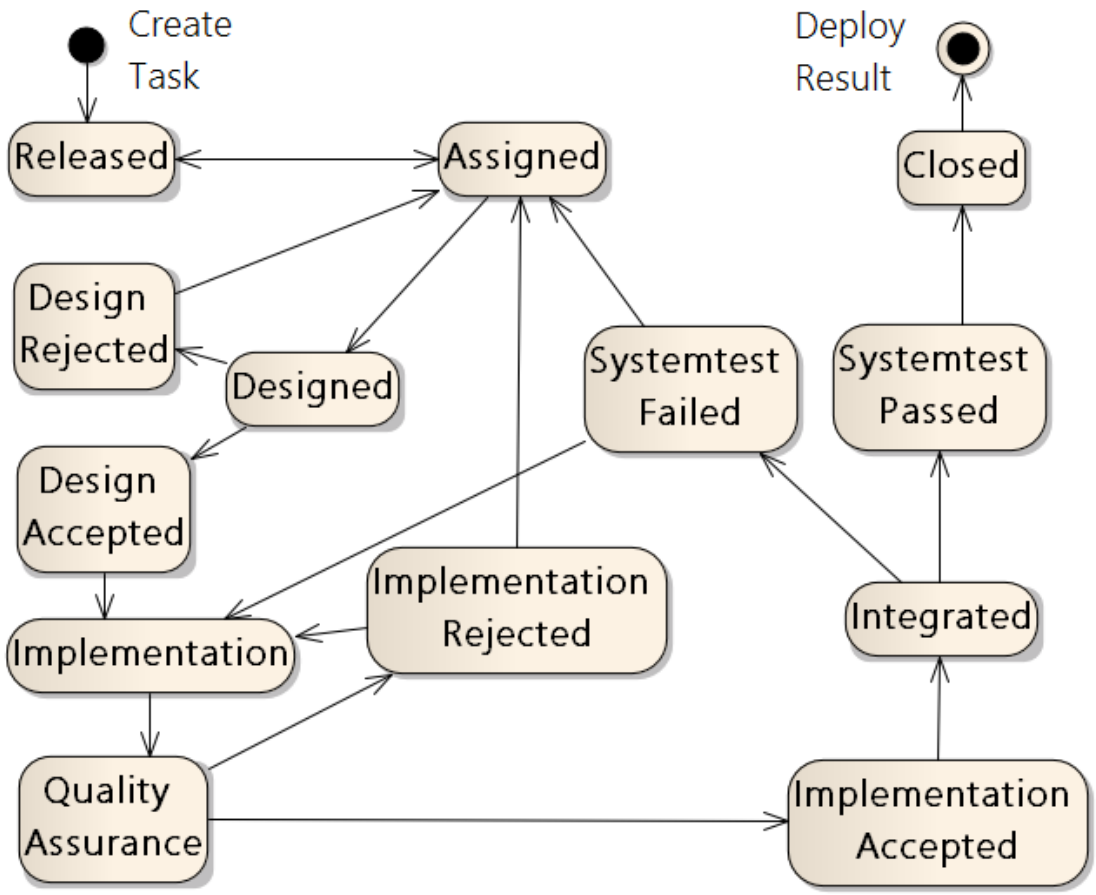

Figure 17: Transitionen der Ticketstatus

7 Kriterienkatalog für Issue-Tracking-Systeme

In diesem Kapitel werden die Kriterien für die Bewertung von verschiedenen Issue-Tracking-Systemen vorgestellt. Für alle Anforderungen die innerhalb der Kriterien formuliert sind gilt, dass sie auch als erfüllt angesehen werden, falls eine stabile Erweiterung für das jeweilige System existiert, die die Anforderung erfüllt

7.1 KO-Kriterien

In diesem Abschnitt werden die KO-Kriterien präsentiert. Jedes Issue-Tracking-System das für eine Empfehlung in Frage kommt, muss bei jedem dieser KO-Kriterien mit mehr als 0 Punkten abschneiden. Anhand dieser KO-Kriterien wird in Kapitel 8.2.1 eine Voruntersuchung durchgeführt. Alle Issue-Tracking-Systeme, die mindestens ein KO-Kriterium nicht erfüllen, werden für eine detaillierte Evaluation und eine mögliche Empfehlung nicht mehr in Betracht gezogen.

