



Testmetriken - Ein Erfahrungsbericht

18. STEV-Österreich-Fachtagung,
Wien, 23. Mai 2003

Version: 1.0

Datum: 01.05.2003

Dipl.-Ing. Manfred Baumgartner manfred.baumgartner@anecon.com



Software Design und Beratung G.m.b.H.

Alser Straße 4 / Hof 1

A-1090 Wien

Inhaltsverzeichnis

1	Abstract.....	2
2	Metriken – wozu?	2
3	Bereiche für Verbesserungen.....	4
4	Konkrete Metriken aus der Praxis	5
4.1	Voraussetzungen.....	6
4.2	Testprozess – Verifikation des Testdesigns.....	6
4.3	Projekt und Test Performance – Der Fehlermanagementprozess	9
4.4	Testplanung und Controlling – Laufende Anpassung	10
4.5	Produktqualität – Die Quality-Line	11
5	Risiken beim Einsatz von Metriken	13
6	Abschluss.....	15

1 Abstract

Metriken im Zusammenhang mit dem Test von Software gibt es unzählige. Sinn machen aber nur diejenigen, die auch tatsächlich in operative Maßnahmen zur Qualitätssteigerung des Produktes und zur Produktivitätsverbesserung der Test- und Entwicklungsprozesse umgesetzt werden können. Anhand konkreter Beispiele aus der Praxis werden mögliche Effekte durch den konsequenten Einsatz von Kennzahlen dargestellt und auch die damit verbundenen Risiken diskutiert.

2 Metriken – wozu?

Allgemein dienen Metriken dazu, um etwas messen zu können. Messungen ihrerseits sind die objektive Feststellung von Objekteigenschaften bezogen auf die jeweilige Metrik, wie etwa Länge, Gewicht, Anzahl o.ä. Bei Durchführung einer Messung bleibt es aber in den seltensten Fällen bei der reinen Erhebung eines Wertes. Fast immer ist damit eine relative Bewertung des Ergebnisses gegen einen

vorab definierten Erwartungswert verbunden. Dieser richtet sich wiederum nach der entsprechenden Zielsetzung der durchgeführten Messung.

Im täglichen Leben messen und bewerten wir ständig, immer verknüpft mit konkreten Absichten unterschiedlicher Art, auch wenn wir uns dessen nicht immer bewusst sind. Der Blick auf den Tachometer im Ortsgebiet dient der **Steuerung** und Anpassung an die erlaubte Geschwindigkeit. Der Tachometer am Prüfstand zeigt die maximale **Leistungsfähigkeit** des Autos unter gewissen Rahmenbedingungen. Der Kilometerzähler weist auf die **Erreichung** einer definierten Wegstrecke hin. Die Tankanzeige ermöglicht den **Vergleich** von Verbrauchsdaten bei unterschiedlichen Motoreinstellungen und liefert die **Bestätigung** für die geplanten Werte. Wird der Druck auf das Gaspedal verstärkt, die Motoreinstellung geändert, das Benzin-Luft-Gemisch angepasst, zeigen die nächsten Messungen, ob die erwarteten **Veränderungen** eingetreten sind oder nicht.

Welche Metriken, Messungen, Zielsetzungen sind nun im Software-Entwicklungsprozess, insbesondere im Testprozess zweckmäßig? Schon die wenigen Beispiele im vorhergehenden Absatz zeigen auf, wie viel an Phantasie im Bereich des Messens entwickelt werden kann. Es gibt wahrscheinlich unzählige mögliche Messungen, Statistiken, Vergleiche und Trendanalysen – jede für sich sinnvoll interpretierbar. Leicht kann es passieren, dass die Schaffung von Metriken als stand-alone Tätigkeit zum reinen Selbstzweck führt und außer einer Vielzahl an statistischem Material keine Ergebnisse liefert. Metriken sind daher stets in Maßnahmen zur laufenden Verbesserung des Prozesses und des Produktes einzubinden.

Die Frage ist also nicht: „Was kann man alles messen?“, sondern „Was will man verbessern und welche Messungen unterstützen dabei?“. Nicht die Anzahl der Metriken, sondern deren direkte Umsetzbarkeit in operative Verbesserungsmaßnahmen ist entscheidend.

Entscheidend ist jedoch, dass für jede Zielsetzung oder Verbesserungsmaßnahme der zu erreichende „Zielwert“ genau definiert ist, und darüber auch ein gemeinsames Verständnis der handelnden Gruppen und Personen herrscht. Ist dies nicht gegeben, wird sich der erhoffte Nutzen nicht einstellen.

