DIPLOMARBEIT

zum Thema

Problemmanagement bei Softwareprodukten

Eingereicht von

Christian GRÜBL

Kennzahl J151

Matrikel-Nr.: 9851392

Wien, 2005-06-20

WIRTSCHAFTSUNIVERSITÄT WIEN

DIPLOMARBEIT

Titel der Diplomarbeit:
Problemmanagement bei Softwareprodukten

Verfasserin/Verfasser: GRÜBL, Christian

Matrikel-Nr.: 9851392

Studienrichtung: J151 555

Beurteilerin/Beurteiler: o.Univ. Prof. Dkfm. Dr. Wolfgang H. Janko

Ich versichere:

dass ich die Diplomarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

dass ich dieses Diplomarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/ einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Datum _____ Unterschrift _____

Abstract

Beinahe jeder Mensch hat heutzutage mit Softwareprodukten zu tun und damit verbunden ebenso mit Fehlern oder anderen Problemen der Software. Diese Arbeit richtet sich an Personen, die auf der Seite des Produzenten von Softwareprodukten stehen und eben dort diese Probleme managen sollen.

Dieses Werk zeigt die Grundlagen des Problemmanagements, von der Einordnung in die Information Technology Infrastructure Library bis zu der Grenzziehung zum Change-Management und dem Incident Management, und führt in die Welt von Trouble-Ticket-Systemen ein um anschließend die einzelnen Anforderungen an den Lebenszyklus eines Trouble-Tickets von der Einmeldung durch einen Kunden über die Abstimmung des Problems, deren Umsetzung und Test bis zur Abnahme bzw. Behebungsbestätigung durch den Kunden, zu diskutieren und schlussendlich einen Workflow auszuarbeiten, der alle zuvor diskutierten Anforderungen erfüllt und als Basis für jedes Trouble-Ticket-System dient, das eine dynamische Gestaltung des Workflows zulässt.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Softwareprodukte gehören in der heutigen Zeit ebenso zum normalen Leben wie das Internet, Autos oder Fernseher. Doch genauso sind Fehler oder Mängel in Softwareprodukten nahezu unmöglich auszuschließen. Selbst der zur Zeit größte und mächtigste Softwarehersteller der Welt stolperte vor laufenden Kameras bei einer Produktvorstellung über einen Fehler seines Softwareproduktes.

Wie kann nun ein Softwareproduzent mit solchen Fehlern umgehen. Handelt es sich überhaupt um einen Mangel oder liegt vielleicht ein Erweiterungswunsch vor, der bis dato nicht Umfang des Produktes war? Wie dringend muss dieser Fehler behoben werden, reicht es ihn mit der nächsten Release zu liefern oder ist eine unverzügliche Lieferung von Nöten, bzw. welchen Aufwand stellt die Behebung dieses Mangels dar? Gibt es dafür überhaupt die nötigen Ressourcen oder müssen anderer Arbeiten zurückgestellt werden? Alle diese Fragen stellen nur einen kleinen Ausschnitt aus den vielen Problemfeldern dar, mit denen sich ein Softwareunternehmen konfrontiert sieht, wenn ein neues Problem, ob per Telefon, E-Mail oder Fax, gemeldet wird.

Sämtliche für ein Softwarehaus wichtige Aspekte sollten berücksichtigt werden um den Weg zwischen Meldung des Problems und Lieferung der Problembehebung so kurz als möglich zu gestalten. Dem gegenüber darf natürlich auch die Qualität des Produktes nicht aus dem Auge verloren werden.

1.2 Inhalt der Arbeit

Um den Umfang dieser Arbeit nicht zu sprengen, möchte ich mich wie der Titel bereits besagt auf das Problemmanagement beschränken. Dieses beschäftigt sich mit dem Umgang, der Ursachenforschung und der Abarbeitung von gemeldeten Problemen. Die Grenzziehung zu benachbarten Prozessen wie z.B. dem Change-Management wird nicht immer haarscharf möglich sein, dennoch werde ich Themen wie Änderungsanforderungen, die wohl oft aufgrund von Problemen entstehen, und deren Umsetzung bzw. Implementierung oder die verschiedenen Auslieferungsvorgehen nur insoweit anschneiden als sie eben zur Grenzziehung zum Problemmanagement relevant sind oder für das Verständnis der Problembehandlung unerlässlich sind.

Gleiches soll auch für Softwareprojekte gelten. Auch zu diesem Thema werde ich Abgrenzungen vornehmen, aber ich werde nicht auf die Behandlung oder andere Spezifika von Mängeln im Projekt Stadium eingehen.

Als das zentrale Anliegen dieser Arbeit möchte ich vor allem auf den Lebenszyklus bzw. das Phasenmodell eingehen. Dieses begleitet uns während der gesamten Erledigung eines Problems und beschreibt am besten den Prozess der Problembehebung von der Entstehung bzw. Meldung bis zur endgültigen Beseitigung bzw. bis zur Bestätigung der Erledigung durch den Kunden.

Ich will in der hier vorliegenden Arbeit, sehr detailliert die Anforderungen, die an den Workflow für den angeführten Teilbereich gestellt werden, ausarbeiten, um anschließend zu zeigen, welche Möglichkeiten es gibt diese in einem Tool zu implementieren und welche Vor- bzw. Nachteile verschiedenste Optionen in der einen oder anderen Phase des Behebungsprozesses dem Softwareproduzenten bzw. seinen Mitarbeitern bieten.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich ganz grob in zwei Teile. Einen theoretischen, der versucht die Begriffe abzugrenzen, und einen 2. Teil, der sich um Praxis relevante Aspekte kümmert und auf Basis der zuvor abgegrenzten Begriffe tiefer in die Materie eintaucht.

Nach der Einleitung umfasst der theoretische Teil zwei Kapitel:

Das erste Kapitel erläutert den Begriff Problemmanagement näher. Zunächst werde ich einen kurzen Einblick in die Einordnung dieses Prozesses in das Service-Management geben, sowie im Anschluss den Begriff von seinen Nachbarn im Service-Management, dem Change- wie auch dem Incident-Management abgrenzen, um im zweiten Kapitel der Arbeit den Begriff Softwareprodukt vom Projekt abzugrenzen. Wo beginnt das Produkt und endet das Softwareprojekt bzw. wo sind die Unterschiede und welche Charakteristika besitzen Softwareprodukte im besonderen Kontext zum Problemmanagement, um damit die Basis für den Praxisteil dieser Arbeit zu schaffen. Es soll anhand von konkreten Ansatzpunkten auf die Charakteristika des PM eingegangen werden, um die Notwendigkeit der unterschiedlichen Handhabung von Problemen in Projekten bzw. Produkten erläutern zu können.

Der Rest dieses Werks umfasst den praktischen Teil, wobei das Kapitel 4 ausnahmsweise eine Mischung von Praxis und Theorie darstellt.

In diesem möchte ich das so genannte Trouble Ticket System vorstellen. Ich werde die Aufgaben, die diesen Systemen durch die Literatur zugedacht werden anführen, um dann verschiedene Ausprägungen dieser Systeme zu erläutern und eine Abgrenzung vorzunehmen. Das Ende dieses Kapitels wird einige kostenlos erhältliche und einige kostenpflichtige Systeme, die sich auf dem Markt befinden, vorstellen.

Anschließend folgt das Herzstück dieser Arbeit. Es beschäftigt sich sehr ausführlich mit den Anforderungen der Praxis, die an den Workflow eines Problemmanagement bzw. Trouble Ticket System gestellt werden. Es soll einem verantwortlichen Produktmanager helfen, die Auswahl eines Werkzeuges treffen zu können, dass ihn optimal bei der Verwaltung seiner Probleme unterstützt.

Dieses Kapitel umfasst alle relevanten Abschnitte des Lebenszyklus eines Problems, von der Einmeldung durch einen Kunden oder eines hauseigenen Testers über die Problemabstimmung, die mit einer übereinstimmenden Sicht auf die wichtigsten Punkte des Problems endet, weiter über die Planung, Behebung und den Test des Mangels, sowie den Lieferprozess bis hin zum abschließenden Test des Kunden, der damit bestätigt dass das Problem endgültig behoben ist.

Diese Information bildet die Grundlage um im abschließenden Kapitel ein möglichst vielseitig einsetzbares Phasenmodell zu definieren, dass den zuvor aufgestellten Anforderungen gerecht wird und damit einem Problemmanagement Tool zugrunde liegen sollte, welches in einem Softwarehaus zur Verfolgung von Problemen eingesetzt werden könnte.

2 Grundlagen zu Problem-Management (PM)

Problemen begegnen wir natürlich nicht nur im Umfeld der EDV, ganz allgemein sieht man Probleme als schwierige Aufgabe, die man lösen muss, zu deren Lösung aber weder Mittel noch Weg bekannt sind [Schu96, 11].

In diesem Kapitel werde ich zeigen wie sich das PM in den gesamten Bereich der IT-Services einordnen lässt. Für diesen Bereich hat sich ein branchenübergreifender Leitfaden quasi ein „De - Facto - Standard“ für IT Service Management, die Information Technology Infrastructure Library (ITIL), herauskristallisiert [ITIL2003, 1]. Sie werde ich, nach dem ich sie vorgestellt und die Einordnung des PM aufgezeigt habe, nutzen um den Prozess des PM zu beschreiben und ihn von seinen Nachbarn, dem Incident-Management und dem Change-Management, abzugrenzen. Im weiteren Verlauf dieses Kapitels werde ich noch auf die Ziele, die Aufgaben und die Funktionen des PM näher eingehen.

2.1 IT Service-Management (IT-SM) und ITIL

Um die Einordnung des PM in diesen Bereich aufzeigen zu können, muss zuerst geklärt werden, was man sich unter Service-Management vorstellen kann und aus welchen Teilen es besteht.

„IT Service Management ist eine prozessorientierte Methode für das Management beliebig komplexer, kleiner und großer IT-Services“ [Grun2003, 6]. Eine andere Definition stammt von Kuhlig: „IT Service Management handelt von der Planung und Bereitstellung einer kundenorientierten Dienstleistung (Service) mit Hilfe eines prozessorientierten Verfahren“ [Kuhl2003, 2]. IT-SM stellt eine spezifische Dienstleistung dar, die aus einer Kombination von Produkt und Service besteht [Grun2003, 5]. Der Kunde kauft also nicht nur einfach ein IT-Produkt, sondern auch eine Dienstleistung, die den gesicherten Betrieb des Produktes sichern soll. Das IT-SM soll eine möglichst hohe Kundenzufriedenheit sicherstellen und die beauftragte Qualität mittels definierter und geregelter Prozesse garantieren [Grun2003, 6].

Beinahe untrennbar mit dem Begriff IT Service Management ist der Begriff ITIL in der Literatur zu finden. Anzumerken ist zwar, dass es zahlreiche andere Methoden und Modelle in der Welt des IT-SM gibt [Olbr2004, 2], dennoch hat sich ITIL zu einem anerkannten Standard entwickelt, bei dem es sich um eine Art von Sammlung bzw. Bibliothek von IT-Best-Practices handelt [Bern2003, 63].

ITIL umfasste ursprünglich mehrere 100 Bücher. Mittlerweile ist die Sammlung seit ihrer ersten Veröffentlichung 1989 auf knapp 70 zusammengeschrumpft wobei in der Praxis vorwiegend ein Kern von 6 Büchern herangezogen wird. Die Bibliothek stellt keine verbindliche Norm dar, sondern beschäftigt sich mehr mit der Frage was zu tun ist und nicht wie etwas getan werden muss [Olbr2004, 1].

Für das in dieser Arbeit behandelte Thema ist vor allem der Bereich Service Support, der sich im Gegensatz zum Service Delivery mehr auf die operativen Prozesse ausrichtet, interessant [Olbr2004, 13ff]. Dieser Bereich umfasst laut ITIL die folgenden fünf Kernbereiche oder besser gesagt Prozesse [Elsä2005, 48]:

- Incident-Management
- Problem-Management
- Change-Management
- Release-Management
- Configuration-Management

2.2 Prozesse und Grundbegriffe des PM

Der Hauptzweck des PM ist in erster Linie die Ursachenforschung von Störungen. In der Literatur werden dem PM vorrangig folgenden Themenbereiche zugeordnet [Olbr2004, 35]:

- Problembehandlung
- Fehlerbehandlung
- Unterstützung zur Behebung von schweren Störungen
- Proaktive Störungs- und Problemvermeidung
- Informationsvermittlung und Reporting

Um auf die Themenbereich genauer eingehen zu können, möchte ich vorweg einige Begriffe genauer erläutern [Kuh2003, 30].

Problem

Ein Problem ist die unbekannte Ursache für eine oder mehrere Störungen.

Bekannter Fehler

Ein bekannter Fehler ist die bekannte Ursache eines Problems, wobei zu diesem Zeitpunkt noch keine Lösung bekannt ist.

Workaround

Ein Workaround ist eine Umgehungsmöglichkeit eines Problems, welches dem IM für die Zeit der Behebung als Übergangslösung zur Verfügung gestellt wird, damit nicht sofort behebbare Probleme umgangen werden können.

2.2.1 Ablauf des PM nach ITIL

Nachdem nun die wichtigsten und oft auch leicht verwechselbaren Begriffe klar beschrieben sind, werde ich nun das Phasenmodell des Problemmanagements nach ITIL vorstellen [Olbr2003, 36ff]. Zur leichteren Verfolgbarkeit soll uns die Abbildung 1 dienen.

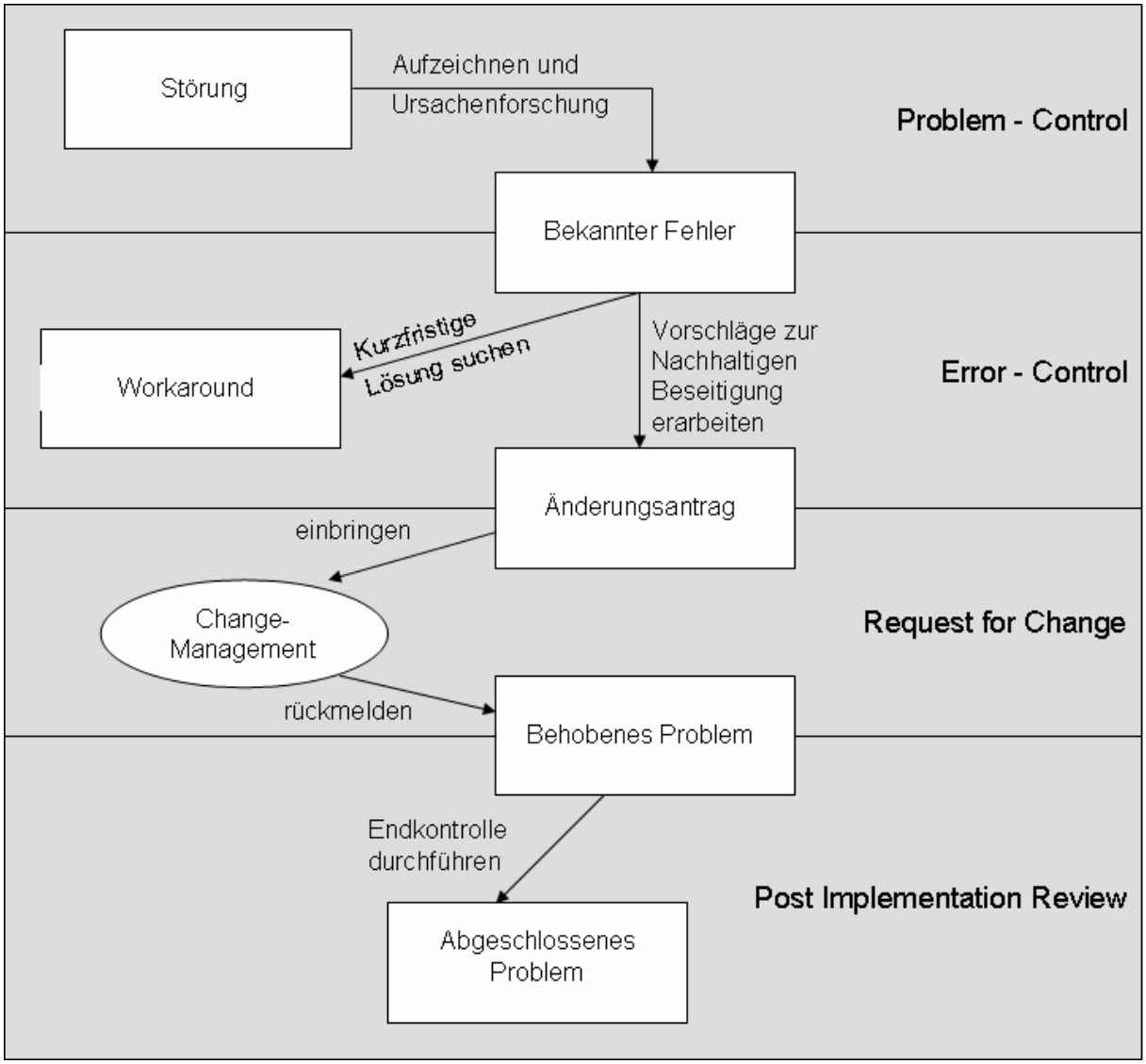


Abbildung 1: Ablauf des PM nach ITIL

Zunächst werden unbekannte Fehler identifiziert und aufgezeichnet. Für diese Aufgabe wird jeweils ein Datensatz in einer Datenbank angelegt. Diesem Datensatz sollen alle in der Zukunft neu gewonnen Erkenntnisse und Informationen, die bis zum Abschluss eines Problems gewonnen werden, angehängt werden. Diese erste Phase nennt ITIL Problem Control. Hauptaufgabe dieser Phase ist die Umwandlung von Problemen in bekannte Fehler mittels Ursachenforschung.

Als nächster Schritt erfolgt die konkrete Fehlerbehandlung. Diese Phase nennt ITIL Error Control. An dieser Stelle werden vorrangig zwei Ziele verfolgt. Einerseits versucht man kurzfristige Lösungsmöglichkeiten zu suchen, um die Fehler zu umgehen, und auf der anderen Seite werden Vorschläge erarbeitet, die zu einer nachhaltigen Beseitigung von Fehlerquellen führen sollen.

Diese Vorschläge resultieren in dem Einreichen von Änderungsanträgen und damit der dritten Phase, dem Request for Change. Die Durchführung und Prüfung auf Durchführbarkeit obliegt dem Change-Management (CM) während das PM auf die Erledigung warten muss.

Den Abschluss eines Problems übernimmt die Phase des Post Implementation Review. Diese stellt eine Endkontrolle der Änderungen durch das CM dar, prüft vor allem die Vollständigkeit der Umsetzung, die ordnungsgemäße Durchführung und fungiert in diesem Zusammenhang als Qualitätssicherungsprozess.

In allen diesen Phasen wird großer Wert auf eine saubere Dokumentation aller Aktionen und Maßnahmen gelegt, um einerseits eventuelle Informationen für zukünftige Probleme leichter zur Verfügung zu haben und auf der anderen Seite durch genaue Analyse der vorhandenen Datensätze möglicherweise das eine oder andere Problem schon prophylaktisch, durch die so genannte proaktive Funktionalität des PM, verhindern zu können.

2.2.2 Fehlermanagement nach Probst

An dieser Stelle möchte ich noch einen anderen Ansatz zum Ablauf von Problemen vorstellen. Er stammt von Probst, der eine noch strengere Trennung von Fehler- und Problemmanagement bevorzugt, wenn gleich er aber einräumt, dass das Fehlermanagement das wichtigste Gebiet des PM ist und deshalb oft im Vordergrund steht. Er bemängelt, dass die beiden Begriffe oftmals als Synonym verwendet werden und erklärt den Zusammenhang wie in Abbildung 2 zu sehen [Prob95, 19].

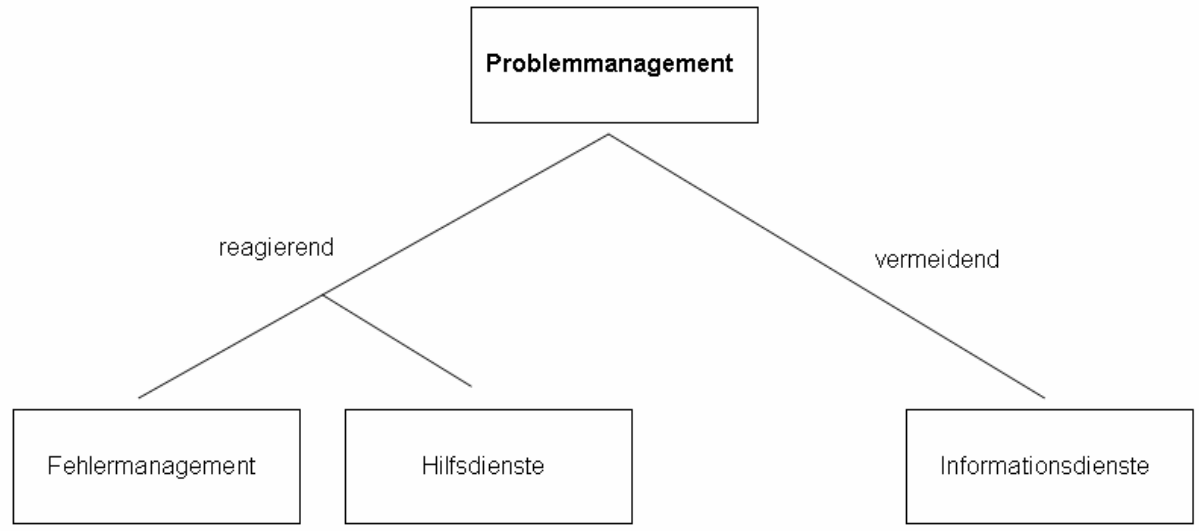


Abbildung 2: Aufgabengebiete des PM

Probst sieht das Fehlermanagement als eine Kernaufgabe des Benutzerservices und gliedert es in 3 Phasen [Prob95, 20ff]:

- Fehlererkennung
- Fehlerdiagnose
- Fehlerbehebung

Die Phase der Fehlererkennung dient zur frühzeitigen Erkennung von Störungen und Problemen. Diese werden entweder durch den Benutzerservice, durch den Anwender, einen Systemadministrator oder auch durch eine Managementplattform erfasst. Es muss darauf geachtet werden, dass alle wesentlichen Daten zu der Störung, zum Störungsmelder und zur betroffenen Systemkomponente aufgenommen werden. Nach der Erfassung erfolgt die Klassifizierung der Störung durch Zuordnung zu einem Sachgebiet. Beendet wird die Fehlererkennung mit der Zuweisung und Benachrichtigung des Zuständigen.

Die Phase der Fehlerdiagnose benützt die erfassten Daten um die Störungsursache zu identifizieren, wobei auf Seiten des Bearbeiters ein fundiertes Problemlösungswissen vorhanden sein muss.

Gelingt die Diagnose der Störung durch den Bearbeiter nicht, so muss die Störungsmeldung an einen Experten weitergeleitet werden. Dieser Vorgang wird Eskalation genannt. Ist die Ursache des Fehlers gefunden, kann die Fehlerbehebung beginnen. Nach erfolgreicher Behebung wird der Störungsmelder über die Behebung der Störung informiert und die Störungsmeldung wird geschlossen. Die Abbildung 3 zeigt den Ablauf noch einmal in graphischer Form [Krus2001, 4].