7.1.1 Betriebskosten (BETR)

Beschreibung Bei diesem Kriterium geht es darum, die Issue-Tracking-Systeme nach ihren Betriebskosten zu bewerten

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
BETR001	Die jährlichen Betriebskosten für das Issue-Tracking-System dürfen 12000 EUR für 200 Benutzer nicht übersteigen. Ein einmalige Anschaffungspreis für das System wird hierbei über 5 Jahre als jährliche Kosten verteilt

Punktevergabe

Vergebene Punkte	Anforderungen
2	BETR001 erfüllt
0	BETR001 nicht erfüllt

7.1.2 Remote-Schnittstelle (RMI)

Beschreibung Dieses Kriterium erfasst, ob man mit den Daten aus dem Issue-Tracking-System über eine Remote-Schnittstelle interagieren kann

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
RMI001	Das Issue-Tracking-System muss eine REST oder SOAP-Schnittstelle anbieten

Punktevergabe

Vergebene Punkte	Anforderungen
2	RMI001 erfüllt
0	RMI001 nicht erfüllt

7.1.3 Definition eigener Tickettypen und Datenfelder (TTD)

Beschreibung Dieses Kriterium erfasst, ob man im Issue-Tracking-System eigene Datenfelder und Tickettypen definieren kann.

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
TTD001	Es muss möglich sein, eigene Tickettypen zu definieren und jedem definierten Tickettyp beliebige eigene Datenfelder zuzuordnen

Punktevergabe

Vergebene Punkte	Anforderungen
2	TTD001 erfüllt
0	TTD001 nicht erfüllt

7.1.4 Integrationsmöglichkeit in Entwicklungsumgebungen (ENTW)

Beschreibung Mittels dieses Kriteriums wird untersucht, ob sich ein Issue-Tracking-System über Entwicklungsumgebungen benutzen lässt

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
ENTW001	Für das Issue-Tracking-System muss eine stabile Erweiterung verfügbar sein, die die Nutzung des Systems über Eclipse und Visual Studio ermöglicht
ENTW002	Für das Issue-Tracking-System sollte eine stabile Erweiterung verfügbar sein, die die Nutzung des Systems über Netbeans ermöglicht

Punktevergabe

Vergebene Punkte	Anforderungen
2	ENTW001 und ENTW002 erfüllt
1	Nur ENTW001 erfüllt
0	ENTW001 nicht erfüllt

7.1.5 Definition eigener Ticket-Workflows (WKF)

Beschreibung Mittels dieses Kriteriums wird untersucht, ob das Issue-Tracking-System die Definition eigener Workflows je Tickettyp zulässt

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
WKF001	Es müssen eigene Workflows für alle definierten Tickettypen erstellt werden können
WKF002	Das Issue-Tracking-System sollte die Möglichkeit zur graphischen Modellierung von Workflows anbieten

Punktevergabe

Vergebene Punkte	Anforderungen
2	WKF001 und WKF002 erfüllt
1	Nur WKF001 erfüllt
0	WKF001 nicht erfüllt

7.1.6 Multi-Project Fähigkeit (MPF)

Beschreibung Mittels dieses Kriteriums wird untersucht, ob über eine Instanz des Issue-Tracking-Systems mehrere Projekte parallel verwaltet werden können

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
MPF001	Das Issue-Tracking-System muss mehrere Projekte parallel verwalten können

Punktevergabe

Vergebene Punkte	Anforderungen
2	MPF001 erfüllt
0	MPF001 nicht erfüllt

7.1.7 Integration mit Quellcode-Repositories (QCR)

Beschreibung Mittels dieses Kriteriums wird untersucht, ob ein Quellcode-Repository in das Issue-Tracking-System eingebunden werden kann

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
QCR001	Es muss möglich sein, in das Issue-Tracking-System sowohl SVN, Git als auch Mercurial Repositories einzubinden. Dabei muss es möglich sein, mit dem Issue-Tracking-System durch das Repository zu navigieren, als auch Commits mit Tickets zu verknüpfen

Punktevergabe

Vergebene Punkte	Anforderungen
2	QCR001 erfüllt
0	QCR001 nicht erfüllt

7.2 Sonstige relevante Kriterien

7.2.1 Duplikaterkennung (DUP)

Beschreibung Mittels dieses Kriteriums wird untersucht, ob das Issue-Tracking-Systems Funktionalität anbieten, die das effiziente Erkennen von Duplikaten unterstützt, sodass ein mehrfachen Anlegen von Tickets für dasselbe Problem vermieden wird.