3 Bereiche für Verbesserungen

Testen ist nur ein Aspekt im Software-Lebenszyklus. Aber er ist mindestens gleichbedeutend mit all den anderen, wie Projektmanagement, Konzeption, Design oder Entwicklung. Daher ist auch die eigenständige Betrachtung des Tests hinsichtlich der unterschiedlichen Dimensionen für Verbesserungspotentiale nicht nur zulässig sondern notwendig. Für die folgenden Bereiche der Software-Entwicklung wurden daher Ansätze für Verbesserungen der Testprozesse identifiziert und diese in weiterer Folge durch die Definition von Metriken unterstützt.

- **Der Software-Entwicklungs-Prozess.** Verbesserungen der Entwicklungsprozesse können auf der Ebene der Organisation oder innerhalb spezifischer Projekte geschehen. Zwar ist meist jedes Entwicklungsvorhaben nicht direkt mit anderen oder vorhergehenden vergleichbar, dennoch gibt es oft ähnliche Ursachen dafür, warum Projekte erfolgreich sind oder scheitern.

Der Testprozess, als eigenständiger Prozess innerhalb des Software-Entwicklungs-Prozesses, ist in hohem Maße am Erfolg oder Misserfolg des Produkteinsatzes beteiligt. Da aber in den meisten Projekten gerade das Testen sowohl von der Planung als auch von dessen Bedeutung her immer noch unterschätzt wird, ist es von besonderer Wichtigkeit, den Testprozess, d.h. die Entwicklung der Testdesigns, die Ableitung von Testfällen und Testdaten sowie die Dokumentation und Verfolgung von Fehlern, optimal zu gestalten.

- **Projekt Performance.** Innerhalb einzelner Entwicklungszyklen sind viele Ereignisse und Ergebnistypen sowohl häufig als auch wiederholt, wie zum Beispiel der Änderungsprozess, Codierung, Reviews, Build-Prozess etc. Gerade diese bieten gute Möglichkeiten zu Prozessverbesserungen auf Basis von Messungen mit einem sehr unmittelbaren Nutzen für das laufende Projekt.

Für die Test-Performance sind insbesondere der Fehlermanagement-Prozess sowie die effektive und effiziente Vorbereitung und Durchführung der Tests relevant.

- **Projektplanung und -controlling.** Ohne verlässliche quantitative Informationen über das Projekt sind Planung und Controlling nicht vernünftig machbar. Metriken, die sowohl Aufwände und Ergebnisse darstellen, erlauben die Bewertung der aktuellen Projektpformance und eine Abschätzung für den zukünftigen Projektverlauf.

Eine spezifische Testplanung und konsequentes Controlling erlauben die Beurteilung des Teststatus und eine Anpassung der Testpläne anhand durchgeführter Tätigkeiten und erzielter Ergebnisse. Dies ist insofern von ganz besonderer Bedeutung, da sich eine wesentliche Planungsannahme für den Test, nämlich die zugrundeliegende Qualität des entwickelten Produktes, erst im Laufe der Zeit konkretisiert und auch weiterhin laufend verändert.

- **Produktqualität.** Ziel jedes Software-Projektes ist es, ein qualitativ hochwertiges Produkt in Einsatz zu bringen und diese Qualität auch über die Zeit hinweg beizubehalten. Messungen können nun dazu dienen, neben der Qualität der Prozesse auch die Qualität des Produktes oder die Zufriedenheit der Anwender mit dem Produkt zu bewerten.

Für den Test sind hier insbesondere klassifizierte Fehlerraten während des Projektes aber auch nach Produkteinsatz von großer Wichtigkeit. Dabei werden sowohl absolute Fehlerzahlen als auch Fehlerrends bewertet. In der Gewissheit, dass es keine fehlerfreie Software gibt, sind hier Qualitätsziele festzulegen, die dennoch ein geplantes Testende definieren.

4 Konkrete Metriken aus der Praxis

Die im Folgenden dargestellten Metriken wurden von meinen KollegInnen und mir nun bereits in mehreren Projekten eingesetzt, um sowohl die Prozesse als auch die entwickelten Produkte in ihrer Effizienz und Qualität zu verbessern. Dies gelang einerseits durch die relative Einfachheit der Metriken und andererseits durch die bewusst herbeigeführte Transparenz dieser Informationen im Projekt.