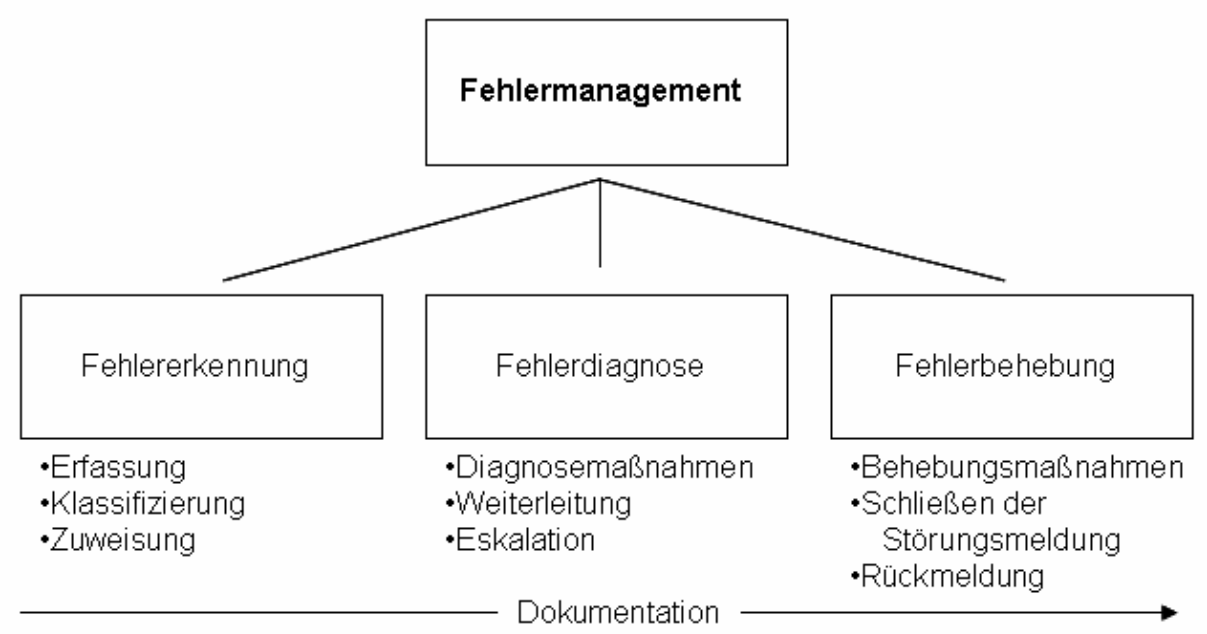


Abbildung 3: Phasen des Fehlermanagements

2.3 Zusammenspiel des PM mit dem Incident und dem Change Management

Man kann also erkennen, dass der Prozess des Problem-Managements in sehr engem Kontakt zu zwei anderen ITIL-Kernbereichen steht. Diese sind:

- Incident-Management
- Change-Management

Die Abbildung 4 zeigt auf einen Blick wie diese drei Begriffe zueinander stehen und welcher Zusammenhang zwischen Störungen, Problemen, bekannten Fehlern und Änderungen besteht [BoKP2002, 60].

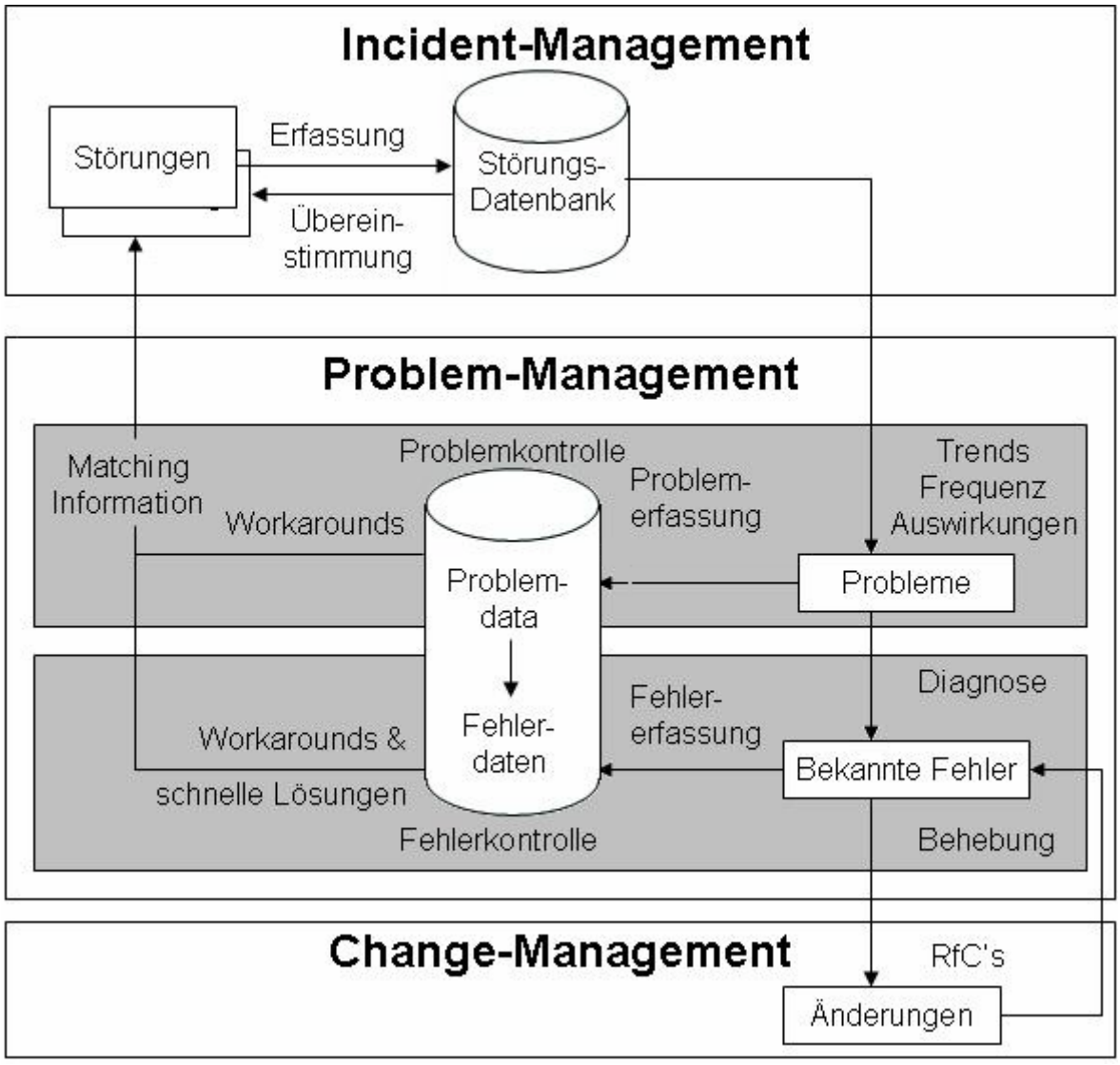


Abbildung 4: Zusammenhang IM, PM und CM

Während also im IM die Störung erfasst wird und bei einer Übereinstimmung mit einer früher bereits aufgetretenen Störung auch geholfen werden kann, überprüft das PM Trends oder Auswirkungen von Störungen um daraus Probleme zu definieren. Einerseits wird das Problem nun erfasst und wenn möglich ein Workaround gefunden, welcher an das IM weitergegeben wird, und auf der anderen Seite wird versucht, mittels Diagnose die Ursache herauszufinden. Das dadurch nun zum bekannten Fehler gewordene Problem wird abermals erfasst, bzw. viel mehr dessen Ursache, eventuell wird ein besseres Workaround oder eine schnelle Lösung gefunden und an das IM zurück gemeldet. Der bekannte Fehler kann seinerseits nun ein oder mehrere Requests of Change auslösen, um damit über das CM den bekannten Fehler zu beheben. Das abschließende Implementation Review ist zwar in der Abbildung nicht sichtbar, es bildet aber dennoch, wie bereits erwähnt, den Abschluss eines Problems. Die beiden nächsten Kapitel (2.4 und 2.5) sollen noch einmal im Detail die Abgrenzungen zu den beiden Nachbarprozessen des PM ausführen.

2.4 Abgrenzung des PM zum Incident-Management (IM)

Im Rahmen des Incident-Managements werden Maßnahmen ergriffen, wenn sich eine Störung eingestellt hat. Ist die ursprüngliche Situation wiederhergestellt, so sind diese Maßnahmen nicht länger erforderlich. Da die Ursachen der Störungen nicht immer ergründet werden, besteht die Gefahr, dass die Störungen immer wieder auftreten. Im Gegensatz dazu untersucht das PM die Infrastruktur und versucht die Ursachen für potenzielle Störungen zu bestimmen, es nimmt sich also die Zeit, die Ursache zu ergründen und zu beseitigen, während das Incident-Management in erster Linie an einer raschen Behebung der Störungen interessiert ist [BoKP2002, 59].

Die Einführung ins FITS Problem Management, die die British Educational Communications and Technology Agency zur Verfügung stellt, spricht von fünf Unterschieden zwischen PM und IM [PrMa2003, 1]:

- Ziel des IM ist es, die Arbeitsfähigkeit eines Systems so schnell als möglich wiederherzustellen, oft eher mit einer Umgehungsvariante als dem Versuch eine permanente Lösung zu finden.
- PM unterscheidet sich vom IM vor allem darin, dass für das PM als Hauptziel die Entdeckung der Ursache einer Störung anzusehen ist. Die beste Lösung

und ein langfristiger Schutz vor einem erneuten Auftreten der Störung stehen im Vordergrund.

- In vielen Situationen können die Ziele der beiden in direktem Konflikt zueinander stehen
- Die Entscheidung welcher Lösungsvorschlag verfolgt wird, erfordert eine gründliche Überlegung. Ein sinnvoller Lösungsvorschlag wäre, das System so schnell als möglich wieder herzustellen (=IM), aber gleichzeitig sicher zu stellen, dass alle Details erfasst werden. Dies ermöglicht dem PM seine Arbeit aufzunehmen, sobald ein workaround implementiert wurde.
- Disziplin ist notwendig, da in der Regel das Fixen von Incidents im Vordergrund steht. Wie auch immer, der Incident wird wieder auftreten, wenn die Lösung des Problems nicht gefunden wird.

Es wird also oft der Fall sein, dass neben einer Störung auch ein Problem definiert wird, um ein erneutes Auftreten zu verhindern. Es ist aber beispielsweise auch durchaus möglich, dass einem Problem mehrere Fehler zu Grunde liegen oder dass mehrere Problemdefinitionen auf ein und denselben Fehler zurückzuführen sind [BoKP2002, 59].

2.5 Abgrenzung des PM zum Change-Management (CM)

Hat man die Ursache eines Problems herausgefunden, so ist es nicht selten der Fall, das eine Änderung an der Software von Nöten ist. An diesem Punkt nähern wir uns nun der Grenze zum Change-Management. "Das Change-Management ist für die kontrollierte Durchführung von Änderungen, einschließlich der Änderungsanträge, die das Problem-Management vorlegt, zuständig, um strukturelle Fehler zu beseitigen" [BoKP2002, 62].

Die Einführung ins FITS Change Management von der British Educational Communications and Technology Agency führt als Ziel des CM die Vermeidung von Incidents und Problems mit Hilfe effektiver Planung an [ChMa2003, 2].

Ein Change kann also sowohl aus einem Lösungsvorschlag eines Problems resultieren als auch aus ganz anderen Gründen entstehen [ChMa2003, 1]. Ein gutes Beispiel für einen solchen Fall wäre hierfür die nachträgliche Änderung der Software aufgrund eines Kundenwunsches oder Änderungen aufgrund neuer Hardware Anforderungen.

2.6 Zielsetzung des PM

In diesem Kapitel sollte sehr klar herausgekommen sein, dass nur die Vermeidung beziehungsweise die Reduktion von Störungen als das oberste Ziel gelten kann. Zur Erreichung dieses Zieles werden, wie bereits erwähnt, sowohl proaktive als auch reaktive Aktivitäten ausgeführt.

Bon, Kemmerling und Pondman schreiben zu diesem Thema: „Im Rahmen des reaktiven Problem-Management sucht man nach der Ursache für bereits eingetretene Störungen und initiiert daraufhin Vorschläge zur Verbesserung beziehungsweise Korrektur der Situation. Proaktives Problem-Management versucht, Störungen zu verhindern bevor sie zum ersten Mal auftreten, indem Schwachstellen in der Infrastruktur identifiziert und Vorschläge zu deren Beseitigung geprüft werden“ [BoKP2002, 60].

Welche Aufgaben ergeben sich nun, um die Zielsetzung bestmöglich verfolgen zu können? In der Literatur sind immer wieder die Schlagwörter lokalisieren, dokumentieren und verhindern zu finden.

Diese Schlagwörter zeigen schon sehr gut, welche Richtung das PM einschlägt. Fehler müssen natürlich aufgefunden werden und sollen auch auf lange Sicht gesehen vermieden werden, aber dennoch ist es unumgänglich die Symptome und kurz- wie auch langfristige Lösungsvorschläge, die in der Vergangenheit bei der Problembehebung auftraten bzw. ausgearbeitet wurden, zu dokumentieren und zu protokollieren.

Martin G. Bernhard definiert das Ziel des PM als die Stabilisierung der IT-Service durch [Bern2003, 72]:

- Minimieren der Folgen von Incidents
- Beseitigung der eigentlichen Ursachen von Incidents
- Vermeidung sowohl von Incidents als auch Problemen
- Verbesserung der produktiven Nutzung von Ressourcen

3 Abgrenzung des Begriffs Softwareprodukt

Dieses Kapitel beschäftigt sich in erster Linie damit, die Unterschiede beim Management von Problemen zwischen einem Softwareprodukt und einem Softwareprojekt herauszuarbeiten. Die Grenzziehung zwischen den beiden Begriffen ist noch einigermaßen einfach.

Den großen Unterschied bestimmt der Faktor Zeit. Während ein Projekt zeitlich begrenzt ist, wird ein Produkt in die Welt gesetzt und endet, wenn es für Tod erklärt wird [Zveg87]. Blazey unterscheidet in ihrem Werk zwischen Entwicklungsprojekten und Wartungsprojekten und verweist zwecks genauerer Definitionen auf [Wald97]. Sie definiert ein Projekt als einzigartiges Vorhaben mit klarem Anfangs- und Endtermin und zeigt damit ebenfalls den großen Unterschied zum Produkt auf, welches ja kein definiertes Ende aufweist [Blaz2000, 21].

„In diesem unvorhersehbaren Zeitabschnitt zwischen Geburt und Tod eines Softwareprodukts finden viele Projekte statt. Jeder Zustandswechsel des Produkts ist das Resultat eines Projekts. Je nachdem wie viele Zustandsübergänge man zulässt, gibt es mehr oder weniger viele Projekte. Sie können hintereinander und auch nebeneinander ablaufen. Das Produktmanagement plant, organisiert und steuert sie. So gesehen ist Produktmanagement gleich einem Projekt der Projekte. Es setzt die Ziele der untergeordneten Projekte, stößt sie an und kontrolliert, ob sie am Ziel ankommen“ [HaTS2004, 3].

Neben dem Faktor Zeit ist aber für diese Arbeit der Umstand des Echteinsatzes von viel größerer Bedeutung.

Befindet sich ein Stück Software bereits in einem Stadium nach der erfolgreichen Inbetriebnahme durch zumindest einen Kunden, so spreche ich in diesem Kontext von einem zu wartenden Softwareprodukt. Man könnte auch sagen, das erste große Projekt der Projekte ist abgeschlossen und da die Arbeiten mit diesem Echteinsatz nicht ebenso abgeschlossen sind, zieht die Inbetriebnahme den Start einiger neuer Projekte, wie zum Beispiel Wartung oder Weiterentwicklung des Produktes bzw. die Akquirierung von Neukunden, nach sich, wie das ja die Definition des Produktmanagements verlangt.

Warum ist diese Unterscheidung nun aber für das Problemmanagement so wichtig und welche Probleme ergeben sich aus der Tatsache, dass dieses Stück Software

bereits im Echteinsatz ist? Diese Fragen sollen nun aufgeteilt auf einige Teilaspekte geklärt werden.

3.1 Risiko im Echtbetrieb

Das Thema Risiko im Echtbetrieb stellt sich in diesem Zusammenhang sehr einfach dar. Es liegt auf der Hand, dass Mängel im Produktivbetrieb viel problematischer und gefährlicher sind, als wenn diese in einem Testsystem auftreten, denn ein Systemabsturz in der Testabteilung ist wohl nicht mit einem Absturz in z.B. einer Bank, in der womöglich auch noch zig Kunden auf die eine oder andere Dienstleistung warten, zu vergleichen.

Aber nicht nur die Gefahr des Auftretens eines sehr schwerwiegenden Mangels ist enorm, viel peinlicher wird es für einen Softwareproduzenten, wenn durch die Behebung eines eigentlich geringen Mangels ein weitaus schwerwiegenderes und verheerendes Problem entstehen würde, welches ursprünglich gar nicht aufgetreten wäre. Möglicherweise hat dieses dann auch noch Auswirkungen auf Kunden, für die der geringere Mangel nicht einmal bestanden hatte.

Zur Illustration ein kleines Beispiel:

Der Kunde A meldet einen Incident zu einem Unterdialog D. Dieser Dialog ist mit der Beschriftung „Warnung“ versehen, sollte aber das Wort „Achtung“ in der Kopfzeile beinhalten. Dieser Incident ist zugegebenermaßen nicht sehr schwerwiegend und beeinträchtigt den Echtbetrieb nicht, es gibt aber dennoch kein workaround. Die Ursache wird erforscht und ein als gering eingestuftes Problem akzeptiert. Nach genauer Ursachenforschung wird der tatsächliche Fehler im Aufruf des Unterdialogs gefunden, worauf der zuständige Programmierer diesen ändert, ohne zu beachten, dass dadurch alle anderen Unterdialoge, die von diesem Softwarestück aufgerufen werden, gar nicht mehr funktionieren. Es ist nicht einmal mehr möglich, diese zu öffnen. Würden wir uns jetzt vor dem Zeitpunkt der Inbetriebnahme befinden, würde ein nur sehr geringer Schaden entstehen. Beim nächsten Test der anderen Unterdialoge wird das Problem wahrscheinlich entdeckt und behoben. Ist die Live-Setzung bereits erfolgt, wird aber das Einspielen der neuen Software einen viel größeren Schaden verursachen. Ein Rückstieg auf den alten Stand wäre wohl unausweichlich und ein Stillstand des Betriebs für mehrere Stunden sicher. Weiters wären wahrscheinlich auch noch alle anderen Kunden dieses Produktes von der fehlerhaften Behebung betroffen, obwohl sie den Unterdialog D vielleicht gar nicht

zur Verfügung haben und somit den ursprünglichen geringen Mangel nie gemeldet hätten.

Dieses Beispiel veranschaulicht also sehr eindrucksvoll, wie viel größer das Risiko ist, Schaden welcher Art auch immer davon zu tragen. Nicht zu Ende gedachte oder oft einfach falsche Lösungsansätze bzw. das Nichtbedenken von Auswirkungen auf andere Softwareteile, können nie geahnte Auswirkungen haben. Darum sei auch an dieser Stelle angemerkt, wie wichtig es ist sich dieses Risikos bewusst zu sein und dementsprechende Sicherheitsvorkehrungen zu treffen. Ein zusätzlicher Gesamttest vor der Lieferung oder ähnliches hätten in unserem Beispiel den Schaden wohl deutlich reduziert.

3.2 Datenproblematik

Auch die Datenproblematik bei Softwareprodukten ist sehr leicht nachvollziehbar bzw. erklärbar. Befindet sich die Software im Entwicklungsstadium, so werden in der Regel bestenfalls Daten verwendet, die den Testern zur Unterstützung ihrer Tests bereitgestellt werden oder solche, die durch ihre Tests entstanden sind.

Werden durch den Test Datenbestände zerstört, so reicht es aus, dass für den Fall, dass die Daten von einem Kunden zur Verfügung gestellt wurden, auf genau diesen Stand zurück gestiegen wird. Die in der Zwischenzeit erfassten Testdaten wie auch Testdaten, die von Anbeginn im Testbetrieb entstanden sind, gehen meist verloren oder werden von alten Sicherungen zurückgespielt. Dies stellt jedoch in den seltensten Fällen ein Problem dar, da die Daten zwar für die Durchführung von dem einen oder anderen Testfall wichtig sind, jedoch werden die Daten nicht für andere Zwecke benötigt. Nicht der Inhalt der Daten ist wichtig, sondern deren korrekte Befüllung. Es ist beispielsweise egal, ob im Test einer Bankensoftware auf einem Konto 10 Euro oder 10000 Euro aufscheinen, nur das Zustandekommen dieser Summe muss korrekt sein, da das Geld ja ohnehin nicht real existiert.

Sprechen wir aber von einem bereits im Einsatz befindlichen Produkt, wo von der Richtigkeit des Datenbestandes tausende Kunden abhängig sein könnten, so ist ein durch einen Mangel verursachter Datenfehler natürlich nicht so einfach durch das Einspielen eines alten Standes oder gar die Löschung der fehlerhaften Daten behebbar. Es müssen oft neben der Lieferung der Problembehebung auch entsprechende Update Statements angeboten werden, um den fehlerhaften Datenbestand in einer Datenbank zu korrigieren.

Auch diesen Fall möchte ich anhand eines Beispiels veranschaulichen:

Der Speicher-Button im Dialog D soll den Namen, die Anschrift, die Festnetztelefonnummer und die Handynummer eines Kunden in der Tabelle T in den Attributen „N“, „A“, „F#“ und „H#“ abspeichern. Durch einen Programmierfehler werden aber bei der Neuanlage von Kunden die Handynummern in „F#“ und die Festnetznummern in „H#“ abgelegt. In einem Stadium vor dem Echteinsatz dieser Software könnte man nun einfach den Fehler beheben, die Tabelle löschen und mit neuerlichen Tests beginnen. Ganz anders sieht die Situation bei unserem Kunden A aus. Dieser hat natürlich kein Interesse alle bis zu diesem Zeitpunkt angelegten Kunden-Einträge neu anzulegen, sondern wird neben der Behebung des Speicherbuttonproblems auch eine Behebung des verursachten Datenschadens verlangen.

Man kann also erkennen, dass die Problembehebung in Softwareprodukten um einiges umfangreicher und schwieriger ist, da neben der neuen Software ja auch ein Programm zur Richtigstellung des Datenbestandes bereitgestellt werden muss.

Doch kann natürlich genauso der Fall eintreten, dass die Erstellung eines solchen Korrekturprogramms gar nicht mehr den ganzen Datenbestand richtig stellen kann. Das Risiko, dass Daten vollkommen verloren gehen, darf in diesem Zusammenhang nicht außer Acht gelassen werden. Es ist ja möglich, dass ein fehlerhaftes Programm Datensätze löscht, überschreibt oder irreparabel verändert. Auch dieser Punkt soll nochmals verdeutlichen wie groß der Unterschied zwischen Produkt und Projekt ist und welche Bedeutung die Unterscheidung für die Auswirkung von Mängeln hat, denn natürlich ist das Verlorengehen von Testdaten ein verhältnismäßig zu vernachlässigendes Problem, wenn man es dem wirtschaftlichen Schaden eines Betriebes gegenüberstellt, dass einen Großteil seines Datenbestandes, sofern überhaupt möglich, neu erfassen muss oder im aller schlimmsten Fall gespeichertes Wissen oder andere Daten für immer verliert.

3.3 Lieferungen

Ein weiterer großer Unterschied liegt in dem Umstand, dass in regelmäßigen Abständen Behebungen von Problemen an die jeweiligen Kunden übermittelt werden müssen.

Während eines Softwareprojekts ist es verhältnismäßig einfach Behebungen zu erledigen. Man ändert ein paar Zeilen Code, kompiliert diesen, spielt die geänderten Module in das Test-System ein, und die veränderte Funktionalität steht zur Verfügung. Natürlich gibt es für diesen Vorgang verschiedene Vorgangsmöglichkeiten in beliebiger Komplexität, aber da eine detailliertere Beschreibung nicht notwendig ist, um die Unterschiede zum Softwareprodukt herauszuarbeiten und um an dieser Stelle den Rahmen nicht zu sprengen, möchte ich hier auf ein tieferes Eintauchen in diese Materie verzichten.

