Fehler-Metriken bei der Initiierung und Verfolgung eines SPI-Programms

Ein Vortrag im Seminar Software Process Improvement - das Business-Reenginieering der Software-Industrie

Marc Schanne

Forschungsbereich
Programmstrukturen, FZI

2. Juli 1999, Karlsruhe

Software Process Improvement -
das Business-Reenginieering der Software-Industrie:
Stellen sie dar, wie und warum sich Fehler-Metriken besonders gut für die Initiierung und Verfolgung eines SPI Programms eignen.

Marc Schanne

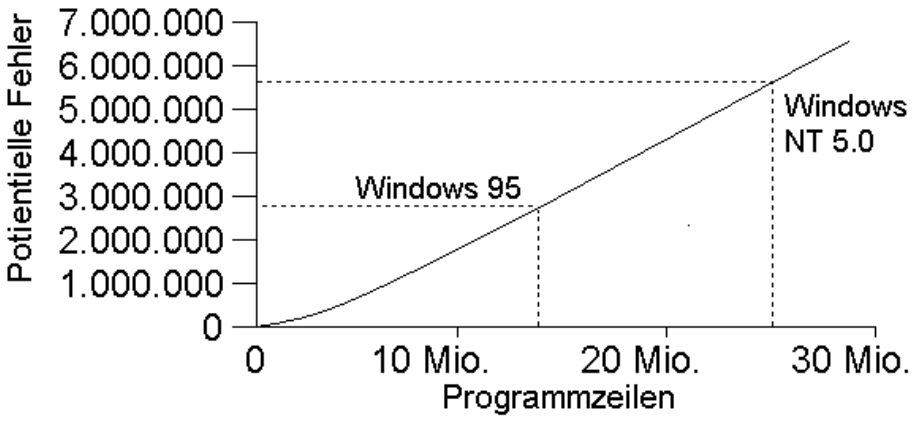
Ein Seminarvortrag am

Forschungszentrum Informatik (FZI),
Forschungsgruppe Programmstrukturen,
Sommersemester 1999

Universität Karlsruhe

2. Juli 1999

Motivation



Fehler-Metriken und SPI Marc Schanne Juli 1999

In [Lew98] stellt Ted Lewis einige Schätzungen zu den Fehler-Potentialen großer Software-Systeme an.

Ausgehend von LOC bei Microsoft Windows 95 errechnet er die ungefähre Größe mit 150.000 Funktionspunkten. Ausgehend hiervon läßt sich ein Fehler-Potential von 2,95 Mio. Fehler bestimmen.

Potentielle Fehler sind nicht gleich tatsächliche Fehler, aber sie sind eine Möglichkeit zu schätzen, wie viele Tests nötig sind um fehlerfreie Software auszuliefern. Eine weitere Daumenregel besagt, dass durch jeden Test- oder Inspektionsschritt der potentiellen Fehler um 30% reduziert wird.

Um das Fehler-Potential von 2,95 Mio. auf dann tatsächlich ausgelieferte 5.000 Fehler zu reduzieren waren bei Microsoft ca. 18 Test-Iterationen nötig. Eine Reduktion auf nur einen Fehler wäre erst mit 42 Durchläufen möglich.

Gliederung

- **Fehler**
- **Metriken**
  - Definition
  - Anforderungen & Gütekriterien
- **Software Prozess**
  - Persönlicher Software Prozess (PSP)
  - Messen von Fehlern
  - GQM-Ansatz
- **Software Prozess Verbesserung**
  - Verbesserungsbereiche
  - Fehler-Metriken
  - Durchsichten / Inspektionen
  - Ausbeute (yield)
- **PSP *vs.* CMM**

Fehler-Metriken und SPI Marc Schanne Juli 1999

Zusammenfassung

Die Auslieferung fehlerfreier Software ist das Ziel jedes Entwicklers. In diesem Seminarvortrag soll ausgehend von Fehler-Metriken die Möglichkeit zur Verbesserung des gesamten Software Prozesses durch die Einbeziehung der Ergebnisse von Metriken aufgezeigt werden.