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
DUP001	Das Issue-Tracking-System sollte den Benutzer beim Anlegen von Tickets auf potentiell ähnliche Tickets hinweisen
DUP002	Das Issue-Tracking-System sollte über ein integriertes Duplikatmanagement verfügen: Anzahl der Duplikate zu einem Ticket erfassen und automatisches Verlinken von Duplikaten zum Originalticket

Punktevergabe

Vergebene Punkte	Anforderungen
2	DUP001 und DUP002 erfüllt
1	Nur DUP001 erfüllt
0	DUP001 und DUP002 nicht erfüllt

7.2.2 Erweiterbarkeit durch Plugins (PLUG)

Beschreibung Mittels dieses Kriteriums wird untersucht, ob das Issue-Tracking-System eine Plugin-Architektur anbietet, die das Einhängen eigener Erweiterungen ermöglicht.

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
PLUG001	Das System sollte die Möglichkeit anbieten, die Funktionalität des Systems durch die Entwicklung von Plugins zu erweitern
PLUG002	Für das System sollte eine sehr gute Dokumentation der APIs zur Verfügung stehen

Punktevergabe

Vergebene Punkte	Anforderungen
2	PLUG001 und PLUG002 erfüllt
1	PLUG001 erfüllt und PLUG002 nicht erfüllt
0	PLUG001 nicht erfüllt

7.2.3 Queries und Datenfilter (QFIL)

Beschreibung Mittels dieses Kriteriums wird untersucht, welche Möglichkeiten ein Issue-Tracking-System zum gezielten Auffinden von relevanten Tickets für einzelne Benutzer und Gruppen anbietet

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
QFIL001	Das System sollte die Möglichkeit anbieten, über seine grafische Benutzerschnittstelle Filter für Tickets zusammenzubauen und für Benutzer und Gruppen zu speichern .
QFIL002	Das System sollte eine eingebaute Query-Language für die Definition komplexer Filter verfügen

Punktevergabe

Vergebene Punkte	Anforderungen
2	QFIL001 und QFIL002 erfüllt
1	QFIL001 oder QFIL002 erfüllt
0	QFIL001 nicht erfüllt

7.2.4 Definition eigener Dashboards (DASH)

Beschreibung Mittels dieses Kriterium wird untersucht, ob Benutzer sich Inhalte aus dem Issue-Tracking-System individuell auf einer persönlichen Seite (Dashboard) zusammenstellen kann.

Beschreibung

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
DASH001	Das Issue-Tracking-System sollte dem Benutzer die Möglichkeit bieten, sich ein personalisiertes Dashboard mit für ihn relevanten Inhalten zusammenzustellen

Punktevergabe

Vergebene Punkte	Anforderungen
2	DASH001 erfüllt
0	DASH001 nicht erfüllt

7.2.5 Nutzung über mobile Plattformen (MOB)

Beschreibung Mittels dieses Kriteriums wird untersucht, ob das Issue-Tracking-System Unterstützung für mobile Endgeräte bietet

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
MOB001	Für das Issue-Tracking-System sollte entweder eine spezielle App oder ein angepasstes Web-Interface für mobile Endgeräte angeboten werden

Punktevergabe

Vergebene Punkte	Anforderungen
2	MOB001 erfüllt
0	MOB001 nicht erfüllt

7.2.6 Altdatenübernahme aus bestehenden Systemen (ALTD)

Beschreibung Mittels dieses Kriteriums wird untersucht, ob das Issue-Tracking-System die Übernahme aus bestehenden Daten anderer Issue-Tracking-Systeme erlaubt

Anforderungen des Industriepartners

Anforderungskürzel	Beschreibung
ALTD001	Das System sollte bestehende Trac-Projekte importieren können

Punktevergabe

Vergebene Punkte	Anforderungen
2	ALTD001 erfüllt
0	ALTD001 nicht erfüllt

8 Evaluierung ausgewählter Systeme

8.1 Bewertungsschema

In diesem Abschnitt wird die Methodik erläutert, nach dem die abschließende Bewertung der Issue-Tracking-Systeme erfolgt. Hierbei werden zunächst Begriffe definiert, die für die Vergabe einer Gesamtpunktzahl relevant sind.