Wie sieht dieser Vorgang nun bei einem Produkt aus? Es gibt wohl keinen Käufer von EDV-Programmen auf dieser Welt, der dem Programmierer eines Softwareproduzenten Zugang zu seinem System gewährt, um dort jede gerade erledigte Behebung eines Mangels sofort ins Echtsystem einzuspielen. Es wird wohl auch kein Kunde sehr erfreut sein, wenn er jeden Tag eine Vielzahl von verschiedensten Mangelbehebungen erhält und diese möglichst sofort ins System einspielen sollte, da womöglich die einzelnen Lieferungen auch noch aufeinander aufbauen und er somit die Behebungen nicht in beliebiger Reihenfolge einspielen kann. Auf der anderen Seite gibt es wohl keinen Kunden, vor allem wenn es sich um Software handelt, die für diesen lebensnotwendig ist, der sich damit zufrieden geben würde, alle paar Jahre einfach eine neue Release des jeweiligen Stücks Software zu kaufen. Diese Kunden drängen immer auf eine prompte und möglichst sichere Behebung eines Mangels und haben zu diesem Zweck in der Regel meist sehr komplexe Wartungsverträge abgeschlossen, um solchen mehr oder weniger großen Problemen mit den Programmen vorzubeugen.

Meiner Erfahrung nach, sehen Softwareproduzenten die Lösung dieses Problems in einer gesammelten und regelmäßigen Übermittlung von Problembehebungen. In welchen Abständen und in welchem Umfang diese Lieferungen erfolgen ist jedoch unterschiedlich. Dies hängt von der jeweiligen Lieferstrategie eines Unternehmens ab, generell wird man aber sagen können, dass:

- In sehr kurzen Abständen Pakete mit wenigen, dafür aber ausgesprochen dringenden und wichtigen Problembehebungen geliefert werden
- In regelmäßigen längeren Abständen größere Pakete, mit insgesamt nicht ganz so dringenden aber wichtigen Mangelbehebungen, geliefert werden.
- In großen Abständen Pakete mit bedeutend mehr, dafür aber insgesamt nicht dringenden Mangelbehebungen, geliefert werden.

Wie genau diese Pakete geschnürt werden und welche Probleme nun wichtig und/oder dringend sind bzw. wo hier eine Grenze gezogen wird, würde den Rahmen dieser Arbeit sprengen und liegt zu einem großen Teil auch an der Philosophie des einzelnen Unternehmens.

Alleine die Tatsache, dass ein Mangel nicht mit der Behebung durch den Programmierer erledigt ist, sondern dass erst noch die Lieferung an den Kunden erfolgen muss und diese nicht sofort, sondern in gewissen Abständen und in großen oder kleinen Paketen, von statten geht, zeigt den Mehraufwand bzw. die größere Verantwortung an Problemmanagement Tools, die auch diesen Unterschied abdecken müssen. Des Weiteren ist dieser Mehraufwand auch eine zusätzliche mögliche Fehlerquelle. Mängel können tatsächlich aus der Software ausgebaut worden sein, nur wurde beim Einspielen oder beim Liefern der Software ein Fehler gemacht. Auch diese Schwierigkeiten treten bei Software im Projekt Stadium nicht auf.

3.4 Externe User

In Bezug auf Tools zur Unterstützung der Arbeit ergeben sich natürlich auch nicht unerhebliche Unterschiede. Der wohl größte ist die Tatsache, dass externe User zum Teil in den Entwicklungsprozess mit einbezogen werden müssen. Auch die Bezieher der Software haben sehr oft teilweise eigene Testteams, denen es genauso möglich sein sollte Probleme an den Lieferanten melden zu können. Auch Fehler aus dem Produktivbetrieb müssen gemeldet werden können. Dies wäre natürlich auch durch telefonische Einmeldung in einem Callcenter möglich. Doch zum einen müsste man ein solches eigens einrichten, und auf der anderen Seite gibt es auch kaum einen Kunden, der akzeptieren würde, dass er keinen Überblick über von ihm gemeldete Probleme hat, die direkten Einfluss auf ein bei ihm im Einsatz befindliches Softwareprodukt haben.

Ein kleines Beispiel:

Der Kunde A hat das Softwareprodukt P im Einsatz. Ein Mitarbeiter von A findet im Programm einen sehr schwerwiegenden Mangel, der die Erfassung von Kundenstammdaten unmöglich macht. Er meldet den Fehler dem Softwareproduzenten. Vier Tage später will er wissen, ob das Problem anerkannt wurde und wenn ja, wann er mit der Behebung des Mangels rechnen kann.

Würden wir uns im Projektstadium befinden, so wäre dies kein Problem, da alle Mitarbeiter im gleichen Unternehmen sitzen und das Problem beispielsweise in eine Datenbank eintragen und dort jederzeit aktuelle Informationen abfragen könnten. In unserem Fall wäre natürlich, wie schon angeführt, auch eine Einmeldung per Telefon, Fax oder Mail möglich, nur ist dann die weitere Verfolgung ein wenig schwierig. Muss ein Vertreter des Kunden einmal pro Woche beim Produzenten anrufen, um den aktuellen Status zu erfahren, oder bekommt er einmal pro Woche ein Mail mit den aktuellen Informationen? Würde man so verfahren, so würde man jede Menge Zeit verschwenden.

Auf der anderen Seite ist es vielleicht auch im Interesse des Herstellers, dass der Kunde nicht alle Details zu einem Problem einsehen kann. Auch wäre es möglich, dass mehrere Kunden das Produkt im Einsatz haben und somit Probleme melden können. Liegt es im Interesse der anderen Kunden, dass der Kunde A auch deren Probleme abrufen kann? Ist es für das Softwarehaus vorteilhaft, dass der Kunde A sehen kann, dass Mängel des Kunden X schneller behoben werden als Mängel des eigenen Unternehmens? Natürlich nicht!

Das Beispiel zeigt also deutlich, dass der Zugang zu Informationen bezüglich gemeldeter Probleme auch externen Usern ermöglicht werden muss, und dieser klare Unterschiede zu Informationen für interne User aufzuweisen hat, wogegen diese Unterscheidung in Softwareprojekten nicht relevant ist.

4 Trouble Ticket Systeme

Nachdem nun die Begriffe PM und Softwareprodukt ausreichend erklärt sein sollten, werde ich mich in diesem Kapitel mit dem Management von Problemen oder viel besser mit dem Werkzeug zum Management von Problemen beschäftigen.

Das Benutzerservice eines Softwareproduzenten hat die Aufgabe, „die Verfügbarkeit und Zuverlässigkeit der Systeme und seiner Dienste durch schnelle Entdeckung und Beseitigung von Fehlern möglichst hoch zu halten“ [Krus2001, 3].

Um dieses Ziel zu erreichen, wird ein Werkzeug benötigt, das die Serviceeinrichtung dabei unterstützt, Probleme professionell zu behandeln und zu dokumentieren. Als Bezeichnung für ein solches Tool hat sich allgemein die Bezeichnung Trouble-Ticket-System (TTS) durchgesetzt. Hier wird ein Datensatz erstellt, der sämtliche Daten, die zu einer Störung gehören, zusammenfasst. Dieser Datensatz wird in der Regel als Trouble Ticket (TT) bezeichnet [Krus2001, 3].

Im Zusammenhang mit TTS möchte ich nun folgenden Fragen genauer nachgehen: Welche allgemeinen Aufgaben sollen diese Systeme erfüllen, welche Ausprägungen von TTS sind bekannt bzw. welche Grenzziehungen sind möglich? Beenden möchte ich dieses Kapitel mit einem sehr kurzen Überblick über am Markt erhältliche Produkte.

4.1 Funktionsblöcke eines Trouble-Ticket-Systems

Im Allgemeinen sieht man in der Aufgabe des Trouble-Ticket-Systems das Speichern und Verwalten von Störungsmeldungen, den so genannten Trouble Tickets, in einer Datenbank. Zu beachten ist, dass sämtliche Tätigkeiten, die man während der Störungsbearbeitung durchführt, im TT dokumentiert werden müssen. Gliedert man das System in Funktionsblöcke, so wird diese Aufgabe als Trouble Ticket Management (TTM) bezeichnet [Krus2001, 5].

Das TTM stellt also die Kernaufgabe des Systems dar. Um die Leistungsfähigkeit zu verbessern, haben sich jedoch eine Vielzahl von Erweiterungen ergeben. [Joh92] „Dabei dienen die Erweiterungen sowohl der Verbesserung des Trouble Ticket Managements als auch der Schaffung neuer Funktionsblöcke wie Statistiken und Reports, Zugriff auf Problemlösungswissen, Integration in das Systemmanagement und Korrelation von Trouble Tickets“ [Krus2001, 5].

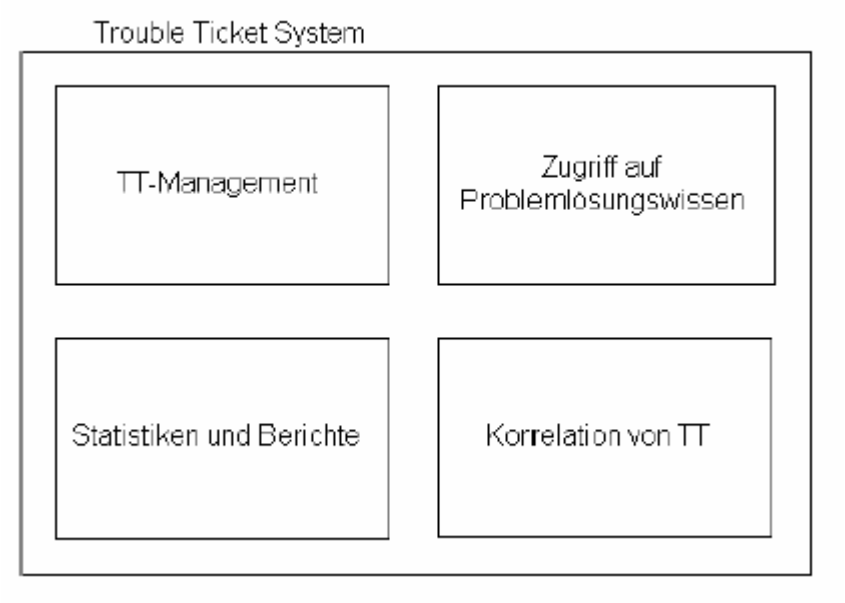


Abbildung 5: Die Funktionsblöcke eines TTS

Abbildung 5 zeigt die Funktionsblöcke des TTS, wobei die Unterscheidung zwischen Basis- und Erweiterungsfunktionen in einem TTS in der Literatur immer wieder anders aufgefasst wird. Durch den Lauf der Zeit haben sich Funktionen von dem Status einer Erweiterung zu einem nicht wegzudenkenden Teil des Funktionsumfangs entwickelt. Ich möchte nun die einzelnen Blöcke kurz erläutern, da sie in vielen Fällen die Grundlage für die Anforderungen im Kapitel 5 darstellen.

4.1.1 Trouble Ticket Management

Das Trouble Ticket Management unterstützt die Mitarbeiter in allen Phasen des Fehlermanagements. Es strukturiert die gesammelten Informationen und speichert sie so ab, dass die Daten für alle am Prozess Beteiligten verfügbar sind.

Den ersten Schritt im Trouble Ticket Management stellt das Erstellen eines TT dar. An dieser Stelle gibt es eine Vielzahl an Möglichkeiten, wie diese in das TTS gelangen können [Krus2001, 6]:

- Anruf eines Anwenders im Benutzerservice, wo ein Mitarbeiter die Daten entgegennimmt und in das TTS einträgt
- Erfassung der Daten durch den Anwender selbst durch einen Zugang zu den Erfassungsformularen über TTS-Clients oder über das Internet
- Übernahme strukturierter E-Mails in das TTS
- Automatische Generierung von TT aus Systemmanagementwerkzeugen

Es liegt auf der Hand, dass die Exaktheit und Vollständigkeit der zu einem TT verfügbaren Informationen von hoher Bedeutung ist. Um wiederholte Recherchen zu einem späteren Zeitpunkt zu vermeiden, sollen die bei der Entstehung eines TT ermittelten Daten und natürlich auch alle im weiteren Prozess gesammelten Daten allen Mitarbeitern, die an diesem Problem arbeiten, zur Verfügung stehen. Kruse weist weiter darauf hin, dass vollkommen egal wie ein TT in das System gelangt ist, eine bestimmte Grundmenge von Daten erfasst werden muss. Er nennt sie die so genannten Grunddaten eines TT [Krus2001, 6], die in vorgegebene Felder der TT-Datenbank abgelegt werden.

- Ticket-ID

  Dieses Attribut stellt die eindeutige Identifikation von Problemen sicher. In manchen Systemen wird auch ein zwei teiliger Schlüssel vergeben, der beispielsweise aus einem Kundenkürzel und einer fortlaufenden Nummer besteht. Es ist aber natürlich auch möglich, generell nur eine fortlaufende Nummer zu vergeben, um die Tickets schnell und eindeutig im System wieder zu finden.

- Ticket-Zustand

  Dies ist meist die Phase, in dem sich das Ticket befindet, z.B. in Diagnose, …

- Benutzerdaten

  Hier sind meist persönlichen Daten des Störungsmelders zu finden.

- Komponenten-/Dienstdaten

  Darunter versteht man die installierten Komponenten oder Einzelheiten zu den betroffenen Geräten oder Diensten, unter denen das Problem aufgetreten ist.

- Zeitstempel

  Zu jeder Aktion wird ein Zeitstempeln mitgespeichert, um zeitliche Verläufe auswerten und Rekonstruktionen vornehmen zu können. Der erste Zeitstempel hält fest, wann der Fehler aufgetreten oder gemeldet worden ist.

- Fehlerdaten

  Ist die Beschreibung der Störung

- Organisationsdaten

  enthalten Angaben zur organisatorischen Abwicklung der Fehlerbearbeitung,

z.B. Erfasser, Bearbeiter des Tickets, aber auch die Priorität des Tickets fällt unter diesen Punkt

Nach der Erfassung des TT erfolgt auf Grund der beschriebenen Störungssymptome die Klassifizierung. Klassifizieren heißt, Einordnen in einen Teilbereich von Problemen. Zumeist wird eine Menge an Begriffen definiert, die häufige Fehlersituationen darstellen, um möglichst einheitliche Einordnungen zu erlangen. Dennoch ist es wichtig, dass neben den durch die Begriffe fix vorgegebenen Auswahlmöglichkeiten auch noch Raum für freien Text zur Verfügung gestellt wird [Krus2001, 7].

Scheitert an dieser Stelle der Versuch, das Problem direkt zu lösen, so wird das TT an eine Ebene weitergeleitet, die eine höhere Qualifikation und Spezialisierung aufweist. Diese Ebene kann in der Regel mit detailliertem Fachwissen von bestimmten Teilbereichen dienen, während die Mitarbeiter der Störungsannahme als „Generalisten“ alle Aspekte des Produktes, jedoch auf relativ niedrigem Niveau, kennen. Die Weiterleitung erfolgt durch das System, wobei die Benachrichtigung darüber sowohl per Mail als auch durch andere so genannte Notifizierungswerkzeuge erfolgen kann [Krus2001, 8].

Um die Kosten für die Lösung von Problemen zu senken und die Kundenzufriedenheit zu steigern, ist es das vorrangige Ziel, eine möglichst hohe Sofortlösungsquote zu erreichen, da die Weiterleitung von Problemen Zeit und Geld sowohl für den Anwender als auch den Produzenten kostet.

4.1.2 Statistik und Reports

Eine durchaus nicht zu vernachlässigende Rolle spielen Auswertungen von Informationen aus einer TT-Datenbank. Diese können sich sowohl auf aktuelle Zustandsabfragen konzentrieren, als auch langfristige Trendanalysen abdecken. Die erstellten Statistiken und Reports dienen [Krus2001, 8]:

- der Unterstützung des Betriebs des TTS.
- der Verbesserung des Fehlermanagements.
- dem Management im Unternehmen zur Versorgung durch Daten.

Auswertungen sollen vor allem helfen, den aktuellen Betriebszustand des verteilten Systems abzufragen. Ein gutes Beispiel für diesen Fall stellen Mitarbeiter im

Benutzerservice dar, die regelmäßig die Anzahl der offenen TT für die Einteilung der täglichen Arbeit heranziehen [Trum2004, 7].

Weitere Daten, die ein gutes Report System liefern sollte, sind Auswertungen

- um das Fehlermanagement im Unternehmen zu verbessern.
- um einen guten Überblick über den Zustand des vernetzten Systems und des Fehlermanagements zu erlangen (z.B. Zeit, die ein System fehlerfrei läuft).
- die Hinweise auf latente Problemquellen geben.
- die zur Unterstützung des Managements herangezogen werden. (Z.B. Überblick über den Aufwand, der im Fehlermanagement betrieben wird) [Krus2001, 8]

4.1.3 Zugriff auf Problemlösungswissen

Durch das Lösen von Problemen und dem genauen Dokumentieren der gesetzten Aktionen entsteht ein Vorrat an Problemlösungswissen, der für alle Mitarbeiter zugänglich ist. Bei neu angelegten Problemen hat man somit die Möglichkeit auf alte Lösungsansätze, die ja in der TT-Datenbank gespeichert sind, zurückzugreifen. Dies eröffnet auch weniger qualifizierten Personen die Möglichkeit, Fehler zu diagnostizieren und zu beheben [Trum2004, 7f].

Eine weitere Option den Mitarbeitern das Fehlermanagement zu erleichtern ist die Einbindung von externen Wissensdatenbanken in den Diagnoseprozess. Hier muss jedoch beachtet werden, dass unternehmensspezifische Abläufe, Konfigurationen und Anwendungen in der Regel nicht berücksichtigt sein können [Krus2001, 10].

Auch das Installieren von intelligenten Assistenten, die auf Daten der TT-Datenbank zurückgreifen, ist möglich. Diese führen die Bediener bei der Fehlerdiagnose durch zum Beispiel ein spezielles Abfrage-Schema, um unerfahrenen Mitarbeitern oder Anwendern zur Seite zu stehen [Krus2001, 10].

4.1.4 Integration in das Systemmanagement

Eine Forderung, die der genaueren Beschreibung von Problemen dienen soll, ist die Einbindung in ein Systemmanagement. Dieses überwacht und steuert das gesamte verteilte System. Die Einbindung soll nun einen direkten Zugriff auf die Netzwerkmanagementwerkzeuge ermöglichen um erfasste Gerätedaten an das TTS weitergeben zu können. Managementwerkzeuge sollen sofort mit der gewünschten

Maske erscheinen, um die benötigten Daten anzuzeigen oder andere gewünschte Aktionen sollen in kürzester Zeit vom TTS aus ausführbar sein.

Eine weitere Forderung in diesem Zusammenhang ist die automatische Erstellung von TT durch vom Systemmanagement direkt erkannte Störungen. Ebenfalls von großer Bedeutung ist der Rückgriff des Systemmanagements und des TTS auf die gleiche Datenbasis, um Redundanzen zu vermeiden und die Pflege der Daten zu vereinheitlichen [Krus2001, 10].

4.1.5 Korrelation von Trouble Tickets

Durch den Umstand, dass Probleme von vielen verschiedenen Stellen gemeldet werden können, wie zum Beispiel vom Kunden A, B oder C oder einem automatisierten Test beim Softwareproduzenten oder einem internen Tester, und die Beschreibung von jeder Stelle zwar anders lauten kann aber dennoch das gleiche Problem beschrieben wird, kann es vorkommen, dass ein und das selbe Problem mehreren Tickets zu Grunde liegt.

„Ziel der Korrelation ist es nun, zu erkennen, dass die ursächlich zusammengehörenden TT auch als Folgestörungen bearbeitet werden können und dadurch eine Mehrfachbearbeitung zu vermeiden. Bei Behebung der Störungsursache sollten dann auch alle mit dem bearbeiteten TT zusammenhängenden „Folgetickets“ automatisch auf gelöst gesetzt werden“ [Krus2001, 10].

4.2 Abgrenzungsversuch

In der Literatur sind jede Menge von verschiedenen Bezeichnungen für Trouble Ticket Systeme zu finden. Zumeist werden Schlagworte, wie zum Beispiel Trouble, Problem, Bug, Request, Feature, Ticket, Tracking und System kombiniert, um auf Besonderheiten der einzelnen Tools hinzuweisen. Die häufigsten Bezeichnungen hat Trummer in seiner Diplomarbeit [Trum2004, 8] aufgelistet:

- Problem Tracking Systeme (PTS)
- Support Tracking System (STS)
- Bug Tracking Systeme (BTS)
- Feature Request Systeme (FRS)
- Knowledge Data Base Systeme (KDB)

Trummer versucht eine Abgrenzung zwischen den angeführten Systemen vorzunehmen, indem er STS als Austauschwerkzeug zwischen Supportmitarbeitern und Kunden einordnet. Ist der Support nicht in der Lage das Problem zu lösen, wird das PTS herangezogen und wird dieses Problem tatsächlich als Fehler in der Software diagnostiziert, so wird das BTS verwendet. Für die Verwaltung neuer Ideen zur Verbesserung der Software kommt das FRS zum Einsatz, und Tipps und Tricks stellt eine Knowledge Data Base zur Verfügung. Er räumt aber ein, dass diese Systeme oft Hand in Hand arbeiten und es nicht selten vorkommt, dass ein System mehrere Aufgaben übernimmt. Als Gemeinsamkeit sieht er das Arbeiten mit Tickets, die während ihres Lebens gemachte Veränderungen speichern. „Support, Problem, Bugtracking und Feature Request Systeme werden gerne, sowohl thematisch als auch in der Praxis, zusammengefasst“ [Trum2004, 8].

Ich werde mich in dieser Arbeit, wie schon im Kapitel 2 erwähnt, an der Abgrenzung nach ITIL orientieren und über Problemmanagementsysteme schreiben. Diese nehmen meiner Ansicht nach sämtliche Aufgaben, die Trummer als Unterscheidungskriterium anführt, war und stoßen erst an die Grenze des Change-Managements und nicht schon an Grenzen wie Bugs oder Features. Bugs und Features sind meiner Meinung nach ebenso Problems und aus diesem Grund auch vom Problemmanagement erfasst. Diese Meinung teilen auch [HaTS2004].

Trotzdem sind diese Unterscheidungsmerkmale von Bedeutung. Ich werde im nächsten Kapitel beispielsweise sehr ausführlich auf die Wichtigkeit der Unterscheidung von Mangel und Erweiterungsauftrag eingehen. Nicht zielführend und zum Teil in der Praxis auch kontraproduktiv ist aber meiner Ansicht nach die Trennung der Aufgaben des Problemmanagement in einzelne Systeme.

Trummer versucht noch eine zweite Abgrenzungsmöglichkeit aufzuzeigen, indem er TTS mit anderen Anwendungen in Relation setzt. Er zeigt auf, dass TTS als Teil anderer Anwendungen angesehen oder ausgebaut werden können und dass ein Trend zur Eindringung von TTS – Herstellern in die eine oder andere Anwendung erkennbar ist. Zu folgenden Anwendungen können Relationen aufgezeigt werden [Trum2004, 8f]:

- Helpdesk Anwendungen
- Groupware Anwendungen
- Cased Based Reasoning Systeme

- Knowledge Management Systeme
- Customer Relationship Management

4.3 Übersicht verfügbarer Systeme

Sehr übersichtliche und brauchbare Informationen zu Trouble Ticket Systemen findet man im Internet unter [Eato2002], diese Seite stellt eine Zusammenfassung bzw. eine Art FAQs dar, die durch die Newsgroup comp.software.config-mgm erstellt wurden. Diese Gruppe beschäftigt sich in erster Linie mit Configuration Management in der Softwareentwicklung, da aber Probleme eine sehr enge Verbindung zu diesem ITIL-Bereich aufweisen [Olbr2004], bietet die Newsgroup auch Informationen zum Problemmanagement.

Auf diesen Internetseiten findet man sowohl eine Auswahl an am Markt erhältlichen kostenlosen und kostenpflichtigen Systemen sowie Kontaktadressen zu den jeweiligen Herstellern. Im weiteren Teil dieses Kapitels möchte ich nun Untersuchungen von Trummer kurz zusammenfassen, der jeweils vier Produkte aus den beiden Sektoren auswählte, um sie näher zu beleuchten [Trum2004, 10ff].