Schon heute werden Fehler-Metriken bei der Software-Qualitätssicherung eingesetzt. Auch wenn es noch schwierig ist, Software-Qualität zu messen, werden Metriken zum besseren Verständnis unterschiedlicher Qualitätsmerkmale verwendet.

Software-Qualität ist auch von der Güte des Software-Entwicklungs-Prozesses abhängig. Mit Metriken, besonders Fehler-Metriken, läßt sich dieser Prozess kontrollieren, steuern und verbessern.

Beispielhaft hier wird der Einsatz im Persönlichen Software Prozess (PSP) vorgestellt. Ein Vergleich des PSP mit den 5 Prozess-Reifegraden des Capability Marturity Model (CMM) ermöglicht die Übertragung der Verwendung auf unternehmensweiten Software Prozessen.

Fehler-Würfel

Kundensicht:

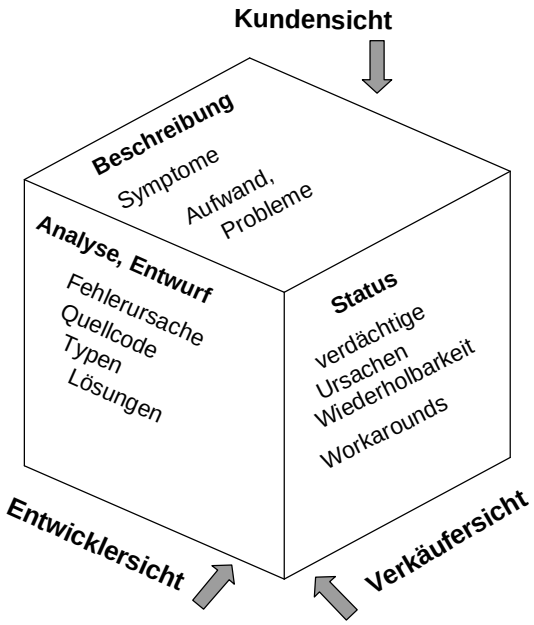
- kritisch, ernst, Verlust von Zeit und Geld

Verkäufersicht:

- Hardware-Subsysteme, Programmkomponente, Dokumentation

Entwicklersicht:

- Entwurfsfehler, Sourcecode, falscher Datentyp, fehlende Logik



Fehler-Metriken und SPI Marc Schanne Juli 1999

Nach [Gra92] läßt sich ein Software-Fehler aus drei verschiedenen Sichten betrachten. Jede Sicht ist charakterisiert durch ihre eigene (Fach-)Sprache und eigenes Hauptproblem.

Das Fehlverhalten eines Systems bedeutet für den Benutzer, dass er nicht das gewünschte Ziel erreichen kann und dieses „ernste und kritische“ Probleme mit sich bringt, mit dem Verlust von Zeit oder Geld zusammenhängt.

Ein Verkäufer zwischen Kunde und Produzent interessiert, welche Programmkomponente für den Fehler verantwortlich ist, ob ein Patch existiert und ab wann eine fehlerfreie Version vorhanden ist.

Die Entwicklersicht beschreibt den Fehler als Folge von gewählten Entwurfentscheidungen, bezeichnet Quellcode-Bereiche und macht Aussagen darüber wie schwierig die Fehlerbehebung ist und wie lange diese benötigen wird.

Metriken

Definition: Metrik
(in der SWT oft synonym zum Begriff Maß (measure))

Abbildung: Zuweisung einer Zahl oder eines anderen Symbols (z.B.: small, medium, large) an ein Objekt zur Beschreibung der Eigenschaft eines Attributs des Objekts.

Anwendungsbereiche für Maße:

- Beschreibung
  einzelner Aspekte als Grundlage für Bewertung und Planung
- Beurteilung
  von Qualität, Effektivität oder Effizienz von Produkten oder Prozessen (Grundlage für Verbesserungen)
- Vorhersage
  für eine objektive Planung

Metriken (2)

Projekte beginnen mit Träumen:

Metriken vereinfachen die Entwicklung (Planung: Ressourcen, Zeit, Größe)

Gütekriterien, Anforderungen an Maße und Metriken:

- Wohldefiniertheit
- Objektivität
- Zuverlässigkeit
- Validität
- Aussagekraft
- Sinnhaltigkeit
- Berechenbarkeit