4.1 Voraussetzungen

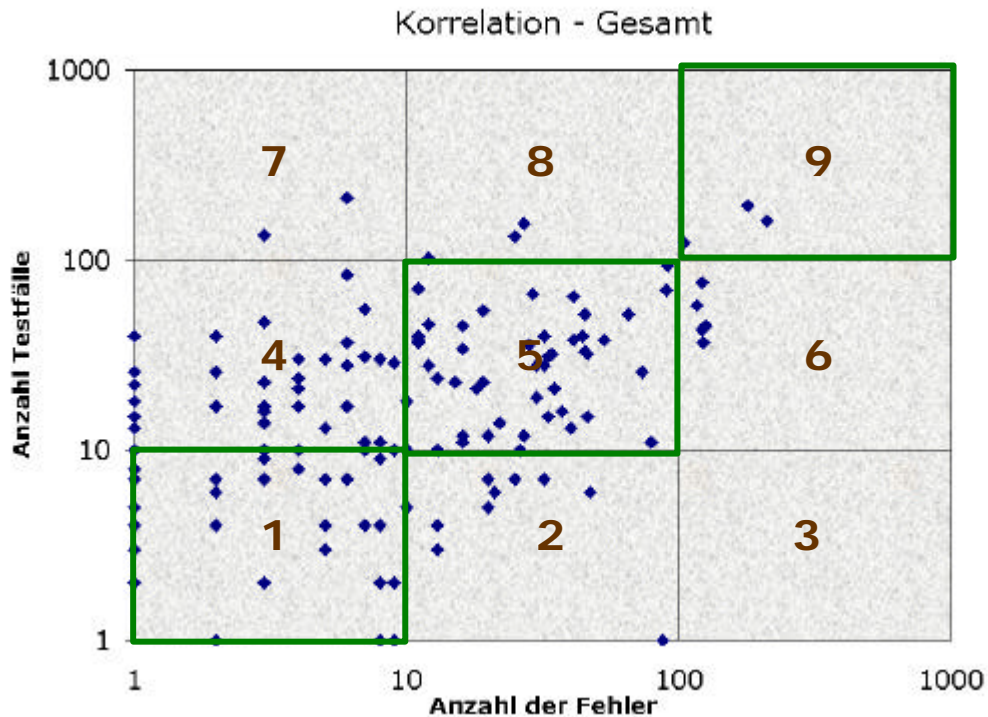
Die angeführten Metriken aus der Praxis beziehen sich auf Informationen wie Geschäftsprozesse, Testfälle, Fehler oder Aufgaben und Aufwände. Es war also auch eine instrumentelle Basis zu schaffen, um die entsprechenden Auswertungen erstellen und darstellen zu können.

- **Geschäftsprozesse, Use-Cases, Business Cases.** Das Testdesign der Projekte, aus denen die Metrik-Beispiele stammen, war sehr stark geschäftsprozessorientiert. Basis für die Planung und Design der Tests war eine Liste der implementierten Geschäftsprozesse, die ihrerseits einer Komplexitätsbewertung aus Geschäftstransaktionssicht unterzogen wurden. Diese Bewertung diente zur groben Abschätzung der zu erwartenden Testfallanzahl je Geschäftsprozess bzw. Geschäftsfall.
- **Testfälle.** Ergebnis eines strukturierten Testdesigns war jeweils ein Testfallkatalog mit wohldokumentierten und nachvollziehbaren Testfällen, die einerseits sowohl den einzelnen Geschäftsprozessen zugeordnet waren und andererseits über das Fehlermanagementsystem den jeweils gefundenen Fehlern.
- **Fehler.** Der Einsatz eines Fehlermanagementsystems war für einen geordneten Test- und Entwicklungsprozess unerlässlich. Basierend auf den Zuordnungen der Fehler zu Testfällen und Geschäftsprozessen waren Rückschlüsse auf die Effizienz des Testdesigns möglich. Ein definierter Workflow erlaubte die Betrachtung der Projektperformance. Fehlerklassen, Fehleranzahlen und Fehlertrends beleuchteten die aktuelle Qualität des Softwareproduktes.

4.2 Testprozess – Verifikation des Testdesigns

Ausgangslage: Für jeden Geschäftsprozess werden im Rahmen des Testdesigns eine Komplexitätsabschätzung vorgenommen. Darauf basierend wird eine bestimmte Anzahl an zu erstellenden Testfällen geplant und eine zu erwartende Fehleranzahl abgeleitet.

Zielsetzung: Sicherstellung, dass das Testdesign entsprechend der Planung quantitativ (Anzahl der Testfälle) und qualitativ (Anzahl der gefundenen Fehler) umgesetzt ist. Die getroffenen Annahmen werden überprüft und gegebenenfalls angepasst.