4.3.1 Nichtkommerzielle Produkte

Die Internetseite [Eato2002] führt acht frei verfügbare TTS an. Diese Zahl zeigt schon, dass es nicht besonders viele Produkte gibt, die man kostenlos einsetzen kann. Trummer wählte in seiner Untersuchung die Produkte Bugzilla, Debian Bug Tracking System (debbugs), GNATS und Double Choco Latte aus, wobei zu beachten ist, dass die ersten drei Systeme auf den Einsatz im Bereich Bugtracking bei Opensource Projekten zugeschnitten sind, Double Choco Latte hingegen bietet zusätzlich noch Ansätze einer Groupware Software. Da Opensource Projekte, so gut wie keine Wartungsverträge oder anderen Support anbieten, um Störungen zu managen, beschränken sich auch deren TTS auf den Bereich Mangelmanagement [Trum2004, 10ff].

Bugzilla

Bugzilla ist auf der einen Seite sehr bekannt und wird auch in zahlreichen Firmen und Open Software Projekten eingesetzt. Es dient in erster Linie dem Bugtracking, also dem Verwalten von offenen Fehlern.

Die größten Stärken sind:

- Möglichkeit von Produkt- oder Projektorientierung
- Abhängigkeiten von Bugs können graphisch dargestellt werden
- fortgeschrittene Möglichkeiten zur Reporterstellung
- Umfangreiche Möglichkeiten zur Konfiguration des Programms
- Unterstützung von E-Mail Formaten und XML Formaten zur Fehlerberichterstattung
- Zugriffsmöglichkeiten über Konsole oder Webbrowser
- Anbindung zu Konfigurationswerkszeugen
- Robustheit und Popularität

Zu den größten Nachteilen zählen die Einschränkung auf nur eine bestimmte Datenbank (mySQL), dass für die Benachrichtigungen E-Mails verwendet werden, dass die Reporterstellung sich nur geringfügig konfigurieren lässt und die Abhängigkeit von nicht standardisierten Bibliotheken [Trum2004, 11f].

Debian Bug Tracking System (debbugs)

Debbugs besteht aus Perl Skripts, die Problemreporte in einer Datenbank verwalten, der übliche Begriff Ticket wurde durch den Begriff Report ersetzt. Vorraussetzung für dieses System sind ein UNIX Betriebssystem und eine eigene Maildomain. Das System wurde bisher für einige Projekte verwendet und basiert hauptsächlich auf E-Mails. Bestes Beispiel ist der Primärschlüssel eines Reports: Eine E-Mailadresse!

Als größte Vorteile werden genannt [Tru2004, 12]:

- weder Zugang zum Hostsystem noch Webzugang notwendig
- Mit E-Mails können einfach zusätzliche Informationen zu einem Report abgelegt werden
- große Leistungsfähigkeit (im Laufe von sechs Jahren 16000 Reporte verwaltet)
- Seit 1997 durch Parametrisierung anpassungsfähig

GNATS

Die Hauptaufgabe von GNATS ist die Verwaltung von Problemberichten, die über Kategorien zu Gruppen zusammengefasst werden und die Kommunikation über diese. Die Pluspunkte sind [Trum2004, 9]:

- Archivierter E-Mailverkehr und Möglichkeit der indizierten Suche darüber
- Zugriffsmöglichkeiten über Befehle für die Kommandozeile, E-Mail, einem Webinterface, …
- einfache Benutzbarkeit und hohe Flexibilität
- Möglichkeiten Felder anpassen zu können, verschiedene Datenbanken zu verwenden und umfassende Web- und Tcl-Benutzerschnittstellen

Double Choco Latte

Hier besteht das Kernkonzept aus den so genannten "Work Orders". Diese sind für die zeitliche Verfolgung von Fehlern, Anfragen, Wartungsaufgaben und Projektaufgaben verantwortlich.

Die herausgearbeiteten Eigenschaften für dieses Produkt bestehen aus folgenden Punkten [Trum2004, 13]:

- mehrere Projekte (+ Unterprojekte) können verwendet werden
- "Call Tickets" um den Kontakt mit dem Kunden zu repräsentieren
- eigene Attribute können definiert werden
- hierarchische Verwaltung von Projektmitarbeitern ist möglich
- besondere Möglichkeit mit E-Mail Benachrichtigungen zu arbeiten
- auch statische Berichte können erstellt werden

4.3.2 Kommerzielle Produkte

Zu den Systemen, die nicht gratis einsetzbar sind, ist einerseits zu sagen, dass es bedeutend mehr Produkte gibt und andererseits, dass die Systeme bedeutend komplexer und umfangreicher sind.

Aus meiner Sicht muss aber stark in Zweifel gezogen werden, dass der Anschaffungswert eines solchen Produktes tatsächlich nur den Kaufpreis eines solchen Systems umfasst. In beinahe allen Fällen wird die Vielseitigkeit und Universalität damit bezahlt, dass mit einem hohen Berateraufwand während der Installation des Systems zu rechnen ist. Manche Hersteller rühmen sich damit, dass eine Anpassung in wenigen Wochen möglich ist, was auf teilweise sehr hohe Aufwendungen bei Konkurrenten schließen lässt.

Beinahe auch für alle Produkte gilt, dass sie zumeist durch Geschwister-Systeme vom gleichen Hersteller erweitert werden können und dadurch die Beurteilung des einzelnen Systems nur sehr schwer möglich ist. Eine Liste von kostenpflichtigen Produkten ist ebenfalls unter der Internetadresse [Eato2002] abrufbar. Trummer stellt in seiner Arbeit die Produkte ClearQuest und ClearDTTS von Rational und 2 umfangreiche Systeme zur Kundenbetreuung Aegis Defect Tracking und OmniTracker vor [Trum2004, 13ff].

Rational ClearQuest

Dieses Produkt soll Softwareproduzenten beim Entwicklungsprozess unterstützen, seine größten Vorteile bestehen in [Trum2004, 14]:

- Der Anpassbarkeit von Benutzeroberflächen, Eintragsfelder und dem Workflow
- dem Vorhandensein von Replizierungs- und Synchronisationsmechanismen
- Schnittstellen zu integrierten Entwicklungsumgebungen wie z.B. Visual Studio

Rational ClearDDTS

Im Gegensatz zu ClearQuest beschränkt sich das vom gleichen Hersteller angebotene ClearDDTS auf die Kommunikation zwischen Testern und Entwicklern im UNIX Umfeld.

Die größten Stärken bestehen in den 40 verschiedene Managementberichten, die dieses Produkt liefern kann. Dem gegenüber werden als Datenbank nur Oracle und eine eigene SQL-fähige Datenbank unterstützt [Trum2004, 14f].

Aegis Defect Tracking

Dieses Produkt ist Teil einer großen Produktfamilie und deshalb schwer bewertbar. Aus diesem Grund werde ich mich auf die von Trummer angeführten Stärken beschränken [Trum2004, 15]:

- vollständig anpassbare und personalisierbare automatische Benachrichtigung durch E-Mails
- automatischer Aufbau einer durchsuchbaren Wissensbasis
- vollständige Kundenverwaltung ist integriert
- Berichtgenerator um zusammenfassende Berichte oder graphische Darstellungen zu erstellen
- mächtige Möglichkeiten zur Filterung
- vollständige Adaptierbarkeit für z.B. Formulare oder Eintragsfelder

OmniTracker

OmniTracker ist ein Produkt, das schon alleine durch einen sehr großen Funktionsumfang geprägt ist und dennoch laut Hersteller nicht nur dynamisch einsetzbar ist, sondern auch noch einfach zu konfigurieren.

Die herausragendsten der zahlreichen Funktionalitäten sind folgende [Tru2004, 16f]:

- Einbringung neuer Tickets durch Telefon, E-Mail, Telefax oder Webzugang möglich
- einfach zu konfigurierendes System
- Anpassung an einen Kunden in 5-8 Tagen

- Konfiguration erfolgt mittels Dialogen
- Workflow, Datenmodell, Benachrichtigung, Eskalationsmodell, Benutzerrechte und Statistiken sind konfigurierbar
- für den Bereiche ITIL gibt es vordefinierte Datenmodelle
- Benachrichtigungen erfolgen nach dynamischen Regeln
- dynamische Eskalationssteuerung mittels Regeleditor
- eingebaute Benutzerverwaltung, Formulareditor, integrierte Volltextsuchmaschine und mehrere Arten von Statistiken

4.4 Zusammenfassung

Trouble Ticket Systeme dienen zur Verfolgung von Fehlern oder Problemen. Sie sind durch das zentrale Element des Trouble Tickets gekennzeichnet, wovon jedes genau ein Problem beschreibt. Die Aufgabe eines Trouble Tickets besteht in der Dokumentation des Lebenszyklus eines Problems.

„Kernaufgabe der Trouble Ticket Systeme ist das Fehlermanagement, bestehend aus Fehlererhebung, Fehlerdiagnose und Fehlerbehebung. Weitere Aufgaben sind die Erstellung von Statistiken und Berichten und das Auffinden von korrelierenden Trouble Tickets“ [Trum2004, 17].

Trouble Ticket Systeme sind am Markt unter vielen verschiedenen Bezeichnungen zu finden und konzentrieren sich auf verschiedenste Einsatzgebiete, die aber alle in enger Relation zu einander stehen. Adaptierbare Systeme decken zumeist mehrere Gebiete ab. Alle vorgestellten Systeme entstammen dem Fehlermanagement für Software, wobei zu erkennen war, dass die nicht kommerziellen Systeme eigentlich durch die Bank nur den Fehlermanagementbereich für Softwareentwicklung abdecken, wohingegen die kommerziellen Systemen in der Lage waren, durch starke Erweiterung der Funktionalität auch andere Anwendungsfelder zu erschließen.

Es sind deutlich mehr kommerzielle als nichtkommerzielle Systeme auf dem Markt erhältlich, und die Bandbreite der abgedeckten Funktionalität ist ebenso deutlich höher [Trum2004, 17].

5 Anforderungen an den Workflow eines Trouble-Tickets

Nachdem nun das Prinzip von TTS vorgestellt und deren Leistungsumfang beleuchtet wurde, widme ich mich nun der Frage, welche konkreten Anforderungen ein Softwareproduzent an ein TTS stellt.

In der Literatur sind zahllose verschiedenste Anforderungen zu finden. Sie reichen von Automatischen Benachrichtigungen bis zu vordefinierten Standardreports [DSDP2001, 1] oder von SMS-Benachrichtigungen bis zu verschiedensten Verknüpfungsmöglichkeiten mehrerer Dokumente [HaTS2004, 197].

Alle Anforderungen aufzugreifen oder in dieser Arbeit zu behandeln, ist unmöglich und würde den Rahmen dieser Arbeit bei weitem sprengen. Aus diesem Grund möchte ich einen Teilbereich aus den Anforderungskatalogen herauslösen, für den durchgängig vollkommene dynamische Adaptierbarkeit gefordert wird. Dieser Bereich ist der Lebenszyklus oder das Phasenmodell eines Trouble-Tickets.

Wie im letzten Kapitel beschrieben, ermöglichen die meisten TTS einen dynamisch konfigurierbaren Workflow, doch wie soll man diese Möglichkeit der Anpassung ausnützen ohne die Anforderungen an einen tatsächlichen Lebenszyklus zu kennen? Alleine die Anpassungsmöglichkeit reicht nicht aus, um ein TTS bei einem Softwareproduzenten einzuführen.

Das Herzstück eines jeden TTS ist das tatsächlich hinterlegte Phasenmodell, das den Prozess der Abarbeitung von Trouble-Tickets am wirkungsvollsten bestimmt und damit den größten Einflussfaktor auf die erfolgreiche Bewältigung von Problemen darstellt. „Ein gutes Verfahren hat ein Phasenmodell, das die einzelnen Arbeitsschritte widerspiegelt. Damit ist auch immer eindeutig feststellbar, welche Mangelbehebungen wie weit gediehen sind“ [HaTS2004, 164]. Dieses Kapitel soll nun die detaillierten Anforderungen an einen solchen Zyklus zusammenstellen, um im nächsten Kapitel ein Modell zu entwickeln, dass diese Anforderungen abgedeckt.

5.1 Eingrenzung des TTS-Nutzers

Bevor ich die Anforderungen und deren Hintergründe zusammenstelle, muss noch der Adressatenkreis, für den das TTS gedacht ist, eingeschränkt werden. Der für die Definition der Anforderungen herangezogen Softwareproduzent stellt zwar Standardsoftware (zur Abgrenzung zur Individualsoftware siehe [Bla2000]) her, er muss jedoch nicht einen unendlich großen Kundenkreis versorgen, und er schließt mit jedem seiner Kunden Verträge über die Nutzung, Weiterentwicklung und Wartung ab.

Diese Voraussetzungen betreffen vor allem kleine bis mittel große Softwareproduzenten, die eine höhere Abhängigkeit von ihren Kunden aufweisen als dies für die *Global Player*, wie zum Beispiel Microsoft einer ist, gilt. Problemmanagement bedeutet für diese eher kleinen Hersteller meist nicht nur die Kundenzufriedenheit und damit das Vertrauen in das Produkt zu stärken, sondern stellt beinahe immer eine vertragliche Pflicht dar.

5.2 Abstimmung mit Kunden

Dieses Kapitel beschäftigt sich mit Anforderungen an ein TTS, die vor einer Umsetzungsentscheidung des gemeldeten Tickets entstehen. Ich werde an dieser Stelle auf folgende Probleme eingehen:

- Eine konzentrierte Freigabe beim Kunden, um fehlerhafte Problemmeldungen zu vermeiden ist erforderlich.
- Die Erzielung einer gemeinsamen Sicht auf das Problem durch den Kunden wie auch durch den Lieferantenvertreter ist notwendig.
- Das Zurückziehen von Problemmeldungen muss möglich sein.

Diese Probleme haben in sehr vielen Fällen die verschiedensten vertraglichen Hintergründe. Ich werde die jeweils zum Verständnis interessanten Punkte in den entsprechenden Kapiteln anführen, um den Zusammenhang zu gewährleisten.

5.2.1 Konzentrierte Freigabe

Bereits zum Zeitpunkt der Problemmeldung, also noch bevor das Softwarehaus von dem Problem Kenntnis hat, entstehen die ersten Anforderungen an den Lebenszyklus. Softwareproduzenten unternehmen immer wieder den Versuch, sich vor einer Überflutung durch vermeidbare Problemmeldungen zu schützen. Für den Kunden ist die Versuchung einfach zu groß, jedes Problem einmal auf Verdacht hin zu melden. Unter diesen Meldungen befinden sich dann zweifellos auch wirkliche Probleme, aber in der Regel werden sie mit Problemmeldungen vermischt, die durch

- Fachwissen entsprechender Stellen beim Kunden,
- durchlesen der FAQs,
- genauere und sorgfältigere Tests oder
- bessere Kenntnis der Verträge oder des Pflichtenheftes

verhindert werden hätten können. Dieser sorglose Umgang sorgt nicht nur für Ärger bei den jeweiligen Beuteilungsinstanzen, er verlangsamt auch noch die Bearbeitung von tatsächlichen Problemen. Um diesem Problem entgegenzuwirken, führt man eine Art von Budget für Falschmeldungen ein. Das heißt, jeder Kunde hat zum Beispiel zwanzig unberechtigte Problemmeldungen pro Monat frei, und für jede weitere muss ein Betrag X bezahlt werden. Solche Regelungen machen die Abstimmung von Problemen natürlich nicht leichter, sie führen aber zu einer Forderung des Kunden, nach einer Vorbeurteilung von neu erstellten TT durch eine entsprechend geschulte Instanz des Kunden.

Der Kunde unseres Softwareproduzenten besteht also auf eine Möglichkeit, Tickets durch seine Testabteilung anlegen lassen zu können, ohne dass diese bereits beim Hersteller einlangen. Vielmehr muss das Ticket zuerst an eine weitere Stelle beim Kunden weitergeleitet werden, damit diese eine Art Vorprüfung durchführen kann, um das Budget für unberechtigte Problemmeldungen nicht allzu schnell aufzubrauchen. Diese Forderung liegt also nicht nur im Interesse der Kunden, sondern hilft auch die Anzahl der Fehlmeldungen und damit die Unzufriedenheit bei der Problemabstimmung als auch die unproduktive Zeit zu reduzieren.

5.2.2 Erzielung einer gemeinsamen Sicht auf das Problem

Eine Problemabstimmung ist vom Prinzip her nichts anderes als eine Verhandlung zwischen einem Benutzer der Software und einem Mitarbeiter des Softwarehauses zu vorgegebenen Punkten. Als Einführung in dieses Kapitel möchte ich ein Beispiel bringen, dass die am häufigsten auftretenden Fälle aufzeigt:

Das Reisebüro Supergünstig betreibt sämtliche Rechner mit dem Softwareprodukt Angebotskalkulierer. Seit zwei Tagen ist eine neue Version im Einsatz und beim Erstellen eines Angebots für einen Kunden fällt dem Mitarbeiter Maier auf, dass plötzlich eine Bearbeitungsgebühr von 20% verrechnet wird, obwohl bisher nur 5 % verrechnet wurde. Er ist aufgrund dieser falschen Berechnung extrem verärgert und meldet ein TT beim Hersteller von Angebotskalkulierer. Er fordert baldige Behebung mit höchster Priorität.

Welche Möglichkeiten sind nun in dieser Situation denkbar?

Möglichkeit 1:

Der Mitarbeiter hat vollkommen Recht. Bei der letzten Version ist dem Programmierer ein Fehler unterlaufen und er hat den Zinssatz versehentlich verändert. Eine Änderung im Code des Programms ist notwendig und muss sobald wie möglich durchgeführt werden. Das gemeldete Ticket muss also vollkommen akzeptiert werden, sowohl was die Priorität als auch die baldige Behebung betrifft.

Möglichkeit 2:

Der Mitarbeiter hat zum Teil Recht. Durch das Umstellen auf die neue Version haben sich die Steuerungsattribute in der Datenbank verstellt. Der Fehler kann kurzfristig durch ein SQL Statement auf die Datenbank bereinigt werden und muss bis zur Lieferung einer nächsten Version behoben werden, um ein erneutes Zurückstellen zu verhindern. In diesem Fall wird das gemeldete Ticket wohl wieder akzeptiert, jedoch kann sowohl die Priorität als auch die Behebungsgeschwindigkeit zu Gunsten des Produzenten verändert werden.

Möglichkeit 3:

Der Mitarbeiter liegt vollkommen daneben. Im Pflichtenheft für die neue Version wurden vorschnell 20% vereinbart, da man vor hatte den Prozentsatz zu erhöhen. Nun hat sich aber herausgestellt, dass diese Erhöhung nicht durchsetzbar ist. Das Reisebüro möchte den Satz gerne wieder auf 5% zurücksetzen. Das Ticket wurde hier auf jeden Fall falsch klassifiziert, dennoch ist eine Umsetzung denkbar. Klar ist aber, dass es sich um keinen Mangel sondern vielmehr einen Erweiterungswunsch oder in diesem Fall noch genauer einem Änderungswunsch handelt und dieser ev. kostenpflichtig ist.

Möglichkeit 4:

Der Mitarbeiter liegt noch weiter daneben. Das Reisebüro hat den Prozentsatz tatsächlich verändert ohne es dem Mitarbeiter mitzuteilen. Im Pflichtenheft für die neue Version wurden deshalb 20% vereinbart. Das gemeldete Ticket ist also gegenstandslos und hätte gar nicht gemeldet werden dürfen.

Möglichkeit 5:

Die Einstellung des Bearbeitungszuschlags obliegt dem Reisebüro alleine. Der Verantwortliche kann den Satz beliebig ändern und der Softwarehersteller trägt gar keine Verantwortung für diese Einstellung. In diesem Fall liegt wohl wieder kein Problem der Software vor.

Diese 5 verschiedenen Möglichkeiten wären beliebig fortsetzbar und beinhalten nur klare Situationen. Natürlich sind auch Unstimmigkeiten über Tatbestände in der Praxis an der Tagesordnung.

Der Melder des Problems übermittelt, wie uns auch das Beispiel zeigt, nicht nur das Problem selbst, er legt auch die erste Verhandlungsgrundlage auf den Tisch. Dies hat er in unserem Beispiel in Form von Priorität und erwarteter Liefergeschwindigkeit getan. Ziel dieser Verhandlung ist zu aller erst eine Prüfung, ob die Einstufung des Problems durch die meldende Organisation generell den Tatsachen entspricht und Vertragskonform ist.

Zusammengefasst müssen also folgende Punkte geklärt werden:

- Handelt es sich um ein berechtigtes Problem?
- Handelt es sich um einen Mangel oder Erweiterungsauftrag?
- Ist die Umsetzung kostenpflichtig?
- Zu welchem Zeitpunkt wird die Problemlösung umgesetzt sein?
- Wie schwerwiegend ist das Problem?

Unser Lebenszyklus muss also folgende Möglichkeiten schaffen:

- Die Sichtweise beider Parteien muss aufgenommen werden können.
- Es muss vermerkt werden können, worüber bereits Einigkeit besteht und welche Punkte noch verhandelt werden müssen.
- Des Weiteren muss auch erkennbar sein, ob und mit welchem Ergebnis die Beurteilung abgeschlossen ist.

5.2.3 Rückzug eines Tickets

Am Ende dieses Kapitels möchte ich noch eine Anforderung anführen, die selbstverständlich ist, jedoch vielfach nicht beschrieben wird.

Enden die Abstimmungsverhandlungen mit der Vereinbarung, dass das Problem nicht behoben werden soll, so kann dies zwei Gründe haben:

- Die Problemmeldung war nicht berechtigt.
- Das Problem ist berechtigt, wird aber aus einem bestimmten Grund nicht behoben.

Der erste Fall liegt auf der Hand. Nehmen wir als Beispiel die Möglichkeit 5 im letzten Kapitel. Der Kunde hat ein Problem gemeldet und es hat sich in den Verhandlungen herausgestellt, dass es sich um gar kein Problem handelt. In diesem Fall muss der Kunde die Möglichkeit haben, das Ticket zurückzuziehen.

Der zweite Fall wird oft vergessen. Meldet zum Beispiel ein Kunde ein Problem, dass in der Zukunft vielleicht nicht wieder auftreten kann, oder ist die Auswirkung so gering, dass der Kunde mit diesem Problem leben kann, die Behebung jedoch viel Geld kosten würde, so muss der TT-Worflow eine Möglichkeit vorsehen, dass die Meldung des Problems zwar berechtigt war, von einer Umsetzung jedoch abgesehen wird.

Die Unterscheidung der beiden Fälle ist dann besonders wichtig, wenn wie im Kapitel 5.2.1 beschrieben, ein Budget für Falschmeldungen vereinbart wurde.

5.3 Unterscheidung zwischen Mangel und Erweiterungsauftrag (CR)

Im letzten Kapitel wurde davon gesprochen, dass eine Übereinkunft darüber erzielt werden muss, ob ein Problem als Mangel oder Erweiterungsantrag angesehen wird. Dies ist auch der Grund, warum ich an dieser Stelle ganz besonders auf zwei zentrale Fragen eingehen werde, die in diesem Zusammenhang als besonders wichtig für Softwareproduzenten anzusehen sind

- Wo ist die Grenze zwischen Mangel und Erweiterungsauftrag zu ziehen?
- Warum ist die Grenze zwischen Mangel und Erweiterungsauftrag so wichtig?

An dieser Stelle möchte ich auch besonders darauf hinweisen, dass die Verwechslungsgefahr zwischen den Begriffen Change-Request und Request for Change sehr groß ist.

Während ein CR eine spezifische Art des Problems darstellt, kann man RfCs oder Änderungsanträge als die Schnittstelle zum CM ansehen. Jedes Problem kann einen oder mehrere Änderungsanträge an das CM auslösen. Ein RfC ist also ein Arbeitsauftrag an die Mitarbeiter des Change-Managements, der zur Umsetzung eines Mangels oder CRs notwendig ist.