8.1.1 Definitionen

ITS_N	Issue-Tracking-System mit dem Namen N
k_j	Evaluierungskriterium j
$\gamma(k_j) \in [0,5]$	Gewicht von Evaluierungskriterium j
$P_{max}(k_j)$	Maximal erreichbare Punktzahl für Kriterium k_j
$P(ITS_N, k_j) < P_{max}(k_j)$	Erreichte Punktzahl von ITS_N unter k_j
$P_{\gamma max}(k_j) = \gamma(k_j) * P_{max}(k_j)$	Maximal erreichbare gewichtete Punktzahl von k_j
$P_\gamma(ITS_N, k_j) = \pi(ITS_i, k_j) * \gamma(k_j)$	Gewichtete Punktzahl von ITS_N unter k_j
$P_{Gesamt}(ITS_N) = \sum_{\forall k_j} P_\gamma(ITS_N, k_i)$	Erreichte Gesamtpunktzahl von ITS_N
$P_{max} = \sum_{\forall k_j} \pi_{\gamma max}(k_j)$	Maximal erreichbare gewichtete Punktzahl
$Q_{ITS_N} = \frac{P_{Gesamt}(ITS_N)}{P_{Max}}$	Gewichteter Quotient von ITS_N

Table 3: Bewertungsdefinitionen

8.1.2 Gewichtung der Kriterien

j	Kriterium k_j	KO	$\gamma(k_j)$	$\pi_{\gamma max}(k_j)$
1	Betriebskosten (BETR)	X	5	10
2	Remote-Schnittstelle (RMI)	X	5	10
3	Definition eigener Tickettypen und Datenfelder (TTD)	X	5	10
4	Integrationsmöglichkeit in Entwicklungsumgebungen (ENTW)	X	5	10
5	Definition eigener Ticket-Workflows (WKF)	X	5	10
6	Multi-Project Fähigkeit (MPF)	X	5	10
7	Integration mit Quellcode-Repositories (QCR)	X	5	10
8	Duplikaterkennung (DUP)		4	8
9	Erweiterbarkeit durch Plugins (PLUG)		4	8
10	Queries und Datenfilter (QFIL)		3	6
11	Definition eigener Dashboards (DASH)		2	4
12	Nutzung über mobile Plattformen (MOB)		2	4
14	Altdatenübernahme aus bestehenden Systemen (ALTD)		1	1

Table 4: Gewichtung der Kriterien

8.2 Evaluierung der KO-Kriterien

In einer ersten Phase wurden die Issue-Tracking-Systeme zunächst darauf hin untersucht, ob sie alle KO-Kriterien erfüllen. Ein Issue-Tracking-System das bei einem dieser Kriterien 0 Punkte erhalten hat, wird von der weiteren Untersuchung ausgeschlossen. Falls ein System bereits ein KO-Kriterium nicht erfüllt hat, wurden weitere Kriterien für dieses System nicht mehr untersucht (angezeigt durch "-" in der jeweiligen Zelle der Tabelle)

$\pi(ITS_i, k_j) * \gamma(k_j)$	**BETR**	**RMI**	**TTD**	**ENTW**	**WKF**	**MPF**	**QCR**		**Vorläufiger Punktestand**
Bugzilla	10	-	0	-	-	-	-		-
Trac	10	-	-	-	-	0	-		-
Redmine	10	10	10	5	5	10	10		60
Bloodhound	10	0	-	-	-	-	-		-
Clarizen	0	-	-	-	-	-	-		-
Gemini	10	10	10	0	10	10	10		-
OnTime	0	-	-	-	-	-	-		-
DevSuite	5	-	-	0	-	-	-		-
JIRA	10	10	10	10	10	10	10		70
Planbox	5	5	-	0	-	-	-		-
MantisBT	10	10	10	5	5	10	10		60
Pivotal Tracker	0	-	-	-	-	-	-		-

Table 5: Evaluierung der KO-Kriterien

8.3 Evaluierung der erweiterten Kriterien

In der zweiten Phase wurden die verbleibenden Issue-Tracking-Systeme anhand der erweiterten Kriterien vollständig evaluiert.