Interpretation: Jeder Punkt in der Grafik repräsentiert einen zu testenden Geschäftsprozess, oder allgemein, ein Testobjekt entsprechend dem High-Level Testdesign. Seine Position entspricht den bisher für diesen Prozess gefundenen Fehlern (x-Achse) und den dafür definierten Testfällen (y-Achse). Die Skalierung und Festlegung der Zielfelder leitet sich aus den Komplexitätsannahmen und erwarteten Fehlerraten ab. Im konkreten Beispiel bedeutet das:

Komplexität	Testfallanzahl	Fehleranzahl*	Zielfeld
Nieder	1-10	1-10	1
Mittel	11-100	11-100	5
Hoch	>100	>100	9

*Fehlerrate: 1,0

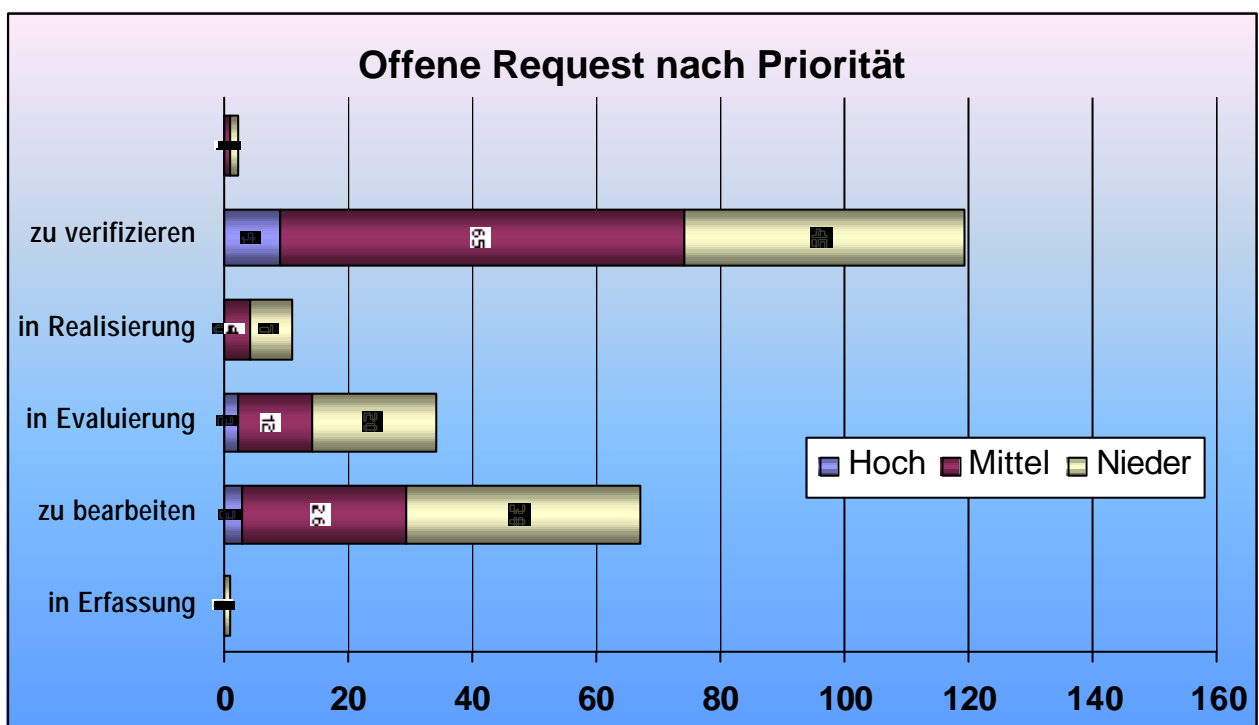
Interpretationen, abhängig von der Positionierung im Zielfeld:

Zielfeld	Nieder	Mittel	Hoch
1	•OK	<ul style="list-style-type: none"> • Zu wenige Testfälle definiert • Testfälle zu allgemein • Überschätzte Komplexität 	<ul style="list-style-type: none"> • Zu wenige Testfälle definiert • Testfälle zu allgemein • Überschätzte Komplexität
2	<ul style="list-style-type: none"> • Fehlerrate zu nieder angesetzt • Unterschätzte Komplexität 	<ul style="list-style-type: none"> • Zu wenige Testfälle definiert • Testfälle zu allgemein • Fehlerrate zu nieder angesetzt 	<ul style="list-style-type: none"> • Zu wenige Testfälle definiert • Testfälle zu allgemein • Fehlerrate zu nieder angesetzt • Überschätzte Komplexität
3	<ul style="list-style-type: none"> • Fehlerrate zu nieder angesetzt • Unterschätzte Komplexität 	<ul style="list-style-type: none"> • Zu wenige Testfälle definiert • Testfälle zu allgemein • Fehlerrate zu nieder angesetzt • Unterschätzte Komplexität 	<ul style="list-style-type: none"> • Zu wenige Testfälle definiert • Testfälle zu allgemein • Fehlerrate zu nieder angesetzt
4	<ul style="list-style-type: none"> • Testfälle zu detailliert dargestellt (Kosten/Nutzen) • Fehlerrate zu hoch angesetzt 	<ul style="list-style-type: none"> • Mangelnde Treffgenauigkeit der Testfälle • Fehlerrate zu hoch angesetzt 	<ul style="list-style-type: none"> • Zu wenige Testfälle definiert • Testfälle zu allgemein • Mangelnde Treffgenauigkeit der Testfälle • Fehlerrate zu hoch angesetzt • Überschätzte Komplexität
5	• Unterschätzte Komplexität	•OK	<ul style="list-style-type: none"> • Zu wenige Testfälle definiert • Testfälle zu allgemein • Überschätzte Komplexität
6	<ul style="list-style-type: none"> • Fehlerrate zu nieder angesetzt • Unterschätzte Komplexität 	<ul style="list-style-type: none"> • Fehlerrate zu nieder angesetzt • Unterschätzte Komplexität 	<ul style="list-style-type: none"> • Zu wenige Testfälle definiert • Testfälle zu allgemein • Fehlerrate zu nieder angesetzt
7	<ul style="list-style-type: none"> • Testfälle zu detailliert dargestellt (Kosten/Nutzen) • Fehlerrate zu hoch angesetzt 	<ul style="list-style-type: none"> • Testfälle zu detailliert dargestellt (Kosten/Nutzen) • Mangelnde Treffgenauigkeit der Testfälle • Fehlerrate zu hoch angesetzt 	<ul style="list-style-type: none"> • Mangelnde Treffgenauigkeit der Testfälle • Fehlerrate zu hoch angesetzt
8	<ul style="list-style-type: none"> • Fehlerrate zu hoch angesetzt • Unterschätzte Komplexität 	<ul style="list-style-type: none"> • Fehlerrate zu hoch angesetzt • Unterschätzte Komplexität 	<ul style="list-style-type: none"> • Mangelnde Treffgenauigkeit der Testfälle • Fehlerrate zu hoch angesetzt
9	• Unterschätzte Komplexität	• Unterschätzte Komplexität	•OK

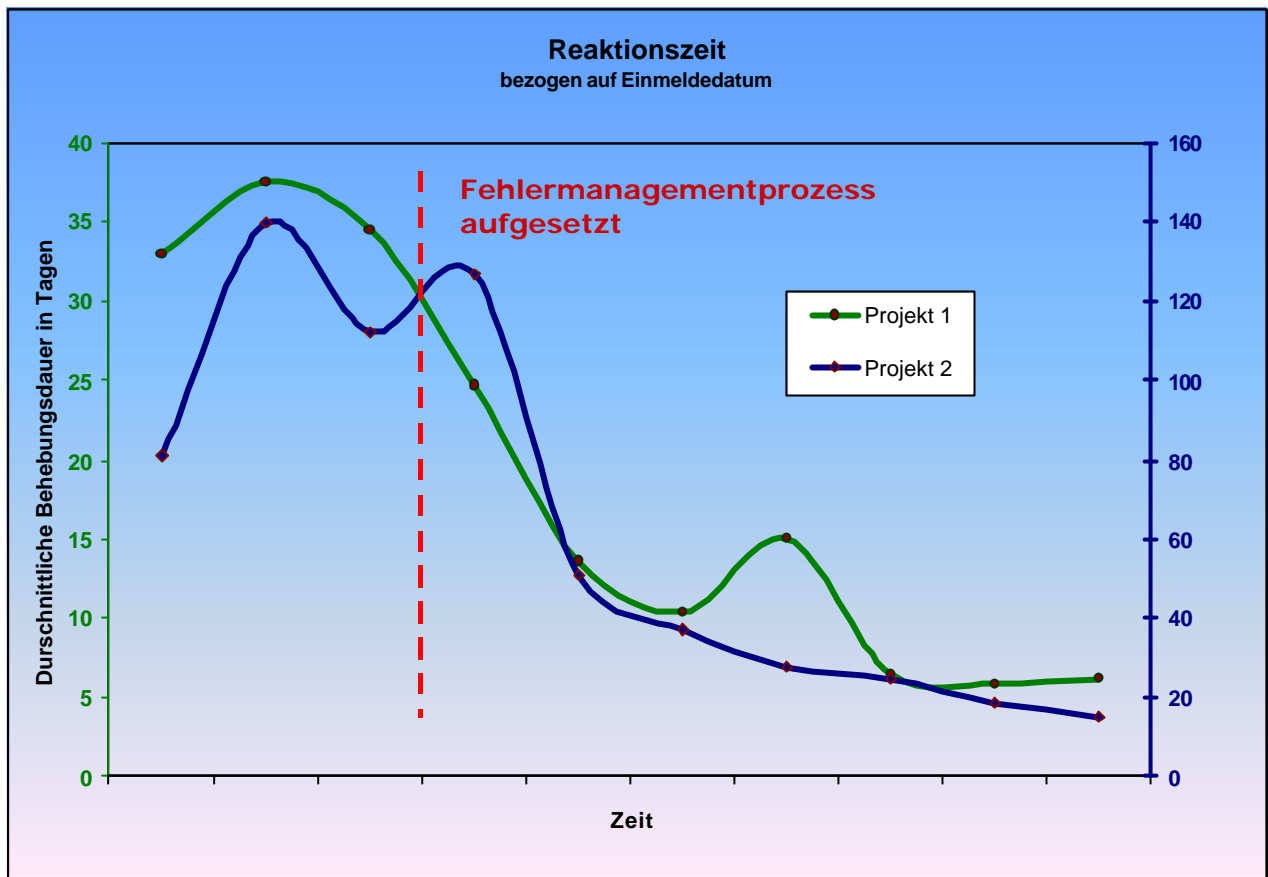
4.3 Projekt und Test Performance – Der Fehlermanagementprozess