Weiters möchte ich darauf hinweisen, dass die Grenze zwischen CM und PM in diesem Abschnitt besonders leicht überschritten werden kann und auch in der Literatur nicht eindeutig gezogen wird. Einerseits gibt es Verfechter, die Erweiterungsaufträge beinahe vollständig aus dem PM ausgrenzen [Olb2003, 39] und auf der anderen Seite vertreten auch einige Autoren die Meinung, dass ein Erweiterungsauftrag ebenfalls nur ein besonderes Problem darstellt, das genauso Vorschläge an das CM macht und diesem mittels Requests for Change diese übermittelt und am Ende die Überprüfung der erfolgreichen Umsetzung durch das PM erfolgt [HaTS2004, 166].

5.3.1 Grenze zwischen Mangel und Antrag

„Kann das Produkt etwas nicht, was in der Anforderungsspezifikation versprochen wurde, dann haben wir es mit einem Mangel zu tun. Ist der Effekt, den der Anwender fordert, gar nicht durch das Fachkonzept abgedeckt, dann liegt ein Erweiterungsantrag vor“ [HaTS2004, 166].

Diese Definition klingt einfach, ist aber in der Praxis teilweise oft sehr schwierig, denn auch das Fachkonzept selbst kann mangelhaft sein. In der Regel kann man also in einem ersten Schritt davon ausgehen, dass, wenn das Verhalten der Software von jener des Konzepts abweicht, ein Mangel vorliegt. Entspricht das Verhalten aber dem Konzept, so gibt es vor allem zwei weitere Gründe, die für die Vorlage eines Mangels sprechen würden [HaTS2004, 166f]:

- **Widersprüchlichkeit**

  Es ist beinahe unmöglich ein Fachkonzept, das oft mehrere hundert Seiten umfasst, ohne Widersprüche abzufassen. Dies gilt vor allem für den Fall, dass mehrere Personen dieses Konzept verfasst haben. „Häufig bestehen die Widersprüche zwischen allgemein gültigen und Ausnahmen: An einer Stelle fordert der Anwender, dass alle Konten fett gedruckt werden, Monate später will er dann doch, dass Girokonten anders dargestellt werden. Das Ergebnis wird bei der Spezifikation des Girokontos festgehalten, auf eine Anpassung der allgemeinen Passage wird aber allzu leicht vergessen. Ist ein fett gedrucktes Girokonto nun Symptom eines Mangels?“ [HaTS2004, 166f]

- **Unvollständigkeit**

  Ein Fall, der eigentlich nicht zu verhindern ist, sind die berühmten "weißen Flecken auf der Landkarte". Das System verhält sich weder im Widerspruch noch konform zur Spezifikation, da die betreffende Situation schlicht und einfach nicht geregelt ist. Vor allem Anforderungen, die nicht auf die Funktionen eines Produktes abzielen, werden am Beginn der Entwicklung gerne vernachlässigt. Niemand weiß zum Beispiel, welche Versicherungsprodukte in den nächsten Jahren den Markt dominieren werden. Die Gefahr ist groß, diese Anforderungen erst relativ knapp vor der Produktionsaufnahme zu klären. Ein weiteres Beispiel sind Anforderungen an das Datenvolumen. Meist ist nicht klar, wie viele Datensätze ein System verwalten muss. Probleme treten hier erst nach einiger Zeit auf [HaTS2004, 167].

5.3.2 Bedeutung der Grenze zwischen Mangel und Antrag

Wie sich unschwer erkennen lässt, hat die Grenze für den Lieferanten eines Softwareprodukts zumindest rein wirtschaftlich eine unglaublich große Bedeutung. Jeder Kunde kann davon ausgehen, dass Mängel ausnahmslos durch einen Wartungsvertrag abgedeckt sind und damit keine Kosten für die Behebung auf Kundenseite anfallen. Im Umkehrschluss dazu trägt also der Lieferant den gesamten Anteil der Kosten.

Ein Erweiterungsauftrag, der ja eben nicht durch eine bereits vereinbarte Spezifikation abgedeckt ist, muss meist zu einem großen Teil vom Kunden bezahlt werden, wobei an dieser Stelle anzumerken ist, dass es im Gegensatz zum Mangel, wo der Hersteller ausnahmslos die Kosten trägt, beim CR sehr wohl in vielen Fällen zu einer Kostenteilung kommen kann oder gar auf dem Wege der Kulanz zu einer Kostenübernahme durch den Produzenten kommt.

Es gibt aber noch andere als rein wirtschaftliche Gründe, die die Bedeutung der Grenzziehung so wichtig machen:

- Wenn nicht klar ist, wo die Grenze zwischen CR und Mangel verläuft, dann ist unklar, wie die **Qualität des Produktes** zu bewerten ist. Sehr oft wird von Managementseite gefragt, wie viele Mängel in einem gewissen Zeitraum eingemeldet wurden, oder ob insgesamt ein Rückgang an Mängeln zu beobachten ist. Es muss also ein Hauptziel des für das Produkt Verantwortlichen sein, die Zahl der Anwendermangelmeldungen auf einem niedrigen Niveau zu halten, um festzustellen, ob das Produkt auf einem guten Weg ist. Dies ist aber ohne klare Mangeldefinition nur sehr schwer bis gar nicht möglich [HaTS2004, 164].
- Der oben genannte wirtschaftliche Aspekt ist für Hasitschka, Teichmann und Sneed ein kleineres Problem, aber wenn Änderungs- oder Erweiterungsanträge als "Mängel" akzeptiert werden, ist die **Gefährdung der Produktstabilität** viel gefährlicher. „Bei der Umsetzung eines Änderungs- oder Erweiterungsantrages ist der Produktverantwortliche in der Regel unter weniger Zeitdruck und geht daher geordneter vor. Eine Mangelbehebung passiert dagegen immer aus der Defensive heraus. Der Lieferant ist vertraglich an Behebungsfristen gebunden, die ihn möglicherweise daran hindern, lange genug nach einer stabilen, für alle Anwender optimalen Lösung zu suchen“ [HaTS2004, 164f].
- Natürlich sind hohe Mangelziffern nicht gerade Gründe, dem für die Software verantwortlichen einen guten Ruf zu attestieren. „Das Verhältnis zu den vorhandenen Anwendern wird belastet, die Chancen auf neue sinken. Das ist umso ärgerlicher für ihn, wenn diese Statistiken Mängel enthalten, die keine sind“ [HaTS2004, 165].
- Aus den bereits angeführten Gründen, resultiert auch noch ein weiterer. Es ist wohl nur natürlich, dass unscharfe Formulierungen in den Mangeldefinitionen von Kunden anders ausgelegt werden als vom Hersteller. Diese unterschiedlichen Auffassungen und die unweigerlich darauf folgende Diskussion kostet beide Seiten Zeit, Ressourcen und verschlechtert das Klima der Vertragsparteien. „Je schärfer die Mangeldefinitionen sind, desto weniger Reibungsverluste gibt es an dieser Stelle“ [HaTS2004, 165].

„Es gibt also gute Gründe, sich um eine möglichst klare Grenzziehung zwischen Mangel und Nichtmangel zu bemühen. Sie ist im Interesse aller Beteiligten:

- Der Produktverantwortliche erspart sich unnötige Diskussionen mit dem meldenden Anwender. Er kommt nicht in die Gefahr, Erweiterungswünsche als "Mängel" umsetzen zu müssen und dadurch Einnahmen zu verlieren. Er kann seine Verbesserungsmaßnahmen zielgerichtet steuern.
- Der meldende Anwender erspart sich ebenfalls Streitgespräche und kommt dadurch schneller zu einer Lösung.
- Die anderen Anwender bekommen ein stabileres Produkt und profitieren überdies von den besser fokussierten Qualitätsverbesserungsmaßnahmen des Lieferanten.

An dieser Stelle soll aber auch klar gesagt werden, dass Entzerrungen der Qualitätsaussagen kein Ersatz für laufende Verbesserungen des Produktes selbst sein können. Die beste Methode zur Reduktion der Mangelzahlen ist nicht, an den Mangeldefinitionen zu arbeiten, sondern das Produkt und die Verfahren zu verbessern“ [HaTS2004, 165].

5.4 Planung

Dieser Teil der Arbeit beschäftigt sich mit den Anforderungen an den TT-Workflow im Bezug auf die Planung und die endgültige Umsetzungsentscheidung. Im Gegensatz zum Mangel hat der Softwareproduzent beim Erweiterungsauftrag die Möglichkeit steuernd einzugreifen. Mängel müssen, wenn sie die entsprechende Wichtigkeit besitzen, behoben werden und so den Vorzug gegenüber Erweiterungsaufträgen erhalten.

Zur Illustration dieser Aussage folgendes Beispiel:

Sind noch zehn Personentage (PT) an Kapazität bis zur nächsten Auslieferung verfügbar, und die Umsetzung eines Erweiterungsauftrages benötigt sieben Tage, so kann dieser nicht umgesetzt werden, wenn ein schwerwiegender Mangel, der unter Umständen den täglichen Betrieb bei einem Kunden stört, mit dem Aufwand von vier PT gemeldet wird.

Der Grund des Vorzuges ist schlicht daran festzumachen, dass die Produktstabilität kurz vor der Produktfreigabe eine höhere Priorität besitzt als die Funktionalität [Pink2001, 18].

Selbst die Möglichkeit generell im Vorhinein Aufwand für erwartete Mängel zu planen, ist in der Praxis nicht möglich. Wer weiß schon wie viele produktionsverhindernde Mängel in der nächsten Woche oder dem nächsten Monat gemeldet werden. Man kann sich nur auf Erfahrungswerte berufen und so etwa zum Beispiel die Hälfte der Entwicklerzeit für Mängel reservieren und den Rest für Erweiterungsaufträge einplanen. Weiters ist es ratsam kurz vor der Fertigstellung einer Version mehr Zeit für die Mangelbehebungen zu reservieren und die Umsetzungsarbeiten von Erweiterungsaufträgen bereits einige Zeit vor der entsprechenden Lieferung abzuschließen, da ja auch Erweiterungsaufträge selbst für neue Mängel sorgen können.

Man muss also sagen, dass die Einbindung von Planungsaktivitäten in den Workflow eines Tickets nur bei Erweiterungsaufträgen Sinn macht. Hier ist auch die Aufwandsschätzung, ohne die keine Planung vorstellbar wäre, von viel größerer Bedeutung, da diese dann die Grundlage für ein Umsetzungsangebot an den Kunden darstellt.

Welche Anforderungen entstehen nun an den TT-Workflow zur Unterstützung der Planung und Angebotslegung? Folgende Probleme sollen hierzu behandelt werden:

- Grobkonzepte müssen zur Verfassung von Spezifikationen angeboten und abgenommen werden können.
- Aufwand und Risiko für die Umsetzung müssen erhoben werden um ein entsprechendes Angebot für den Erweiterungsantrag legen zu können.
- Ein Angebot muss gelegt werden können und die Möglichkeit dieses anzunehmen, abzulehnen oder nachzubessern muss vorhanden sein.

5.4.1 Grobkonzept (GK)

Will der Kunde eine Erweiterung der Funktionalität des Softwareproduktes, so müssen die Anforderungen für dieses neue Stück Software abgeklärt werden. Würde nun schon das gesamte Konzept erstellt werden, um ein Angebot für diesen neuen Teil zu berechnen, so wäre der Aufwand zu diesem Zeitpunkt beträchtlich. Dieser Umstand ist sowohl für den Kunden als auch den Hersteller nicht zumutbar. Aus diesem Grund wird, nachdem die Details für den Kundenwunsch geklärt sind, dem Kunden ein Angebot für ein Grobkonzept unterbreitet. Dieses Grobkonzept beinhaltet die geplanten sichtbaren Änderungen an der Software, ohne aber zu sehr ins Detail zu gehen. In der Regel muss der Preis dafür nur bezahlt werden, wenn das Angebot für die tatsächliche Umsetzung des Erweiterungswunsches nicht angenommen wird.

Welche Auswirkungen hat nun der Umstand, dass ein Grobkonzept bei Erweiterungsaufträgen einen großen Teil der Umsatzbeauftragung darstellt.

Die erste Forderung muss sich mit der Beauftragung eines Grobkonzepts beschäftigen. Der Kunde muss die Möglichkeit haben, für ein bestimmtes TT ein Grobkonzept in Auftrag zu geben, nachdem ihm der Preis für dieses bekannt gegeben wurde. Voraussetzung hierfür ist aber eine bereits getroffene Übereinkunft, dass es sich um einen Erweiterungswunsch handelt. Auch macht es keinen Sinn, ein Grobkonzept beauftragen zu müssen, wenn das Grobkonzept gratis ist.

Wurde das Grobkonzept beauftragt, so ist nun dessen Erstellung an der Reihe. Der zuständige Mitarbeiter beschreibt also welche Änderungen an der Software vorgenommen werden und welche Auswirkungen diese auf das bestehende Produkt haben werden. Hat er seine Arbeit beendet, so wird das Grobkonzept dem Kunden zur Abstimmung übermittelt. Wieder tritt eine Verhandlungssituation ein, wobei in dieser nur der Inhalt des Grobkonzepts abgestimmt werden muss. Ist der Kunde mit dem Grobkonzept einverstanden, so akzeptiert er den Inhalt. Ist er nicht einverstanden, so muss dass GK verändert werden oder unter Umständen sogar gänzlich neu geschrieben werden. Mit der Annahme des Grobkonzepts steht damit auch der Inhalt des Erweiterungswunsches außer Frage.

5.4.2 Aufwand und Risiko

Die einzig offenen Fragen, bevor dem Kunden ein verbindliches Angebot vorgelegt werden kann, lauten:

- Auf welche Höhe belaufen sich die Kosten für die Umsetzung des Erweiterungsantrages?
- Zu welchem Termin, bzw. mit welcher Lieferung erhält der Kunde die vereinbarte neue Funktionalität?

Die beiden Fragen sind untrennbar miteinander verbunden, da der Aufwand den die Änderung verursachen wird sowohl die Kosten maßgeblich beeinflusst, als auch den Liefertermin, natürlich unter Berücksichtigung der verfügbaren Ressourcen, bestimmt.

Die durch diesen Umstand abgeleitete Anforderung an den TT-Lebenszyklus kann sich demnach in erster Linie nur mit der Aufwandsschätzung beschäftigen. Zu diesem Thema sind bereits unzählige Arbeiten verfasst worden, für unsere Zwecke reicht es jedoch zu sagen, dass Phasenmodell muss eine Aufwandsschätzung nach der Grobkonzeptannahme aufweisen, wie der Aufwand ermittelt wird ist in diesem Zusammenhang nicht von Bedeutung.

Ist die Schätzung fertig, so muss als nächste Forderung die nach der eigentlichen Planung angeführt werden. Nach einer Gegenüberstellung des Aufwands mit den zu diesem Zeitpunkt noch verfügbaren Ressourcen trifft in der Regel eine eigens für die Planung zuständige Abteilung die Entscheidung, zu welchem Zeitpunkt der Kunde die neue Funktionalität zu seiner Verfügung haben wird. Dieser Zeitpunkt ist dann für den Softwarehersteller verbindlich. Wichtig für die Anforderung an den Workflow ist der Umstand, dass die Planung beinahe zu 100 Prozent von anderen Personen durchgeführt wird als die Aufwandsschätzung.

5.4.3 Angebot

Nachdem die Planungen abgeschlossen sind und durch das Wissen von Aufwand, möglichem Fertigstellungszeitpunkt und Inhalt des Erweiterungswunsches, ist nun die Legung eines Angebots möglich.

In der Regel wird das Angebot wieder durch eine andere Abteilung, als die, die für Planung oder Schätzung zuständig waren, erstellt. Da die meisten für die Angebotserstellung notwendigen Informationen schon im Ticket selbst festgehalten worden sind, empfiehlt sich eine automatische Angebotsgenerierung. Das dadurch erstellte Angebot wird anschließend um bestimmte Attribute, wie zum Beispiel Stundensätze, Rechtsgrundlagen, usw. ergänzt und kann danach, z.B. als Attachment, im Ticket selbst dem Kunden übermittelt werden.

Wurde das Angebot gelegt, so ist nun wieder der Kunde an der Reihe und eine weitere Verhandlungssituation tritt ein. Da zu diesem Zeitpunkt der Inhalt des Erweiterungswunsches bereits vereinbart ist (siehe Kapitel 5.4.1), sind die strittigen Punkte die Kosten und der Liefertermin. Eine immer wieder auftauchende Forderung der Angebotsersteller ist die Möglichkeit, das Ticket der Planungsabteilung zurückzuschicken um eventuell auftretende neue Terminwünsche des Kunden prüfen lassen zu können.

Die Annahme des Angebots erfolgt meist schriftlich, da es um teilweise sehr hohe Beträge geht, die Unterschrift eines Zeichnungsberechtigten von Nöten ist und 100 prozentige Rechtssicherheit angestrebt wird. Natürlich könnten diese Geschäfte heutzutage auch elektronisch abgewickelt werden, in diesem Fall hat sich die Geschäftsführung des Softwarehauses jedoch für eine schriftliche Annahme entschieden. Auswirkungen auf den TT-Lebenszyklus hat diese Entscheidung nur eine geringe, da nun eben die Angebotsannahme nach dem Eintreffen des unterfertigten Angebots durch Mitarbeiter des Produzenten ins TTS eingetragen werden.

5.5 Umsetzung

Durch die Annahme des Angebots ist nun auch die Umsetzungsentscheidung für einen Erweiterungsauftrag gefallen. Dies führt dazu, dass nun die Anforderungen an den TT-Workflow sowohl seitens des Mangels als auch aus der Sicht des CRs wieder auf einer gemeinsamen Schiene laufen.

Wie schon im Kapitel 2 ausführlich erwähnt, ist vor allem die Umsetzung eines Problems Sache des Change-Managements. Die Problemfelder, die die Anforderungen an den TT-Workflow bestimmen, kommen daher in diesem Kapitel zu einem großen Teil aus dem Bereich des Zusammenspiels zwischen dem PM und dem CM. Konkret sind das:

- die Notwendigkeit von Schnittstellen, die Phasen aufgrund von Vorgängen im CM automatisch weiterdrehen, und
- die Unterscheidung der Umsetzung in die Ausarbeitung von Änderungsanträgen und die tatsächlich Realisierung bzw. Codierung.

Eine weitere Anforderung, die ich erst im Kapitel 5.8.2 näher beleuchten werde, die aber in diesem Zusammenhang ebenfalls auftritt, ist in dem Umstand zu finden, dass ein Softwarehaus kein Interesse daran hat, dass die Kunden erkennen können, in welcher Phase ein Problem während der Umsetzung und dem Test gerade liegt. Diese einzelnen Arbeitsgänge müssen für den Kunden wie einer aussehen.

5.5.1 Ausarbeitung von Änderungsanträgen (RfC)

Die Anforderung, dass der TT-Workflow auch die Ausarbeitung von Änderungsanträgen abzubilden hat, liegt auf der Hand. Durch die positive Umsetzungsentscheidung müssen so genannte Requests for Change durch die Konzeptabteilung ausgearbeitet werden. Diese stellen dann eine Art Arbeitsauftrag für die Realisierungsabteilung dar. Der Workflow muss in der Lage sein, nach der Umsetzungsentscheidung das Ticket an die Konzeptabteilung weiterzuleiten, um die RfCs auszuarbeiten.

5.5.2 Schnittstellen zum CM

Die meisten Softwareproduzenten verwenden zur Unterstützung des Change-Managements so genannte Change- und Configuration-Management-Tools. Ein Beispiel für ein solches Tool ist CM-Synergy. Weitere Tools und eine kurze Zusammenfassung über deren Leistungsumfang findet man auf der Internetseite von Joachim Hagelberger [Hage2004].

Für diese Arbeit von Bedeutung sind nicht die CM-Tools an sich sondern die Übergänge vom PM-Tool zum CM-Tool und zurück. Um dieses Problem genauer untersuchen zu können, soll folgendes Beispiel dienen:

Zur leichteren Versionierung und Zusammenstellung von Softwareteilen, benützt das Softwarehaus X ein CM-Tool. Nachdem Probleme abgestimmt wurden und RfCs für das CM ausgearbeitet wurden, müssen diese dem CM übergeben werden. Dies erfolgt durch Ausdrucken der Vorschläge und Weiterleitung an die Realisierer mittels interner Hauspost. Nachdem die RfCs fertig realisiert wurden, durch das CM-Tool kompiliert und auf einem Server zum Test zur Verfügung gestellt werden, wird der Testabteilung eine Liste mit den durch den letzten Compile fertig gestellten RfCs übermittelt. Diese prüfen nun ob bereits alle RfCs zu einem gemeldeten Problem fertig gestellt wurden, um anschließend einen abschließenden Test des Problems vorzunehmen.

Wenn man sich dieses Beispiel vor Augen hält, kann man sehr deutlich erkennen, wo die Anforderungen für den TT-Workflow ansetzen müssen:

- Um keine Informationen, die mit einem bestimmten Ticket zusammenhängen, zu verlieren, müssen sämtliche RfCs auf dem TT gespeichert werden. Nach Abschluss der RfC-Ausarbeitung, muss das Ticket an die Realisierer weitergeleitet werden können.
- In diesem Zusammenhang eine weitere Forderung ist die Möglichkeit, von einem RfC automatisch einen Task im CM-Tool anlegen zu können, wobei diese Funktionalität keine Auswirkung auf den TT-Workflow hat.

- Eine Auswirkung auf den Lebenszyklus hat jedoch die letzte Anforderung zu diesem Themenkomplex, die eine automatische Retourschnittstelle von CM-Tool ins PM-Tool fordert. Um der Testabteilung die mühsame und oft fehleranfällige Prüfung, in der untersucht wird ob bereits alle RfCs umgesetzt wurden, zu ersparen, muss die Schnittstelle immer dann ein Ticket an die Testabteilung automatisch weiterleiten, wenn alle RfCs umgesetzt wurden. Andernfalls könnten Probleme nachgetestet werden, die noch gar nicht fertig umgesetzt wurden. Dies würde zu einer Verschwendung von Ressourcen und damit verbunden zu nicht notwendigen Kosten führen. Der bei weitem schlimmere Fall ist jedoch, dass Probleme, die bereits fertig umgesetzt wurden, schlicht und einfach vergessen werden. Die Auswirkungen dieses Falls wären, dass die Problembehebung nicht an den Kunden zurückgemeldet wird, dass das Problem nicht nachgetestet wird und dass damit die Kundenunzufriedenheit steigt.

5.6 Test und Lieferung

Ist die Umsetzung eines Tickets erfolgt, und hat die Schnittstelle zum CM das Ticket an die zuständigen Tester weitergeleitet, so fehlen noch drei Schritte, damit der Kunde die neue Software, die nun die in der Zwischenzeit behobenen Probleme beinhaltet, zur Verfügung hat.

Diese Schritte gliedern sich wie folgt:

- Nachtesten der Probleme im internen Testsystem,
- zusammenfassen der einzelnen Problem zu einer Lieferung und
- Übermittlung der neuen Software.

5.6.1 Interner Test

Schon im Kapitel 2.2.1 habe ich über den Test geschrieben:

„Den Abschluss eines Problems übernimmt die Phase des Post Implementation Review. Diese stellt eine Endkontrolle der Änderungen durch das CM dar, prüft vor allem die Vollständigkeit der Umsetzung und ob diese ordnungsgemäß durchgeführt wurde und fungiert in diesem Zusammenhang als Qualitätssicherungsprozess.“

Diese Endkontrolle muss der TT-Workflow natürlich ebenfalls unterstützen. Welche Möglichkeiten müssen nun dem Tester durch den Ticket-Lebenszyklus zur Verfügung gestellt werden, nachdem das Ticket an ihn weitergeleitet wurde?

Diese Frage ist in diesem Zusammenhang sehr komplex. Es wäre das Einfachste zu sagen, das Problem ist nun behoben oder es ist nicht behoben. Doch ganz so einfach stellt sich dieses Problem leider nicht dar.