$\pi(ITS_i, k_j) * \gamma(k_j)$	**DUP**	**PLUG**	**QFIL**	**DASH**	**MOB**	**ALTD**		**Endgültiger Punktestand**
JIRA	8	8	6	4	4	1		$70 + 31 = 101$
Redmine	4	4	3	4	4	1		$60 + 20 = 80$
MantisBT	4	4	3	0	4	0		$60 + 15 = 75$

Table 6: Evaluierung der erweiterten Kriterien

8.4 Das Siegersystem

Nach der Evaluierung hat sich durch JIRA von Atlassian ein klarer Sieger herauskristallisiert. Es erfüllt alle vom Industriepartner gestellten Anforderungen. Ebenso kann der Workflow des Industriepartners darin ohne Einschränkungen abgebildet werden.

9 Empfehlung an den Industriepartner

Angesichts des deutlichen Ergebisses der Evaluierung empfehlen wir dem Industriepartner das Issue-Tracking-System JIRA der Firma Atlassian einzusetzen. Es erfüllt alle gestellten Anforderungen vollständig und bietet darüber hinaus besondere Funktionalität an, die durch die Anforderungen nicht oder nicht in dem Maße gefordert war:

Sehr feingranulare Anpassbarkeit Durch die Anforderung TTD001 wurde zwar ein gewisses Maß an Anpassbarkeit gefordert, doch JIRA bietet weitaus mehr an. So gibt es ein sehr gute Unterstützung bei der Definition eigener Datenfelder durch das System und eine große Zahl an Datentypen und entsprechenden Eingabeelementen stehen zur Auswahl. Es können für einzelne Ticketzustände und Tickettypen eigene Eingabemasken definiert werden, die nur für den Zustand relevante Daten anzeigen und nur gültige Übergänge in andere Zustände zulassen.

Ansprechender Workflow-Editor In der neuesten Version bringt JIRA einen mächtigen Workflow-Editor auf HTML5 Basis mit, der vollständig über die Web-Schnittstelle von JIRA bedient werden kann. Die Benutzung des Editors ist intuitiv und benötigt wenig Einarbeitung.

IDE-Integration direkt vom Hersteller Von den untersuchten System ist JIRA das einzige gewesen, für das die IDE-Integration für Eclipse und Visual Studio direkt vom Hersteller geliefert wird und nicht von Drittanbietern.

Der Workflow-Editor kann für den Industriepartner einen Mehrwert darstellen, in dem er die Ausbringung des Task-Workflows in das System unterstützt. Der Industriepartner sollte sich die hohe Anpassbarkeit von JIRA zunutze machen, indem er zustandsbasierte Eingabemasken definiert, um die korrekte Ausführungdes Workflows sicherzustellen.

Es sollte nicht unerwähnt bleiben, dass JIRA auch Schwächen besitzt. Durch die vielen Anpassungsmöglichkeiten entsteht unweigerlich eine gewisse Komplexität bei der Administration der Systems. Auch das Berechtigungskonzept von JIRA ist komplex und erfordert Einarbeitungszeit.

Trotz dieser Schwächen stellt JIRA das mit Abstand beste Gesamtpaket bereit und in der Preisklasse wird man kaum ein vergleichbares System mit ähnlich großem Funktionsumfang finden.

References

[1] Antonio Carzaniga, Alfonso Fuggetta, Richard S. Hall, Dennis Heimbigner, André van der Hoek, , and Alexander L. Wolf. A characterization framework for software deployment technologies. Technical report, Department of Computer Science, University of Colorado, 1998.

[2] Dr. Wolfgang Riggert. *ECM – Enterprise Content Management*, chapter 3.2.1 Workflow. Vieweg+Teubner — GWV Fachverlage GmbH, Wiesbaden 2009, 1th edition, 2009.

[3] Jochen Ludewig und Horst Lichter. *Software Engineering - Grundlagen, Menschen, Prozesse*, chapter 22.4.2. dpunkt Verlag, 1th edition, 2007.

Erklärung

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

Stuttgart, den 10. Dezember 2013

Andreas Rempel

Kai Friedrich

Andreas Gross