Ausgangslage: Alle identifizierten Fehler werden in einem Fehlermanagementwerkzeug erfasst und entsprechend eines zugrundeliegenden Workflows abschließend bearbeitet.

Zielsetzung: Sicherstellung kurzer Fehlerbehebungszyklen, im Idealfall korrelierend zu den Test-Release-Zyklen.



Interpretation: Diese gängige Darstellung zeigt den aktuellen Status aller offenen Fehlereinträge und hilft bei der Feststellung von Engpässen im Test (z.B. Verifikation) und in der Entwicklung. Dementsprechend können Steuerungsmaßnahmen getroffen werden.

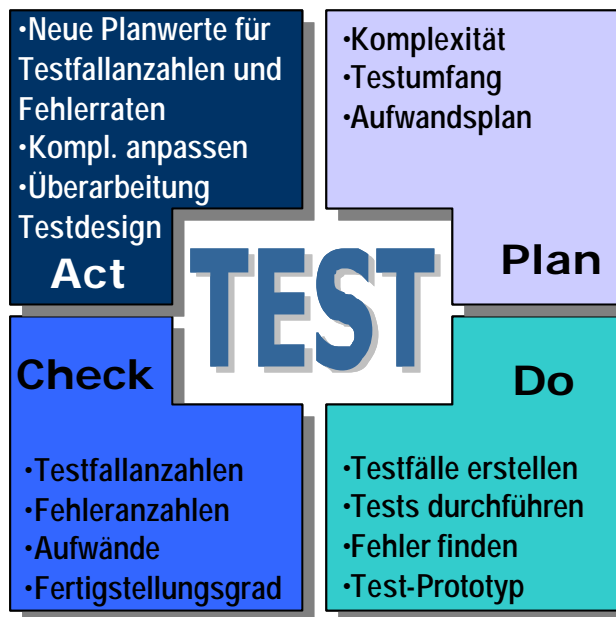


Interpretation: Die obige Grafik zeigt die durchschnittliche Zeitspanne von der Erfassung bis zur Schließung eines Fehlereintrages in zwei unterschiedlichen Projekten. Die Auswirkungen eines kontrollierten Fehlermanagementprozesses sind unter anderem an der Beschleunigung der Fehlerbearbeitung deutlich erkennbar.

4.4 Testplanung und Controlling – Laufende Anpassung

Ausgangslage: Die Testplanung basiert auf Annahmen über Komplexitäten der Testobjekte, Aufwände für einzelne Testphasen (Testdesign, Testdurchführung und Fehlerverfolgung) und Fehlerraten. Für alle Planungsgrößen existieren auch Ist-Aufzeichnungen.

Zielsetzung: Laufende Verifikation der Annahmen sowie Anpassungen der Testpläne auf Grund des aktuellen Testfortschrittes.