Zur Verdeutlichung ein kleine Beispiel:

Eine Bank kauft vor 15 Jahren ein Softwareprodukt zur Abwicklung ihres täglichen Geschäfts. Mit dem Aufkommen der Eurodiskussion und der zu erwartenden Euroeinführung, meldet sie ein Problem an das Softwarehaus, von dem die Bank das Produkt gekauft hat. Es steht außer Frage, dass es sich hier um einen CR handelt, da dieser Erweiterungswunsch aber unabdingbar für die Bank ist, wird das Angebot angenommen und die Umsetzung dieses CRs vereinbart. Nachdem alle RfCs umgesetzt wurden, landet der CR in der Testabteilung. Der Test ergibt, dass die Basisfunktionalität hervorragend funktioniert, aber die Beschriftung von zwei Dialogen nicht richtig ist und zwei kleinere Reports noch Fehler enthalten.

Wie soll nun ein Tester reagieren? Hätte er nur die Möglichkeit Test erfolgreich oder Test fehlgeschlagen, so müsste er den Test negativ bewerten, obwohl die dringend benötigte Software eigentlich einsatzfähig wäre und nur noch kleine Probleme auftreten. Dieser Umstand führt zu der Forderung an den TT-Workflow, dass folgende Möglichkeiten für den Tester zur Verfügung stehen müssen:

- Das Problem wurde erfolgreich umgesetzt.
- Das Problem wurde mangelhaft umgesetzt, ist aber dennoch einsatzfähig. Diese Option muss die Möglichkeit bieten neue Probleme zu diesem Problem zu erfassen und das eigentliche Problem, mit der Zusatzangabe an den Kunden dass noch kleiner Probleme offen sind, für die Lieferung bereitzustellen.
- Das Problem wurde nicht erfolgreich umgesetzt und muss damit verbunden an die Umsetzung zurückgeschickt werden.

5.6.2 Warten auf die Lieferung

Die Anforderung des „Zusammenwartens“ von Problemen ist schnell erklärt. Es ist nicht möglich dem Kunden sofort nach einem erfolgreichen Test jede Problembehebung zu übermitteln. Sie müssen zu Paketen zusammengeschnürt werden. Damit dies realisierbar ist, muss der TT-Workflow eine Möglichkeit schaffen, dass behobene Probleme eine Eigenschaft bekommen, die besagt dass dieses Problem mit der nächsten Lieferung an den Kunden mitgeliefert werden soll.

5.6.3 Gemeinsame Lieferung

Hat man einen Liefertermin erreicht, so wird dem Kunden in welcher Art auch immer ein neues Stück Software bereitgestellt. Zu diesem Zeitpunkt ist der Lebenszyklus des Tickets gefordert. Er muss nun dem Kunden zeigen, welche Probleme mit dieser Lieferung behoben wurden. Zu diesem Zweck müssen alle Probleme, die auf diese Lieferung gewartet haben in einen Zustand gebracht werden, der dem Kunden sagt, dass diese Probleme nun nicht wieder auftreten sollten.

5.7 Kundentest

Normalerweise müsste man nun sagen, dass der TT-Lifecycle an dieser Stelle endet. Dies ist auch beinahe der Fall nur wird zu gerne ein wichtiger Faktor übersehen. Nur weil der interne Test das Problem als erledigt identifiziert hat, heißt dies nicht, dass der Kunde der gleichen Meinung ist. Es ist auf der einen Seite denkbar, dass der Kunde eine andere Auffassung des Problems hatte, oder dass das Problem im System des Kunden erneut auftritt, obwohl es im System des Produzenten behoben war. Aus diesen Umständen ergeben sich wieder einige Aspekte, die ich in diesem Kapitel näher betrachten will:

- Jeder Kunde muss die erfolgreiche Umsetzung eines Problems bestätigen, ähnlich wie die Gegenzeichnung eines Lieferscheins auf einer Baustelle.
- Die Umsetzungsbestätigung beim Mangel unterscheidet sich von der des CRs in einigen Fällen.
- Ein Mangel kann wiedereröffnet werden, wenn er in gleicher Weise wieder auftritt, da sich in diesem Fall das Problem in keiner Hinsicht verändert hat. Auch die Beurteilung solcher Wiedereröffnungen muss durch den Lebenszyklus abgedeckt werden.

5.7.1 Notwendigkeit einer Bestätigung

Eine der Hauptaufgaben eines TTS ist die Dokumentation. Dieser Umstand ist auch der Grund, warum der Lebenszyklus des Tickets nicht mit der Auslieferung endet. Natürlich wäre es legitim zu sagen, dass die Kunden sich selbst um das Nachtesten von Problemen kümmern müssen und bei einem negativen Test einfach ein neues Problem melden sollen.

Der Vorteil in der Zurverfügungstellung von Kundentestphasen im TT-Workflow liegt darin, dass auch sämtliche Informationen des Kundentests am Ticket gespeichert werden und somit die Hauptaufgabe des Dokumentierens abgedeckt wird.

5.7.2 Abnahme des Erweiterungsantrages

Das Nachtesten eines CRs, inkludiert auch gleichzeitig die Abnahme des CRs. Das heißt in den meisten Fällen, dass damit auch der Bezahlung des Erweiterungsantrages nichts mehr im Wege steht. Die Möglichkeiten, die ein Kunde im Zusammenhang mit der Abnahme eines CRs fordert sind folgende:

- Die Abnahme eines CRs muss möglich sein.
- Es muss möglich sein, neue Probleme mit einem Bezug zu dem CR melden zu können und gleichzeitig zu vermerken, ob dieses Problem den Einsatz der neuen Funktionalität gefährdet oder es sich um ein leichteres Problem handelt, dass die Abnahme nicht in Gefahr bringt.
- Sind alle Probleme, die die Abnahme verhindert haben erledigt, so muss der zuständige Kundentester informiert werden.

Gleichzeitig gibt es auch Anforderungen seitens des Softwareproduzenten an den Lebenszyklus eines TT zu diesem Thema:

- Automatische Abnahme von CRs, wenn in einer gewissen Zeitspanne keine Probleme gemeldet werden, die die Abnahme verhindern. Dieser Punkt ist sehr heikel, da sich dadurch die Kunden oft unter Druck gesetzt fühlen, auf der anderen Seite müssen aber den Kunden gewisse Grenzen gesetzt werden, sonst könnte jeder Kunde einen CR auf unbestimmte Zeit liegen lassen und damit der Bezahlung entgehen.
- Die automatische Abnahme muss nach Behebung von Problemen, die die Abnahme verhindert haben, wieder einsetzen.

5.7.3 Bestätigung der Mangelbehebung

Anders als beim Erweiterungsantrag, ist keine Zahlung mit dem erfolgreichen Nachtesten des Problems verbunden. Ähnlich wie beim internen Test eines Mangels wie im Kapitel 5.6.1 beschrieben, fordert ein Kunde für seinen Test ebenfalls folgende Möglichkeiten:

- Die Bestätigung der erfolgreichen Umsetzung eines Problems.
- Die Bestätigung der ausreichend erfolgreichen Umsetzung eines Problems. Hier muss wieder die Möglichkeit vorhanden sein, neue Probleme zu diesem Problem zu erfassen und das eigentliche Problem auf erledigt setzen
- Die erneute Meldung des Problems, da dieses unverändert auftritt und nicht behoben wurde.

Der letzte Fall ist zweifellos der für den Produzenten schlechteste. Das beschriebene Problem ist in seiner Ausprägung vollkommen unverändert wieder aufgetreten. In dieser Situation spricht man von einer Wiedereröffnung oder dem Reopen eines Mangels.

Die Anforderungen, die für so einen Fall an den TT-Workflow gestellt werden, sind folgende:

- Das Problem muss in die Beurteilung zurückgeschickt werden, und die Verhandlungen beginnen von vorne.
- Fachlich gesehen muss das Problem als neues Ticket angesehen werden, nur sind die Beschreibung und die bisher gesammelten Daten mitzunehmen.
- Die Beurteilung des wiedereröffneten Mangels muss genauso abgelehnt oder zurückgezogen werden können wie ein neues Problem.
- Der Rückzug eines solchen Problems darf nicht gleich bedeutend sein mit dem Rückzug des ursprünglichen Problems.
- Aus Managementsicht ist es wichtig, einem Problem anzusehen ob die Behebung erfolgreich oder nicht erfolgreich war. Dies dient vor allem der Verhandlungsposition bei Kundenbeschwerden. Es ist ein großer Unterschied, ob die letzten 20 Probleme nur zur Hälfte umgesetzt wurden oder zu beinahe 100 Prozent.

5.8 Weitere Anforderungen

Neben den tätigkeitsbedingten Anforderungen an den TT-Workflow, die ich bislang behandelt habe und sich aufgrund von vorzunehmenden Arbeitsschritten in der jeweiligen Bearbeitungsstufe eines Tickets ergeben, müssen unbedingt noch zwei weitere Anforderungen an den Lifecycle des TT herangetragen werden, die nicht mit einzelnen Arbeitsschritten und der damit verbundenen Weiterleitung an zuständige Personen in Zusammenhang stehen. Trotzdem haben diese Forderungen sehr großen Einfluss auf den letztendlich ausgearbeiteten optimalen Lebenszyklus (siehe dazu Kapitel 6) eines Tickets.

Konkret ergeben sich die Anforderungen aufgrund folgender Probleme:

- Probleme können verschiedenste Beziehungen zueinander haben, beispielsweise haben sie den gleichen Inhalt. In diesen Fällen muss der Workflow automatische Phasendrehungen zu seinen Brüdern machen.
- Manche Unterteilungen von Arbeitsschritten, die zum Beispiel von zwei verschiedenen Abteilungen vorgenommen werden, aber dennoch eigentlich einen größeren gemeinsamen Arbeitsschritt darstellen, sollen dem Kunden nicht bekannt gegeben werden. Auf der anderen Seite soll leicht erkennbar sein, ob das produzierende Softwarehaus den nächsten Arbeitsschritt setzen muss oder der Kunde.

5.8.1 Verknüpfungsmöglichkeiten

Oft kommt es vor, dass mehrere Kunden das gleiche Problem melden. Für so einen Fall muss natürlich genauso eine Verhandlung mit jedem Kunden stattfinden und in einer Übereinkunft enden. Dennoch muss das Problem natürlich nicht mehrmals behoben werden. Für einen solchen Fall muss das TTS eine Verknüpfungsmöglichkeit vorsehen, mit der man die Beziehung einzelner Probleme untereinander abbilden kann. Andere Verknüpfungsmöglichkeiten wären z.B.:

- „Umsetzung von Problem A ist Voraussetzung für Umsetzung von Problem B“ bzw. in die andere Richtung „Umsetzung von Problem B hat als Voraussetzung die Umsetzung von Problem A“
- „Umsetzung von Problem A führte zu Problem B“ bzw. „Problem B entstand aus Umsetzung von Problem A“
- „Problem A trat auf beim Nachtesten von Problem B“

Für den Workflow des TTS sind vor allem zwei Verknüpfungen wichtig, die auch die Anforderungen an den TT-Lebenszyklus bestimmen:

- Haben zwei Probleme den gleichen Inhalt, so müssen sie die Umsetzung, den Test und die Lieferung gemeinsam durchleben. Das heißt, dass ein Ticket als Master funktioniert und sich die mit diesem Typ verknüpften Tickets einfach mitziehen lassen, ohne dass ein weiterer manueller Eingriff notwendig ist.
- Ist die Umsetzung eines Problems A Voraussetzung für die Umsetzung eines anderen (in diesem Fall das Problem B), so darf B erst dann in den internen Test geschickt werden, wenn A zumindest bereits im Test ist.

5.8.2 Verschiedene Phasenmodelle

Die Anforderung, die ich in diesem Kapitel detaillierter betrachten möchte, lautet:

Der Lebenszyklus eines Trouble-Tickets muss intern anders gestaltet werden können als extern. Dieser Fall tritt in mehreren Situationen ans Tageslicht:

- Während der Abstimmung muss anhand der Phase erkennbar sein, ob die jeweilige Organisation an der Reihe ist oder auf einen Kommentar des anderen wartet.
- In der Umsetzung liegt es nicht im Interesse des Produzenten, dass der Kunde genau mitverfolgen kann, ob ein Problem gerade beim Konzeptionisten, dem Realisierer, dem Test oder gar auf der Wartebank vor der Auslieferung liegt.
- Gleiches gilt für einen Erweiterungsauftrag, wo nicht transparent sein soll, ob der Aufwand geschätzt wird, die Planung erfolgt oder das Angebot gerade geschrieben wird.
- Ein weiterer Fall ist die konzentrierte Freigabe. In diesem Fall soll das Problem ja noch gar nicht beim Produzenten bekannt sein.

Diese Fälle führen dazu, dass es unumgänglich ist, die Möglichkeit zu schaffen, mehrere Phasen zu einer zusammenzufassen und die Phasen auf der Kundenseite anders benennen zu können als die auf der internen Seite.

6 Optimaler Lifecycle

An dieser Stelle werde ich nun einen Lebenszyklus für ein Trouble-Ticket vorstellen, der sämtliche Anforderungen des vorherigen Kapitels abdeckt.

Des Weiteren soll der Workflow auch als Basis dienen, um dem Problemmanager eines Softwarehauses eine bestmögliche Unterstützung bei der Evaluation von TTS zu gewähren, da dieser schon im Vorhinein prüfen kann, ob der hier vorgestellte Zyklus auch von den evaluierten Systemen abgedeckt werden kann.

Last but not least stellt der Lifecycle natürlich auch die Basis für den durch den Problemmanager einzusetzenden Workflow dar. Der hier vorgestellte Lebenszyklus ist in jedem Softwarehaus einsetzbar und ermöglicht aber dennoch die eine oder andere Adaptiermöglichkeit, um besondere Spezifika eines Softwarehauses einfließen zu lassen.

Beginnen möchte ich dieses Kapitel mit den Grundsätzen, die ich bei der Erarbeitung dieses Modells berücksichtigt habe, und den Darstellungsregeln.

Die weiteren Unterkapitel beschäftigen sich dann mit den 5 großen Blöcken des Workflows, die so wie in Abbildung 6 dargestellt aufeinander folgen.

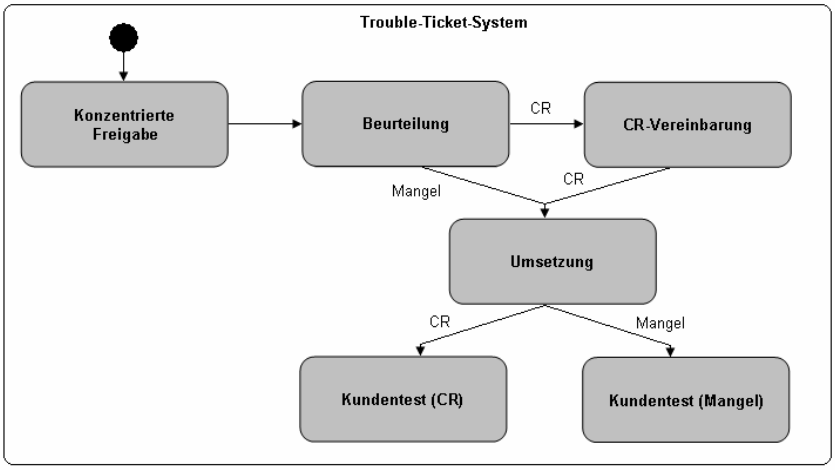


Abbildung 6: Grobes Phasenmodell

6.1 Allgemeines zum Modell

Die allgemeinen Anmerkungen konzentrieren sich auf Grundsätze und Darstellungsarten. Die Grundsätze beinhalten in erster Linie fachliche Aspekte, die der besseren Verständlichkeit dienen, während sich die Darstellung um eine erhöhte Lesbarkeit kümmert.

6.1.1 Grundsätze

Grundsätze eines Phasenmodells sind Vorsätze, die sich durch das gesamte Modell ziehen und dem Anwender, aufgrund von zum Beispiel bestimmten Wörtern oder farblichen Kennzeichen, schon auf den ersten Blick einige Informationen geben. Welche Grundsätze liegen nun dem hier erarbeiteten Zyklus zugrunde?

Grundsatz 1

Der erste Grundsatz bestimmt bereits die Bezeichnung der Phasen. Beginnt eine Phase mit dem Wort „Warten“, so bedeutet dies, dass das Problem auf eine Aktion wartet. Diese Aktion kann entweder von einem anderen Tool, z.B. dem Change-Management Tool, von einem anderen Problem oder von dem korrespondierenden Gegenüber (entweder Vertreter des Kunden oder des Produzenten) kommen. In manchen Fällen steht dem User auch in einem „Warten“ - Zustand eine Möglichkeit zur Phasendrehung zur Auswahl. Diese Drehung ist aber durchwegs eine Ausstiegsvariante. Wird sie nicht benutzt, so greift früher oder später die fremd gesteuerte Drehung.

Grundsatz 2

Jede Phase, die nicht mit dem Wort „Warten“ beginnt, verlangt nach einer Aktivität durch den Zuständigen. Die verlangte Tätigkeit wird bestmöglich mit der Bezeichnung der Phase beschrieben und endet mit dem manuellen Bestätigen durch den Bearbeiter, dass dieser Arbeitsschritt nun abgeschlossen ist.

Grundsatz 3

Ein weiterer Grundsatz erfüllt die Anforderung aus dem Kapitel 5.8.2, bei dem die Forderung nach zwei verschiedenen Phasenmodellen gestellt wurde. Jede Phase kann 2 Bezeichnungen haben. Sieht sich der Kunde das Problem 1 an, so kann die Phase eine andere Bezeichnung zeigen, als wenn ein Vertreter des Herstellers das Problem öffnet. Technisch ist dieses Problem leicht zu lösen, indem als Phase ein eindeutiger technischer Schlüssel vergeben wird und je nach User die zugehörige Bezeichnung für den Kunden oder den Hersteller angezeigt wird.

Diese Information wird in einer Tabelle abgelegt, die 3 Attribute aufweist:

- Technischer Schlüssel (z.B. Phasennummer)
- Bezeichnung aus Hersteller Sicht
- Bezeichnung aus Kunden Sicht

Des Weiteren ist es mit dieser Variante auch möglich, mehreren Phasen die gleiche Bezeichnung z.B. auf der Kundenseite zuzuordnen.

Grundsatz 4

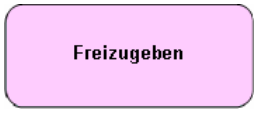
Dieser Grundsatz behandelt die häufig gestellte Frage nach unterschiedlichen Phasenmodellen von intern gemeldeten Problemen und extern gemeldeten Tickets. Diese Frage ist in diesem Modell leicht zu beantworten. Das interne Testcenter wird ganz einfach als Kunde behandelt. Das bedeutet, dass ein Tester des Softwareproduzenten die Phasenbezeichnungen der Kunden im TT angezeigt bekommt und auch Phasendrehungen auf Kundenseite vornimmt. Dies erreicht man durch eine Steuerung über die UserIDs. IDs haben ein Attribut, dass die Seite, die ein User im TTS einnimmt, steuert. Dadurch ist es auch möglich, dass ein User 2 IDs besitzt und damit sowohl als Kunde als auch als Hersteller-Vertreter auftreten kann. Ob interne Anweisungen diesen Problemen eine höhere oder niedrigere Priorität einräumen ist für das Phasenmodell nicht von Bedeutung und kann mit einfachen Dienstanweisungen geregelt werden.

6.1.2 Darstellung

Die Darstellung versucht, die Lesbarkeit des Modells so gut wie möglich zu gewährleisten. Dies ist auch der Grund, warum das Modell auf verschiedene Blöcke aufgeteilt und eine farbliche Unterscheidung der Phasen vorgenommen wurde.

In dem hier vorgestellten Lebenszyklus werden folgende Symbole für Phasen bzw. Zustände verwendet:

Diese Darstellung zeigt die Bezeichnung des Zustands in der Mitte. Die farbliche Kennzeichnung weißt darauf hin, dass es sich um keinen Unterzustand handelt. Da Zustände in verschiedene sequentielle oder parallele Unterzustände aufgesplittet werden können [UML], macht auch der Lebenszyklus von dieser Möglichkeit gebrauch. Das heißt, dass alle anderen Symbole Unterzustände darstellen und nur dieses Symbol einen „Haupt“ - Zustand darstellt.



Diese Darstellung zeigt die Bezeichnung der Phase in der Mitte. In diesem Fall heißt die Phase „Freizugeben“. Die farbliche Kennzeichnung weißt darauf hin, dass diese Phase nur auf der Seite des Kunden existiert. Wie bereits erwähnt, stellt dieses Symbol einen Unterzustand dar.

Auch diese Darstellung zeigt einen Unterzustand, aber im Gegensatz zum letzten Symbol weist dieses 2 verschiedene Bezeichnungen auf. In diesem Fall heißt die Phase auf der Seite des Produzenten „GK erstellen“ und der Kunde kennt diese Phase als „Warten auf GK-Erstellung“.

Diese Darstellung zeigt die Bezeichnung der Phase unter dem Symbol, das in UML für einen Endzustand steht.

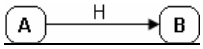
Dies bedeutet, dass das Problem als endgültig abgeschlossen anzusehen ist. In der Graphentheorie würde man von einer Senke sprechen.

Diese Darstellung zeigt die Bezeichnung der Phase wieder in der Mitte. Die farbliche Kennzeichnung weißt darauf hin, dass diese Phase wieder einen Unterzustand darstellt, aber dennoch auf beiden Seiten die gleiche Bezeichnung aufweist.

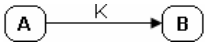
Weiters werden Pfeile dazu verwendet, die einzelnen Phasen zu verbinden und die möglichen Phasenübergänge darzustellen. Ein Pfeil von Phase A zu Phase B bedeutet, dass es eine Möglichkeit gibt, von Phase A auf die Phase B zu drehen. Ein Doppelpfeil zwischen A und B, würde die Möglichkeit darstellen, dass man sowohl von A auf B drehen kann als auch wieder zurück auf A.

Der Pfeil gibt zusätzlich noch die Auskunft darüber, wer die Möglichkeit besitzt die Phasendrehung vorzunehmen. Die Möglichkeit der Drehung könnte sowohl dem Kunden, einem Vertreter des Herstellers als auch einer automatischen Schnittstelle zur Verfügung stehen.

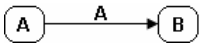
Um diese Möglichkeiten darzustellen, wird jeder Pfeil beschriftet. Folgende Beschriftungen sind möglich:



Dieser Pfeil bedeutet, dass nur der Vertreter des Softwareproduzenten die Möglichkeit hat von dem Zustand A eines TT auf den Zustand B eines TT zu wechseln.



Dieser Pfeil bedeutet, dass nur der Vertreter des Kunden die Möglichkeit hat von dem Zustand A eines TT auf den Zustand B eines TT zu wechseln.



Dieser Pfeil bedeutet, dass ein Automatismus den Zustand eines TT von dem Zustand A auf den Zustand B dreht.

Natürlich sind auch Kombinationen dieser Beschriftungen möglich, wenn mehreren Personenkreisen diese Drehungsmöglichkeit zur Verfügung steht.

Der Ein- bzw. Austritt aus einem der 5 Blöcke kann in 2 unterschiedlichen Formen dargestellt werden:

- Durch einen Schriftzug, der eine Aktion oder einen Zustand des Problems beschreibt.
- Durch eine Phase, die eigentlich schon dem nächsten Block zugeordnet wird und dort ebenfalls eingezeichnet ist.

Jeder Eintritt stellt aber auf jeden Fall eine Quelle und jeder Austritt eine Senke laut Graphentheorie dar.

6.2 Konzentrierte Freigabe

Die Forderungen an diesen Block des Modells kommen aus den Kapiteln 5.2.1 und 5.2.3 und lauten:

- Tickets müssen durch die Testabteilung des Kunden angelegt werden können, ohne dass diese bereits beim Hersteller einlangen.
- Das Ticket muss zuerst an eine weitere Stelle beim Kunden weitergeleitet werden, damit diese eine Art Vorprüfung durchführen kann.
- Der Kunde muss die Möglichkeit haben, das Ticket zurückzuziehen

Die Abbildung 7 zeigt den gesamten Block der konzentrierten Freigabe.

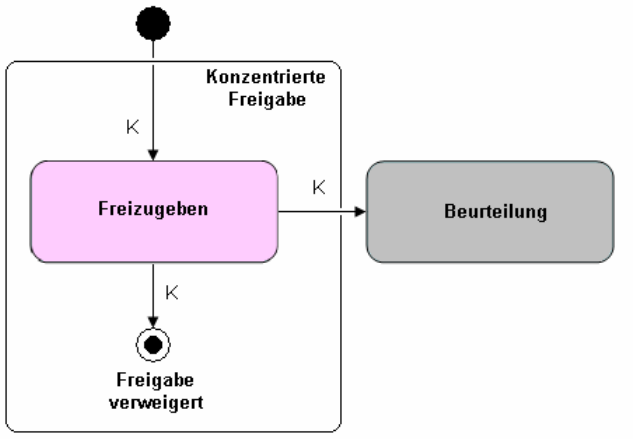


Abbildung 7: Phasenmodell der konzentrierten Freigabe

Die Besonderheit dieses Blocks ist die Phase „Freizugeben“. Probleme mit dieser Phase sind nur dem Kunden bekannt und landen auf der ToDoListe der Vorprüfungsstelle beim Kunden.

Diese Stelle hat nun 3 Möglichkeiten, dieses Problem zu behandeln.

Möglichkeit 1

Die Prüfung ergibt, dass das Problem aus der Sicht des Kunden tatsächlich besteht und deren Behebung erforderlich ist. Der gleiche Fall tritt ein, wenn ein Problem in dem Sinne gemeldet wird, dass ein Wunsch geäußert wird und die Vorprüfungsstelle diesen Wunsch unterstützt.

In diesen Situationen gibt die Stelle das Problem frei und dreht damit die Phase auf „Warten auf Stellungnahme“. Diese Phase hat auf der Seite des Herstellers die Bezeichnung „Stellungnahme abgeben“, wobei ich Details dieser Phase erst im Kapitel 6.3 behandeln werde.

Möglichkeit 2

Diese Möglichkeit ist eigentlich beinahe ident mit der ersten, nur hat die Vorprüfungsstelle natürlich auch die Möglichkeit Rücksprache mit der Testabteilung zu halten und gegebenenfalls einige Attribute des Problems zu verändern.

Attribute, die an dieser Stelle genannt werden müssen sind die gewünschte Lieferung, die Priorität des Problems oder die Unterscheidung zwischen Mangel und Erweiterungsauftrag. Es wäre denkbar, dass ein Tester ein Problem aufwirft, dies als Mangel meldet und die Vorprüfungsstelle erkennt, dass es sich aufgrund der Verträge eigentlich um einen Erweiterungswunsch handelt. Der letzte Schritt der Vorprüfungsstelle ist der gleich wie bei der Möglichkeit 1.

Möglichkeit 3

Die Vorprüfungsstelle ist anderer Meinung und spricht sich gegen eine Weiterleitung des Problems an den Softwarehersteller aus. In diesem Fall wird die Freigabe verweigert und das Problem erhält die Phase „Freigabe verweigert“. Diese Phase ist eine endgültige Phase oder um den Begriff aus der Graphentheorie zu verwenden, eine Senke.

Theoretisch würde auch die Möglichkeit bestehen, diese Phase gleich lautend zu der Phase „Zurückgezogen“ aus dem Kapitel 6.3 zu benennen. Diese Unterscheidung ist jedoch für Management-Auswertungen von großer Bedeutung, da es einen großen Unterschied macht, ob der Hersteller mit einem Problem konfrontiert wird und erst danach das Ticket zurückgezogen wird oder ob sich der Produzent nie mit diesem Problem beschäftigen musste.

6.3 Beurteilung

Die Forderungen an diesen Block des Modells kommen aus den Kapiteln 5.2.2 und 5.2.3 und wurden wie folgt definiert:

- Eine Unterscheidung zwischen Mangel oder Erweiterungsauftrag muss getroffen werden können.
- Es muss erkennbar sein, wer als nächstes eine Stellungnahme abzugeben hat.
- Es muss erkennbar sein, wie weit die Beurteilung vorangeschritten und ob sie bereits abgeschlossen ist.
- Der Kunde muss die Möglichkeit haben, das Ticket zurückzuziehen.
- Der TT-Worflow muss eine Möglichkeit vorsehen, dass die Meldung eines Problems zwar berechtigt war, von einer Umsetzung jedoch abgesehen wird.

Die Abbildung 8 zeigt den gesamten Block der Beurteilung eines Problems.

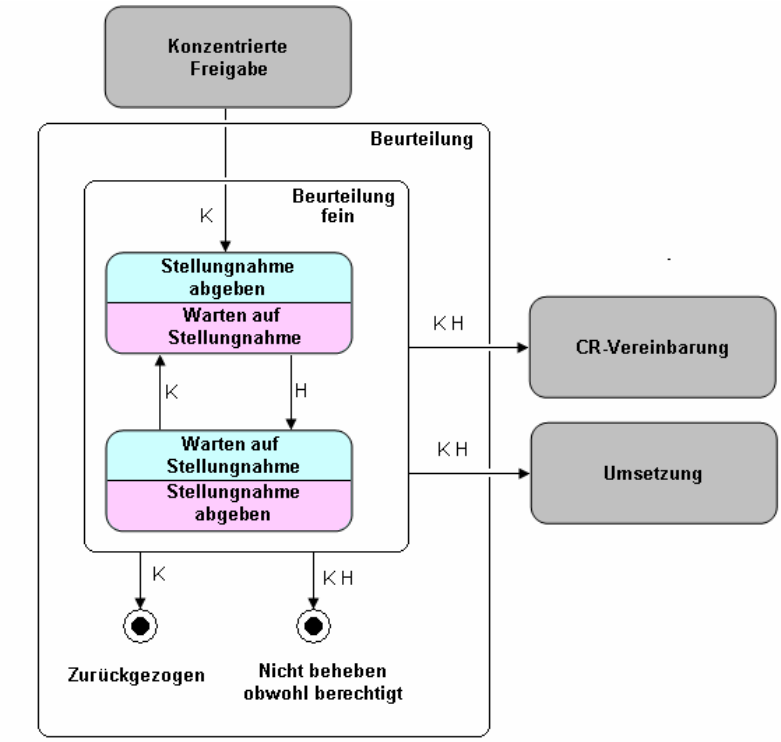


Abbildung 8: Phasenmodell der Beurteilung

Wie im vorigen Kapitel beschrieben, landet ein Problem nach der Freigabe durch die Vorprüfungsstelle des Kunden das erste Mal im Softwarehaus.

Hier eingetroffen, ist zumeist ein aufgrund des gemeldeten Problemgebiets automatisch ermittelter Bearbeiter an der Reihe eine Stellungnahme abzugeben.

Diese Stellungnahme kann grob gesehen in 4 verschiedene Richtungen laufen:

- Das Problem wird als unberechtigt angesehen.
- Das Problem wird voll inhaltlich akzeptiert.
- Das Problem wird als berechtigt angesehen, aber über manche Attribute herrscht noch Uneinigkeit.
- Das Problem wird zwar als berechtigt angesehen, eine Umsetzung aber dennoch abgelehnt.

Um diese Meinungen festzuhalten müssen während der Abstimmung entsprechende Attribute im TT vorgesehen werden. Diese Attribute gibt es in 3facher Ausführung, einerseits um die Meinung des Herstellers zu zeigen, andererseits um die Auffassung des Kunden wiederzuspiegeln und schlussendlich um anzuzeigen worüber bereits Einigkeit besteht. Die in dieser Situation benötigten Attribute möchte ich an dieser Stelle kurz erwähnen. Diese sind:

- Ein Attribut, das anzeigt, ob das Problem unberechtigt, berechtigt aber nicht behoben, ein Mangel oder ein Erweiterungswunsch ist.
- Ein Attribut, das den Liefertermin des Problems beinhaltet
- Ein Attribut, das die Priorität des Problems abdeckt.

Das Phasenmodell hat jedoch lediglich die Aufgabe, anzuzeigen wer gerade an der Reihe ist die nächste Stellungnahme abzugeben und wer auf eine Stellungnahme des Gegenübers wartet. Zu diesem Zweck gibt es die beiden Phasen „Stellungnahme abgeben“ und gegengleich dazu „Warten auf Stellungnahme“. Hat eine Seite eine Stellungnahme abgegeben, wird die Phase in den Warten-Zustand gedreht und der Gegenpart ist an der Reihe, da seine Phase nun auf „Stellungnahme abgeben“ gedreht wurde.

Die Möglichkeit der Phasendrehung in die jeweils andere Stellungnahme-Phase, besteht nur für denjenigen, der sich in der Phase „Stellungnahme abgeben“ befindet. Der wartende User kann ja ohnehin jederzeit eine weitere Stellungnahme abgeben und würde anschließend wieder auf den „Warten“ – Zustand.

Das Modell sieht 4 Möglichkeiten vor, den Wechsel dieser beiden Phasen zu beenden:

- Nur der Kunde hat die Möglichkeit das Problem zurückzuziehen, da nur der Melder eines Problems dieses auch zurückziehen kann. Ein weiterer Grund liegt in dem Umstand, dass der Kunde auch jederzeit ohne die Zustimmung des Herstellers ein Problem zurückziehen kann, wohingegen bei allen anderen Möglichkeiten unbedingt eine Übereinstimmung zustande kommen muss. Der Kunde dreht das Problem auf die Phase „Zurückgezogen“ und beendet damit das Leben des Problems, da diese Phase eine Endgültige ist. Diese Möglichkeit steht ihm auch in der Phase „Warten auf Stellungnahme“ zur Verfügung, da wie bereits erwähnt keine Zustimmung des Herstellers erforderlich ist.
- Beide Seiten können, wenn der jeweils andere zustimmt, ein Problem als Mangel akzeptieren und somit in die nächste Phase befördern. Zustimmung heißt in diesem Fall, dass das Attribut, welches anzeigt, ob das Problem unberechtigt, berechtigt aber nicht behoben, ein Mangel oder ein Erweiterungswunsch ist, auf der anderen Seite mit dem Vorgehen übereinstimmt. Das heißt, dass ein Kunde nur dann ein Problem als Mangel weiterdrehen kann, wenn in dem Attribut, das die Herstellermeinung beinhaltet, auch das Problem als Mangel angesehen wird.
- Die gleiche Möglichkeit besteht für beide Seiten, um nach der Zustimmung des anderen ein Problem als Erweiterungswunsch zu akzeptieren.
- Ebenfalls jede Seite hat die Möglichkeit, nach der Zustimmung des anderen, das Problem in den Zustand „Nicht beheben obwohl berechtigt“ zu bringen. Dieser Zustand stellt wiederum eine Senke dar und ist somit das endgültige Ende des Problems.

6.4 CR-Vereinbarung

Auch in diesem Kapitel möchte ich zunächst die Anforderungen, die in dem Kapitel 5.4 definiert wurden nochmals kurz zusammenfassen:

- Der Kunde muss die Möglichkeit haben, für ein bestimmtes TT ein Grobkonzept in Auftrag zu geben, nachdem ihm dafür ein Angebot übermittelt wurde.
- Wurde das Grobkonzept beauftragt, so muss anschließend der zur Erstellung zuständige Mitarbeiter des Herstellers benachrichtigt werden.
- Der Inhalt des Grobkonzepts muss abgestimmt werden.
- Das Grobkonzept muss abgenommen werden können.
- Dass Phasenmodell muss eine Aufwandsschätzung nach der Grobkonzeptannahme aufweisen.
- Ist die Schätzung fertig, muss als nächstes die Planung durchgeführt werden.
- Ein Angebot für den CR muss erstellt werden können.
- Der Kunde darf die einzelnen Arbeitsschritte „Aufwand schätzen“, „Planung„ und „Angebotserstellung“ nicht unterscheiden können.
- Ein Angebot muss gelegt werden können und die Möglichkeit dieses anzunehmen, abzulehnen oder nachzubessern muss vorhanden sein.

Die Abbildung 9 zeigt den Weg von dem Zeitpunkt der Vereinbarung, dass es sich um einen Erweiterungswunsch handelt bis zu der tatsächlichen Umsetzungsentscheidung des CRs.

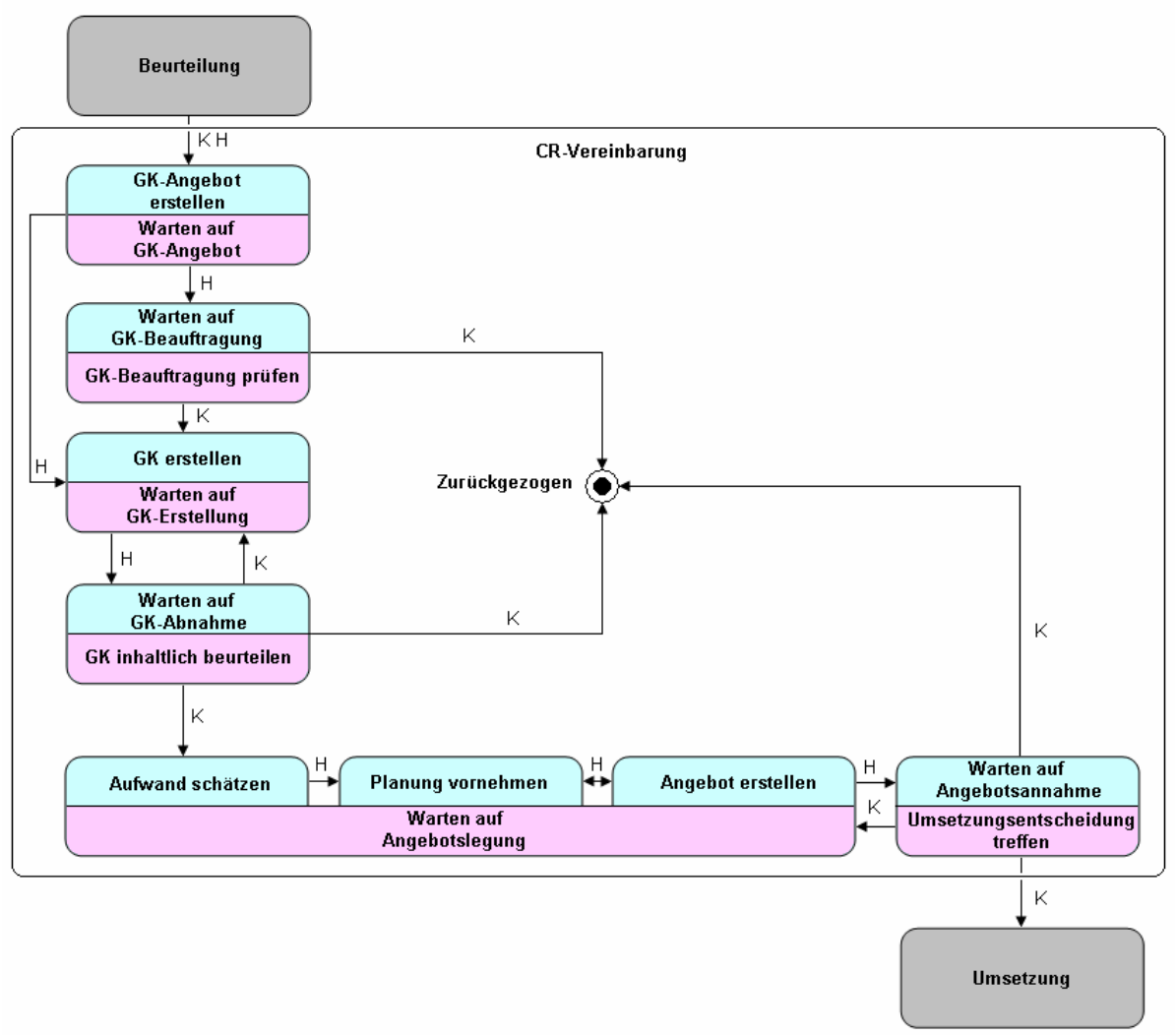


Abbildung 9: Phasenmodell der CR-Vereinbarung

Nach dem Abschluss der Beurteilung und der übereinstimmenden Meinung, dass es sich bei dem Problem um einen Erweiterungswunsch handelt, ist der Softwarehersteller an der Reihe, dem Kunden den Aufwand für die Erstellung eines Grobkonzeptes zu übermitteln. Dies geschieht in der Phase „GK-Angebot erstellen“. Zu diesem Zeitpunkt wartet der Kunde auf das GK-Angebot.

Wurde der Aufwand geschätzt, so wird durch den Phasenübergang auf „GK-Beauftragung prüfen“, der Kunde aufgefordert, entweder die Erstellung eines Grobkonzepts zu beauftragen oder, für den Fall dass ihm dieses zu teuer ist, das Problem zurückzuziehen.

Einen Sonderfall stellt die Situation dar, dass das GK keinen Aufwand verursacht, oder dem Kunden gratis angeboten werden soll. In diesem Fall muss das Grobkonzept nicht extra beauftrag werden. Aus diesem Grund wird ganz einfach die Phase „GK-Beauftragung prüfen“ übersprungen und somit sofort auf „Grobkonzept erstellen“ gedreht.

Wird das Problem zurückgezogen, so ist wie schon im letzten Kapitel angesprochen das Problem abgeschlossen. Im anderen Fall, landet das Ticket wieder auf den ToDoListen des Softwarehauses und hat dort die Phase „Grobkonzept erstellen“. Nachdem der Mitarbeiter das Konzept fertig gestellt hat, wird es in einem Attribut des TT abgespeichert und mit der Phasendrehung auf „GK inhaltlich beurteilen“ dem Kunden übermittelt.

Dieser hat nun 3 Möglichkeiten:

- Er kann das Problem zurückziehen, mit dem Resultat, dass der Aufwand des GK bezahlt werden muss. In diesem Fall dreht er das TT auf „Zurückgezogen“.
- Er kann das GK akzeptieren und ein Umsetzungsangebot anfordern. Zu diesem Zweck dreht er das TT auf „Warten auf Angebotslegung“ bzw. „Aufwand schätzen“ auf der Herstellerseite.
- Er kann Mängel im GK aufzeigen oder Änderungswünsche deponieren und damit das Problem wieder auf „Grobkonzept erstellen“ bzw. „Warten auf GK-Erstellung“ zurückdrehen. In diesem Fall geht die Prozedur wie bereits beschrieben weiter.

Akzeptiert der Kunde das GK, so laufen nun 3 Schritte auf der Seite des Herstellers ab, die der Kunde jedoch nur als einen einzigen wahrnimmt.

Durch die Anfrage nach einem Umsetzungsangebot, wird das Ticket in die Phase „Aufwand schätzen“ gedreht. Nach Abschluss der Schätzung, geht das TT in die Phase „Planung vornehmen“, um nach deren Abschluss ein Mitarbeiter des Produzenten durch die Phase „Angebot erstellen“ damit beauftrag wird, ein Angebot zu tippen und dieses dem Kunden zu übermitteln.

Mit diesem Schritt wird auch die Phase wieder auf beiden Seiten sichtbar weitergedreht und erhält die Bezeichnung „Umsetzungsentscheidung treffen“ bzw. „Warten auf Angebotsannahme“.

An dieser Stelle hat der Kunde das letzte Mal die Chance, die Umsetzung dieses Erweiterungswunsches zu stoppen, indem er das Ticket auf „Zurückgezogen“ dreht. Auch in diesem Fall wir er das GK bezahlen müssen.

Eine weitere Möglichkeit des Kunden, die in dieser Situation abgedeckt wird, ist dann anzuwenden, wenn der Kunde entweder mit dem Preis oder mit dem Termin nicht einverstanden ist. In diesem Fall kann das TT auf die Phase „Angebot erstellen“ zurückgedreht werden.

Genau in dieser Situation kann auch noch der Fall eintreten, dass die zuständigen Personen der Angebotserstellung eine neue Planung einfordern müssen, um zum Bespiel dem Wunsch nach einer früheren Lieferung nachzukommen. Für diesen Fall gibt es die Möglichkeit von „Angebot erstellen“ auf „Planung vornehmen“ zurückzugehen.

Ist der Kunde sowohl mit dem Preis als auch dem Liefertermin einverstanden, so unterschreibt er das Angebot und initiiert damit die Drehung der Phase auf „RfC ausarbeiten“ bzw. „Warten auf Lieferung“, die im nächsten Kapitel behandelt werden.

6.5 Umsetzung

Die Anforderungen, die diesen Block des Phasenmodells beeinflusst haben befinden sich in den Kapiteln 5.5 und 5.6 und konzentrieren sich auf folgende Punkte:

- Der Workflow muss in der Lage sein, nach der Umsetzungsentscheidung das Ticket an die Konzeptabteilung weiterzuleiten, um die RfCs auszuarbeiten.
- Nach Abschluss der RfC-Ausarbeitung, muss das Ticket an die Realisierer weitergeleitet werden können.
- Eine automatische Retourschnittstelle von CM-Tool ins PM-Tool ist erforderlich.
- Für den Tester müssen folgende Möglichkeiten zur Verfügung stehen:
    - Problem wurde erfolgreich umgesetzt
    - Problem wurde mangelhaft umgesetzt
    - Problem wurde nicht erfolgreich umgesetzt
- Probleme müssen auf eine Lieferung „Zusammenwarten“ können
- Alle Probleme, die auf eine spezielle Lieferung gewartet haben, müssen in einen Zustand gebracht werden, der dem Kunden sagt, dass diese Probleme nun nicht wieder auftreten sollten

Die Abbildung 10 zeigt das Phasenmodell für die Umsetzung eines Problems inklusive Test und Lieferung.

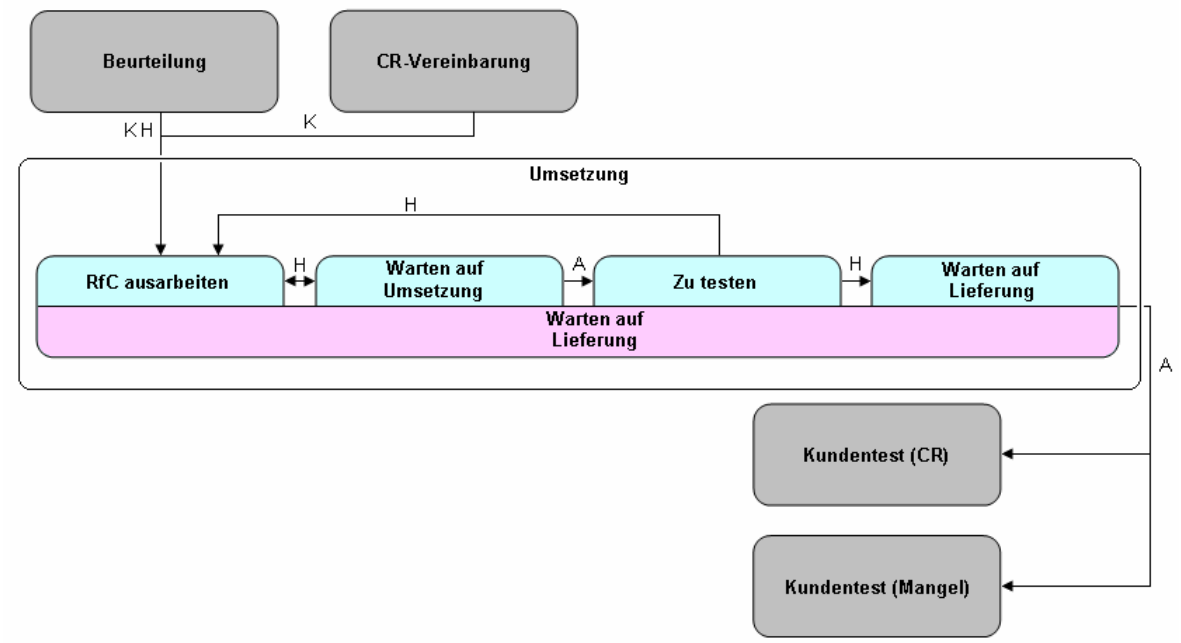


Abbildung 10: Phasenmodell Umsetzung

Nachdem das Angebot für die Umsetzung des CRs angenommen wurde oder der Kunde und der Hersteller zu der Meinung gelangt sind, dass dieses Problem als Mangel behoben werden soll, kann mit der Umsetzung des Problems begonnen werden.