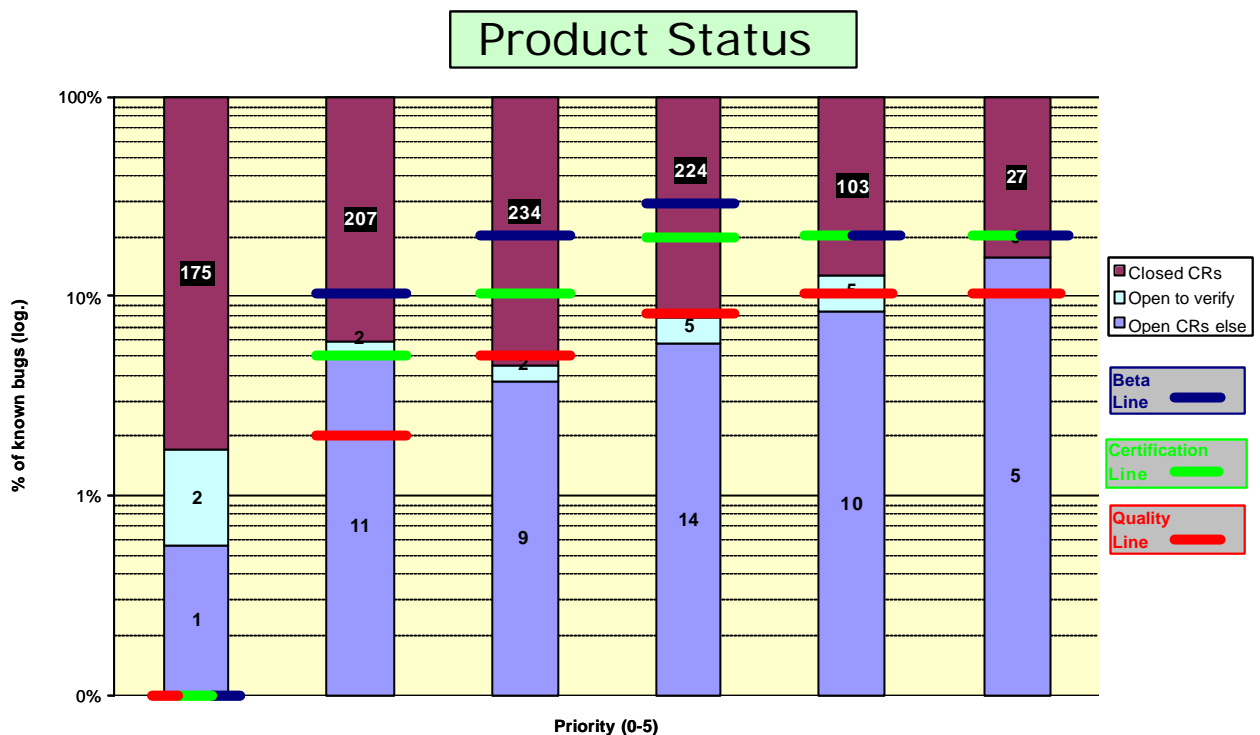
Interpretation: Bereits auf Basis einer kleinen Auswahl an Testobjekten jeder Komplexitätseinstufung können Planannahmen angepasst und eine qualifizierte Hochrechnung auf zukünftige Aufwände gemacht werden.

Wesentliche Planungsgrößen sind Komplexität, davon abgeleiteter Testumfang (Testfälle), erwartete Fehlerraten und Aufwände je Testergebnistyp.

4.5 Produktqualität – Die Quality-Line

Ausgangslage: Gefundene Fehler werden in Fehlerklassen eingeteilt und im Fehlermanagementwerkzeug verwaltet. Die Anzahlen der offenen und geschlossenen Einträge spiegeln den aktuellen Informationsstand bzgl. der Qualität des entwickelten Produktes wieder.

Zielsetzung: Neben anderen Kriterien, wie dem erlangten Testabdeckungsgrad oder Fehlertrends, dienen Kennzahlen hinsichtlich des Fehlerbehebungsgrades als Entscheidungsgrundlage für das Ende der geplanten Testdurchführungen je Teststufe und für die Freigabe von Produktreleases.



Interpretation: Für das entwickelte Software-Produkt ist vorab festgelegt, wie viele Fehler je Fehlerklasse zum jeweiligen Freigabezeitpunkt (Beta-Test-Release; zertifizierte Release) noch offen sein dürfen, ohne dass dies eine Freigabe verhindert. Diese Fehler werden als „Known Bugs“ in die Releasebeschreibung mitaufgenommen.

Nur für die Klasse der „Blocker“ ist dieser Wert mit „0“ angesetzt. Für die restlichen Fehlerklassen sind anstatt absoluter Anzahlen unterschiedliche Prozentwerte definiert, um so auch der Gesamtkomplexität der Applikation Rechnung zu tragen.

Die Quality Line definiert den anzustrebenden Wert in der nachfolgenden Produktbetreuung.

5 Risiken beim Einsatz von Metriken

Wie in den vorhergehenden Kapiteln aufgezeigt, konnte in einigen Projekten durch den Einsatz weniger, aber dafür konsequent angewendeter Metriken ein wesentlicher Beitrag zum Projekterfolg geleistet werden. Dennoch ist die Anwendung von Messungen immer auch mit besonderen Risiken verbunden, die nicht den erhofften Nutzen sondern Schaden nach sich ziehen. Dies geschieht oft dann, wenn die Messergebnisse - anstatt auf die Prozesse oder auf das Produkt - auf Teams oder handelnde Personen angewandt werden:

- **Bedingungslose Transparenz.** Die Darstellung von Messergebnissen, meist in schön aufbereiteten Grafiken oder Listen, liefert ein schnell erfassbares Bild vom Status des Projektes oder des Produktes. Positiv betrachtet, handelt es sich bei dieser Transparenz um das Schaffen und Gewähren eines Einblicks in wichtige Informationen des Projektes. Negativ wahrgenommen, kann diese aber auch als das Bloßstellen gewisser Ergebnisse oder gar Personen gesehen werden. „Einblick gewähren“ oder „Bloßstellen“ – die Grenze ist oft schleichend und verlangt immer einen sensiblen Umgang in der Definition, Darstellung und Erläuterung der Metriken sowie in der Ableitung von Maßnahmen aufgrund der Messergebnisse.
- **Unzulässige Interpretation der Ergebnisse.** In Anlehnung eines bekannten Spruches ist zu sagen: „Deute keine Statistik, deren Modell du nicht voll verstehst“. Dies ist besonders dann ein heikler Punkt, wenn versucht wird, das statistische Material zur Bewertung (Produktivität, Qualität,...) handelnder Personen heranzuziehen.

Fall: Ein Manager verwendete das Zahlenmaterial aus dem Fehlermanagement-Werkzeug dazu, um festzustellen, welcher Entwickler am meisten Fehler zugeordnet hatte und gebrauchte diese Informationen als Bewertungskriterium im Mitarbeitergespräch. Was der Manager übersah, war die Tatsache, dass aus der Zuordnung eines Fehlers zu einem Entwickler noch lange nicht darauf geschlossen werden kann, dass dieser Entwickler den Fehler verursacht hat. Im konkreten Fall traf es den Entwickler doppelt unglücklich: zum einen hatte er Code-Teile eines Mitarbeiters übernommen,

der das Unternehmen verlassen hatte und den Code im schlechten Zustand hinterließ, zum anderen wurden ihm vom Projektleiter auch andere problematische Fehler zugeordnet, da es sich um einen besonders zuverlässigen Entwickler handelte.

- **Falsch verstandener Sportgeist.** Es geht nicht wie im Sport darum, wer höher springt oder schneller im Ziel ist, nicht darum wer mehr Fehler findet oder weniger produziert.

Fall: Eine ausgehängte Liste, ein Ranking, welcher Tester am meisten Fehler gefunden hatte, führte dazu, dass ein findiger Tester innerhalb eines Tages gleich 50 Fehler einmeldete. Tatsächlich handelte es sich um lediglich einen Fehler, der zu einer Vielzahl von Folgefehlern führte, da praktisch nichts mehr funktionierte.

- **Tester als natürlicher Feind des Entwicklers.** Dieses gängige Bild kann durch Anwendung bestimmter Metriken durchaus noch verstärkt werden. Aber letztlich ist es das Ziel, als Team ein gemeinsames, hochwertiges Ergebnis zu liefern. Im sensiblen Umgang mit den Messergebnissen und den handelnden Personen liegt ein wesentlicher Erfolgsfaktor für den Einsatz der unterschiedlichen Projektmetriken.

Fall: In den Fehler-Reports des Quality Managers fanden sich immer nur die Aufstellungen, wie viele und welche Fehler von der Entwicklung noch zu beheben waren. Während also einerseits die neu gefundenen Fehler als Erfolg des Testteams dargestellt wurden, waren diese gleichzeitig „Misserfolg“ und Last des Entwicklungsteams. Diese einseitige Anwendung der Metriken führte zu Lagerbildung und gegenseitigen ablehnenden Argumentationen: „dies ist kein Fehler“; „der Fehler ist nicht schwerwiegend“, etc. Als in den Darstellungen auch die Aufgaben des Testteams aufgenommen wurden, z.B. zu verifizierende Einträge, und die Beseitigung eines Fehlers nicht nur als Verdienst des Entdeckers (Testers), sondern auch als Leistung des Behebbers (Entwicklers) kommuniziert wurde, stieg auch die Akzeptanz der eingesetzten Werkzeuge und Metriken.

6 Abschluss

Die in diesem Artikel diskutierten Metriken beziehen sich primär auf den Testprozess und dessen Einfluss auf die Qualität der entwickelten Software. Auf einer anderen Ebene betrachtet ist aber auch der Test selbst eine Metrik, ein „Messinstrument“ für den Entwicklungsprozess und dessen Ergebnisse im Software-Projekt. Somit bietet ein gut aufgesetzter Testprozess viele Chancen, Verbesserungen herbeizuführen. Aber auch die mit dem Einsatz von Metriken verbundenen Risiken treffen auf den Test zu. Es ist daher notwendig, nicht nur Prozesse und Metriken zu definieren, sondern auch die technischen, organisatorischen und ganz besonders die „kulturellen“ Rahmenbedingungen für deren Einsatz zu schaffen.