Die Umsetzung eines Mangels oder eines CRs folgt dem gleichen Prozess. Am Beginn der Umsetzung müssen die RfCs ausgearbeitet werden. Die Ausarbeitung wird zumeist durch eine Konzept-Abteilung vorgenommen und endet damit, dass jeder RfC als Task in einem Change-Management Tool angelegt wird.

Sind alle Tasks angelegt, so dreht der Konzeptionist das TT auf „Warten auf Umsetzung“, da nun sämtliche weiteren Tätigkeiten im CM-System vorgenommen werden. Das CM kümmert sich nun um die Codierung der RfCs durch die Realisierungs-Abteilung, um den Compile der Software und die Einspielung der neuen Version in eine Testumgebung. Sollte die Realisierung während der Umsetzung bemerken, dass die RfCs nicht ausreichend sind um das Problem zu lösen, so gibt es die Möglichkeit, die Phase wieder auf „RfC ausarbeiten“ zurückzusetzen. Der Phasenübergang auf „Zu testen“ erfolgt automatisch durch eine Schnittstelle zwischen dem CM- und dem TTS.

Steht ein Problem auf „Zu testen“, so ist die interne Test-Abteilung an der Reihe um die erfolgreiche Umsetzung des Problems zu untersuchen. Die Tester haben dabei 3 Möglichkeiten:

- Sie können die erfolgreiche Umsetzung bestätigen und das TT in den Zustand „Warten auf Lieferung“ bringen.
- Sie können die mangelhafte, aber dennoch ausreichende Umsetzung bestätigen. Zu diesem Zweck bekommt der Mangel ebenfalls die Phase „Warten auf Lieferung“, allerdings muss ein weiteres Problem angelegt und mit dem getesteten Problem automatisch verknüpft werden.
- Sie können die erfolgreiche Umsetzung nicht bestätigen und schicken das Problem wieder an das Konzept zurück. Zu diesem Zweck wird die Phase auf „RfC ausarbeiten“ gedreht.

Wurde die erfolgreiche Umsetzung des Problems bestätigt, so ist das Problem bereit für die Lieferung. Da in der Regel nicht sofort jedes Problem dem Kunden übermittelt wird, wird es eine gewisse Zeit in diesem Zustand verharren, bis die nächste Lieferung an den Kunden geschickt wird.

Zum Zeitpunkt der Lieferung wird der nächste Automatismus gestartet, der nun alle Probleme untersucht, ob sie auf der einen Seite auf eine Lieferung warten, und auf der anderen Seite die Problembehebung auch in der gerade gelieferten Software enthalten ist, da bei dem Vertrieb von mehreren Produkten durchaus folgender Fall auftreten kann:

Produkt A wird geliefert, das Problem ist jedoch in Produkt B aufgetreten und wartet dennoch auf die Lieferung. Dieses Problem dürfte dann natürlich nicht geliefert werden.

Wartet das TT auf die Lieferung und ist die Umsetzung auch in der Lieferung enthalten, so dreht der Automat das TT in die Phase „Zu testen“, bzw. „Warten auf Kundentest“.

6.6 Kundentest beim Mangel

Sämtliche Anforderungen, die zu der Bestätigung der Behebung eines Mangels gesammelt wurden, finden sich im Kapitel 5.7.3. Zusammengefasst geht es um folgende Anforderungen:

- Die Bestätigung der erfolgreichen Umsetzung eines Problems muss möglich sein.
- Die Bestätigung der ausreichend erfolgreichen Umsetzung eines Problems ist erforderlich. Neue Probleme müssen mit einer Referenz zu diesem Problem erfasst werden können um das eigentliche Problem auf erledigt zu setzen.
- Die erneute Meldung des gleichen Problems, da dieses unverändert auftritt und nicht behoben wurde, muss gewährleistet werden. Dafür muss gelten:
    - Das Problem muss in die Beurteilung zurückgeschickt werden und die Verhandlungen beginnen von vorne.
    - Fachlich gesehen muss das Problem als neues Ticket angesehen werden, nur sind die Beschreibung und die bisher gesammelten Daten mitzunehmen.
    - Die Beurteilung des wiedereröffneten Mangels muss genauso abgelehnt oder zurückgezogen werden können wie ein neues Problem.
    - Der Rückzug eines solchen Problems darf nicht gleich bedeutend sein mit dem Rückzug des ursprünglichen Problems.

Die Abbildung 11 zeigt den Prozess der Bestätigung der Behebung eines Mangels durch den Kunden.

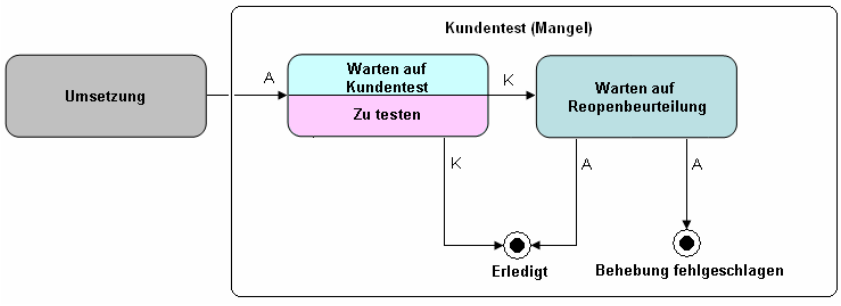


Abbildung 11: Phasenmodell Kundentest beim Mangel

Wurde in der Beurteilung des Problems beschlossen, dass es sich um einen Mangel handelt, so hat der zuständige Tester des Kunden 3 Möglichkeiten den Test zu beurteilen:

- Er kann die erfolgreiche Umsetzung bestätigen und das TT in den Zustand „Erledigt“ bringen. Damit ist das TT abgeschlossen.
- Er kann die mangelhafte, aber dennoch ausreichende Umsetzung bestätigen. Zu diesem Zweck bekommt der Mangel ebenfalls die endgültige Phase „Erledigt“, allerdings muss ein weiteres Problem angelegt und mit dem getesteten Problem automatisch verknüpft werden.
- Er kann aber auch das unveränderte Auftreten des Mangels feststellen und damit das TT in die Phase „Warten auf Reopenbeurteilung“ bringen. In dieser Situation muss durch das TTS ein neues Problem angelegt werden, das die gleiche Problembeschreibung beinhaltet. Dieses Problem durchläuft die gleichen Schritte wie jedes neu angelegte Problem, nur haben verschiedene Phasendrehungen des neuen Problems Auswirkungen auf das ursprüngliche Problem:
    - Wird die Freigabe verweigert, so wird das ursprüngliche Problem automatisch auf „Erledigt“ gestellt.
    - Wird das Problem zurückgezogen, so wird das ursprüngliche Problem ebenfalls automatisch auf „Erledigt“ gedreht.
    - Wird eine Umsetzung vereinbart so erhält das ursprüngliche Problem die Phase „Behebung fehlgeschlagen“.
    - Wird beschlossen, dass das Problem auf „Nicht beheben obwohl berechtigt“ gesetzt wird, so erhält das ursprüngliche Problem ebenfalls die Phase „Behebung fehlgeschlagen“.

6.7 Kundentest beim Erweiterungswunsch

Die Forderungen an diesen Prozess stammen aus dem Kapitel 5.7.2 und enthielten zusammengefasst folgende Punkte:

- Die Abnahme eines CRs muss möglich sein
- Es muss möglich sein, neue Probleme mit einem Bezug zu dem CR melden zu können und gleichzeitig zu vermerken, ob dieses Problem den Einsatz der neuen Funktionalität gefährdet.
- Automatische Abnahme von CRs wenn in einer gewissen Zeitspanne keine Probleme gemeldet werden.
- Die automatische Abnahme muss nach Behebung von Problemen, die die Abnahme verhindert haben, wieder einsetzen.

Die Abbildung 12 zeigt den Prozess der Abnahme eines Erweiterungsantrages.

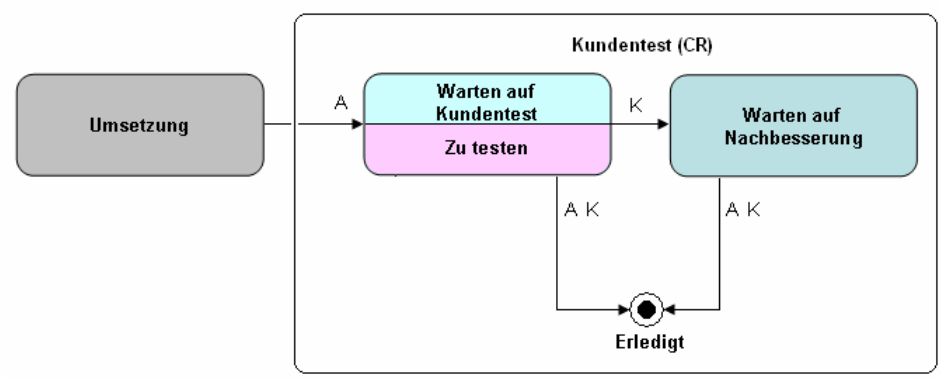


Abbildung 12: Phasenmodell Kundentest beim Erweiterungswunsch

Wurde in der Beurteilung des Problems beschlossen, dass es sich um einen Erweiterungswunsch handelt, so hat der zuständige Tester des Kunden 3 Möglichkeiten den Test zu beurteilen:

- Er kann die erfolgreiche Umsetzung des CRs bestätigen und das TT in den Zustand „Erledigt“ bringen. Damit ist das TT abgeschlossen und es erfolgt zumeist die letzte Zahlung des vereinbarten Preises des Erweiterungswunsches.

- Er kann die mangelhafte Umsetzung des Erweiterungswunsches feststellen. Zu diesem Zweck wird für jedes mit diesem CR in einem Zusammenhang stehende Problem ein TT angelegt, das automatisch mit dem CR verknüpft wird. Für jedes TT muss der Tester nun angeben, ob es so schwerwiegend ist, dass die Umsetzung des Problems eine Voraussetzung für die Abnahme des CRs darstellt. Ist der Test abgeschlossen, so wird das TT wie folgt gedreht:
    - Findet das System noch Tickets, die eine Voraussetzung für die Abnahme darstellen, und sind diese Tickets nicht in der Phase „Zurückgezogen“, „Freigabe verweigert“, „Nicht beheben obwohl berechtigt“, „Behebung fehlgeschlagen“ oder „Erledigt“, so wird dem TT die Phase „Warten auf Nachbesserung“ zugewiesen.
    - In allen anderen Fällen wird das TT in den Zustand „Erledigt“ gedreht.

Befindet sich das TT in der Phase „Warten auf Nachbesserung“, so gibt es 2 mögliche Wege, wie das Ticket in den Zustand „Erledigt“ wechseln kann:

- Der Kunde sagt bewusst, ich nehme den CR nun trotz offener Mängel ab.
- Alle TT, die als Voraussetzung für die Abnahme markiert wurden, befinden sich in einem endgültigen Zustand, sind also sowohl für den Kunden als auch den Hersteller abgeschlossen.

Schlussendlich trägt der Lebenszyklus auch noch der Forderung nach einer automatischen Abnahme nach Zeitablauf Rechnung. Wird innerhalb einer einstellbaren Zeit kein Problem seitens des Kunden gemeldet, dass Voraussetzung für die Abnahme ist, und befindet sich das Ticket in der Phase „Warten auf Kundentest“ bzw. „Zu testen“, so stellt der Automat das TT automatisch auf „Erledigt“.

7 Schlussbemerkungen

Wie diese Arbeit zeigt, beschäftigt sich die Literatur mit vielen Aspekten des Problemmanagements. Angefangen von der Einordnung in die ITIL über die dem PM zugeordneten Themenbereiche bis hin zur Aufteilung in die Funktionsblöcke des PM. Auch über das Thema Trouble-Ticket-System kann man in der Literatur einiges finden, dennoch gab es so gut wie kein Werk, das sich mit der Gestaltung eines Workflows für Trouble-Tickets beschäftigt hat.

Ich hoffe mit dieser Arbeit diese Lücke verkleinert zu haben, und auf der anderen Seite einen Grundlage für Manager geschaffen zu haben, die ein TTS einführen wollen oder ihr bestehendes System überholen möchten.

Der ausgearbeitete Zyklus und auch die zuvor definierten Anforderungen zeigen sehr deutlich wie komplex diese Materie ist, aber auch welche Chancen in der richtigen Gestaltung des Lebenszyklus stecken können, denn kaum ein anderer Prozess kann in der Softwareentwicklung mit einem Tool so stark beeinflusst werden.

An dieser Stelle sollen nun nochmals die wichtigsten Erkenntnisse zusammengefasst werden:

- Durch den Umstand, dass der Prozess des PM durch ein TTS sehr stark beeinflusst werden kann, kann man auch ableiten, wie wichtig es für Ticket-Systeme ist, dass der Workflow dynamische Anpassungsmöglichkeiten an die spezifischen Merkmale eines Softwareunternehmens bietet. Doch diese gewonnene Flexibilität muss man auch ausnützen können und das funktioniert nur, wenn man Lösungsansätze anderer heranziehen kann.
- Unumstritten in der Literatur ist der Umstand, dass sämtliche Aktionen und Maßnahmen im Zusammenhang mit einem Problem auch auf dem jeweiligen TT dokumentiert werden müssen [Olbr2003, 36ff]. Dies liefert auch den Grund dafür, dass jeder Arbeitsschritt durch eine eigene Phase im Lebenszyklus eines TT abgebildet werden soll. Die Vorteile dieses Prinzips liegen in der besseren Verfolgbarkeit von Tickets und der Möglichkeit nach dem Abschluss eines Arbeitsschrittes das TT automatischen an die betreffenden Abteilungen weiterzuleiten.

- Einen zentralen Punkt, der sich ebenfalls durch den gesamten ausgearbeiteten Lifecycle zieht, stellt das Prinzip der 2 Phasenmodelle dar. Das TTS muss einerseits die Möglichkeit vorsehen einer Phase 2 Bezeichnungen zu geben. Eine Bezeichnung dient der Anzeige für den Kunden und die andere der Anzeige für den Vertreter des Produzenten. Auf der anderen Seite muss es auch möglich sein, dass mehrere Phasen nur auf einer Seite zu einer einzigen zusammengefasst werden können, damit zum Beispiel die Umsetzung wie ein einziger Arbeitsschritt auf den Kunden wirkt obwohl dies für den Hersteller mehrere Arbeitsschritte darstellt.
- Besonders wichtig für jedes TTS ist die Unterscheidung zwischen Mangel und Erweiterungsantrag. Beide stellen ein Problem dar und werden als TT gemeldet doch muss während der Beurteilung des Tickets eine Entscheidung darüber getroffen werden, ob das Problem einen Mangel oder einen Change-Request darstellt. Die Unterscheidung in Mangel und CR hat nicht nur wirtschaftliche Gründe, sie erleichtert auch die Bewertung der Qualität des Produktes oder erhöht die Produktstabilität.
- Die Unterscheidung zwischen Mangel und Erweiterungsauftrag bestimmt auch wesentlich die Behandlung des TT. Sie ist im Bezug auf das Phasenmodell besonders in 2 Situationen von Bedeutung:
    - Mängel werden weniger detailliert abgestimmt. Sie gehen sofort nachdem eine gemeinsame Sicht auf des Problem besteht in die Umsetzung. CRs müssen andererseits eine weitaus längere Abstimmung durchlaufen. Sie müssen inhaltlich genauer definiert werden, der Aufwand für die Umsetzung muss geschätzt werden und schlussendlich muss der CR zu einem Umsetzungspreis angeboten werden.
    - Der Kundentest kann nur dann ein Problem wiedereröffnen, wenn es als Mangel akzeptiert wurde. Wurde das Problem als CR vereinbart, so hat der Kunde nur die Möglichkeit Mängel zu diesem CR zu melden, er kann aber nicht das gesamte Problem wieder in die Beurteilung zurückschicken.

- Eine sichere Schnittstelle zum Change-Management ist unerlässlich. Speziell die Nachricht, dass nun sämtliche RfCs im CM-Tool umgesetzt wurden und das Problem dadurch zum Nachtesten bereit steht ist von besonderer Bedeutung, da sonst bei zu frühen Tests unnötige Ressourcen verschwendet werden und bei zu späten Tests Problembehebungen geliefert werden, ohne sie dem Kunden bekannt zu geben bzw. nachgetestet worden zu sein.

Es liegt auf der Hand, dass diese Arbeit nur einen Denkanstoss oder einen Vorschlag für einen Workflow anbietet, der an der einen oder anderen Ecke noch an das jeweilige Unternehmen angepasst werden kann oder vielleicht auch muss.

Dennoch deckt diese Lösung sämtliche Probleme in der Workflowgestaltung ab, die sich einem Softwarehaus, wie es im Kapitel 5.1 definiert wurde, in den Weg stellen können.

8 Abkürzungsverzeichnis

BTS	Bug Tracking Systeme
CM	Change-Management
CR	Erweiterungsantrag bzw. Change-Request
FRS	Feature Request Systeme
GK	Grobkonzept
IM	Incident-Management
ITIL	Information Technology Infrastructure Library
IT-SM	Information Technology Service-Management
KDB	Knowledge Data Base Systeme
PM	Problem-Management
PTS	Problem Tracking Systeme
RfC	Request for Change bzw. Änderungsantrag
STS	Support Tracking System
TT	Trouble-Ticket
TTM	Trouble-Ticket-Management
TTS	Trouble-Ticket-System

9 Literaturverzeichnis

[Bern2001]

Bernhard, Martin G.; Lewandowski, Winfried; Mann, Hartmut: Service-Level-Management in der IT. Symposion Publishing, Düsseldorf 2001

[Bern2003]

Bernhard, Martin G.; Lewandowski, Winfried; Mann, Hartmut; Schrey, Joachim: Praxishandbuch Service-Level-Management. Symposion Publishing, Düsseldorf 2003

[Blaz2000]

Blazey, Martina: Studienarbeit Einflussfaktoren des Einsatzes von Standard bzw. Individualsoftware in Bezug auf die Komponenten der Softwareentwicklung im Software Lebenszyklus. http://www.martina.blazey.org/pdf/studienarbeit.pdf, Mai 2000, Abruf am 2005-03-28

[BoKP2002]

Van Bon, Jan; Kemmerling Georges; Pondman Dick: IT Service Management, eine Einführung. Van Haren Publishing, 2002.

[ChMa2003]

FITS Change Management. http://www.becta.org.uk/tsas/docs/fits_change.pdf, 2003-09-01, Abruf am 2004-07-15

[DSDP2001]

DSD Problemmangement. http://www.sbs.at/pages/dsd_problem.htm , 2001-11-12, Abruf am 2005-04-27

[Eato2002]

Eaton, Dave: Problem Management Tools Summary.

http://www.daveeaton.com/scm/PMTools.html , 2002, Abruf am 2005-04-10

[Elsä2005]

Elsässer, Wolfgang: ITIL einführen und umsetzen. Leitfaden für effizientes IT-Management durch Prozessorientierung. Hanser-Fachbuchverlag 2005.

[Erns99]

Ernst, Jochen: Einsatz von Managementwerkzeugen für eine qualitätsgesicherte Störungsbearbeitung. Diplomarbeit UNI-Karlsruhe, 1999

[Grun2003]

Grunwitz. Kai; Schürmann, Thomas: IT Service Level Management. www.cecmg.de/doc/slm2003_roundtable_vortrag.pdf , 2003 Abruf am 2005-03-24

[Güri91]

Gürich, Wolfgang; Homberg, Willi; Peters, Hartmut: Rechnergestütztes Problem-Management: Aufgaben und Funktionen in einer heterogenen DV-Umgebung

In: G. Schwichtenberg (Hrsg.), Organisation und Betrieb von Informationssystemen. Informatik-Fachberichte 279, Springer-Verlag, Dortmund 1991.

[Hage2004]

Hagelberg, Joachim: Eine kleine Einführung in SCM. http://home.t-online.de/home/Jo.Hagelberg/rcs-d.htm , 2004, Abruf 2005-05-04

[HaTS2004]

Hasitschka, Martin; Teichmann, Maria-Therese; Sneed, Harry M.: Software-Produktmanagement Wartung und Weiterentwicklung bestehender Anwendungssysteme. Dpunkt-Verlag, Wien 2004.

[ITIL2003]

ITIL – De – Facto – Standard für Service Management. http://www.hdi-europe.de/content/tools/images/IET%20Solutions-ITSM%20Whitepaper.pdf, 09/2003, Abruf am: 2005-03-24

[John92]

Johnson, D.: NOC Internal Integrated Trouble Ticket System Functional Specification Wishlist. http://velociraptor.mni.fh-giessen.de/rfc/rfc1297.txt 1992,

Abruf am: 2005-04-07

[Krus2001]

Kruse, Rainer: Entwicklung eines Werkzeuges für die Administration eines Trouble Ticket Systems. Fernuniversität Hagen; 2001

[Kuhl2003]

Kuhlig, Robert: Vortrag zu Service Management nach ITIL. http://www.nm.ifi.lmu.de/Hauptseminare/ws0304/mitsm.pdf 2003, Abruf am 2005-03-27

[Olbr2004]

Olbrich, Alfred: ITIL – kompakt und verständlich. Effizientes IT-Service-Management. Den Standard für IT-Prozesse kennenlernen, verstehen und erfolgreich in der Praxis umsetzen. Vieweg-Verlag 2004 (2., verbesserte Auflage).

[Pink2001]

Pink, Mario: Ausarbeitung Seminar Projektmanagement. http://www.informatik.tu-cottbus.de/~rust/ss2001/swpm/Pink.pdf 2001, Abruf 2005-04-30

[Prob95]

Probst, Erwin: Entwurf einer Trouble-Ticket-Struktur als Grundlage zur Durchführung des Problemmanagements eines verteilten Systems durch einen externen Service-Anbieter. Diplomarbeit an der TU München, 1995

[PrMa2003]

FITS Problem Management. http://www.becta.org.uk/tsas/docs/fits_problem.pdf, 2003-09-01, Abruf am 2004-07-15

[Scha2000]

Schauer, Martin: Spezifikation einer Service-Management-Architektur für einen qualitätsgesicherten IT-Betrieb. Uni Karlsruhe; 2000

[Schu96]

Schulze, Hans: Problem-Management Chancen nutzen – Ziele erreichen. Humboldt, München 1996

[Stei2004]

Steinweg, Carl: Management der Software Entwicklung. Vieweg-Verlag, Wiesbaden 2004

[Trum2004]

Trummer, Christian: Adaptierbare Webapplikationen - Adaptierbarkeit der Hyperwave Information Server Applikation "Support Tracking System". Diplomarbeit TU-Graz, 2004

[UML]

Zustandsdiagramm. http://ivs.cs.uni-magdeburg.de/~dumke/UML/21.htm , Abruf am 2005-06-16

In: *Prof. Dr.-Ing. habil. Dumke, Reiner R.:* UML-Tutorial. http://ivs.cs.uni-magdeburg.de/~dumke/UML/inhalt.htm, Abruf am 2005-06-15

[Valt93]

Valta, Robert; Apostolescu, Victor; Dreo, Gabi; de Jager, Riaan: Unterstützung der Fehlerdiagnose durch ein Trouble - Ticket - System: Anforderungen, Design und Einsatzerfahrungen

In: H. P. Löw und G. Partosch (Hrsg.), Verteilte Systeme - Organisation und Betrieb. Fachgespräch über Rechenzentren, Deutscher Universitäts-Verlag GmbH, Wiesbaden 1993.

[Walk2001]

Walker, Gary; Kern, Harris: It Problem Management (Harris Kern's Enterprise Computing Institute). Prentice Hall PTR, 2001.

[Wald97]

Walder, Franz Peter; Pazak, Gerald: Qualitätsmanagement und Projektmanagement. Vieweg-Verlag, Wien 1997

[Zveg87]

Zvegintzov, N.: Immortal Software, Datamation, 1987