Fehler-Metriken und SPI Marc Schanne Juli 1999

Metriken werden in der Software-Entwicklung verwendet um die Entwicklung von Produktideen auf eine solide planerische Basis zu stellen. Um die Nützlichkeit von Maßen für diesen Zweck sicher zu stellen, lassen sich verschiedene Gütekriterien an Maße und Metriken stellen:

•**Wohldefiniertheit**: vollständige und genaue Definition

•**Objektivität**: kein subjektiver Einfluss des Messenden auf die Messung

•**Zuverlässigkeit (Messgenauigkeit)**: Maß ist stabil und präzise (bei Wiederholung gleiche Ergebnisse)

•**Validität**: Gültigkeit, Messtauglichkeit

•**Aussagekraft, Nützlichkeit**: Information über ein Attribut, das von Belang ist

•**Sinnhaltigkeit**: qualitativ gleichartige Abbildung für gleichartige Werte

•**Berechenbarkeit**: Berechnungsvorschrift algorithmisch und berechenbar

Metriken (3)

Daten sammeln:

- Vergleich mit historischen Daten
- Vorausplanung

Beispiel: Software Projekt Management mit Metriken:

- richtiges Produkt wählen
- Projekt effektiv bearbeiten
- rechtzeitige Markteinführung

Grundlage für den erfolgreichen Einsatz von Metriken ist das Sammeln historischer Vergleichsdaten.

Ein Beispiel für den profitablen Einsatz von Metriken kann das Software Projekt Management in einem Unternehmen sein: Historische Daten über frühere Projekte können - natürlich zusammen mit allgemeinen Marktuntersuchungen - bei der Auswahl des „richtigen“ Produkts helfen, die Planung des Projekts unterstützen und Qualitätsbestimmungen können die Einführung des Produkts auf dem Markt sichern.

Software Prozess (SP)

Definition: Software Prozess

Entwicklung von Software als Prozess in Phasen aufgefasst.
Jede Phase enthält folgende Teile:

- Zweck
- Handelnder
- Eintrittsbedingungen
- Aufgaben
- Ausgangskriterien
- nächste Phase

Beispiel-Prozess nach PSP0:

1) Planung
2) Entwurf
3) Implementation
4) Übersetzung
5) Test

PSP

Persönlicher Software Prozess

Software Prozess für den einzelnen

2 Hauptziele:

- Planung
- Qualitätsverbesserung

Vergleich mit CMM:

vor- & selbstdefinierter Prozess auf Stufe 4
(Übergang zu Stufe 5 vorgezeichnet)

Der Persönliche Software Prozess eignet sich als positives Beispiel für den Einsatz von Maßen.

Planung: LOC

Qualitätsverbesserung:

Fehlermanagement: Fehler vermeiden, sonst zumindest früh entdecken.

Der CMM-Ansatz enthält fünf Reifegradstufen, welche die Qualität des Software-Entwicklungsprozesses beschreiben:

Stufe 1: Chaotischer, ad hoc Prozess

Stufe 2: Intuitiver Prozess

Stufe 3: Qualitativer Prozess

Stufe 4: Quantitativer Prozess

Stufe 5: Rückgekoppelter Prozess

Messen von Fehlern

Für einen quantitativen Prozess werden Metriken benötigt, **aber**

Messen (allgemein) braucht Zeit

Vorteil von Fehler-Metriken:
Einfachheit (prozessnah, leicht erfassbar)

Nachteil: Sabotage durch Mitarbeiter bei Missverständnis über Notwendigkeit von Fehlersammlung möglich

Fehler-Metriken und SPI Marc Schanne Juli 1999

Jede Art der zusätzlichen Datenerfassung neben der Softwareentwicklung braucht Zeit.

Das einfachste Maß der Softwaretechnik, Lines of Sourcecode (LOC), kann unter Einhaltung bestimmter Codierungsregeln sogar mit unterstützenden Tools automatisch berechnet werden. Auch bei der Beseitigung von Fehlern kann dies ohne all zu viel Mehraufwand geschehen.

Humphrey hat für den PSP in [Hum95] ein Fehler-Formblatt eingesetzt. Hier werden maßgebliche Daten katalogisiert.

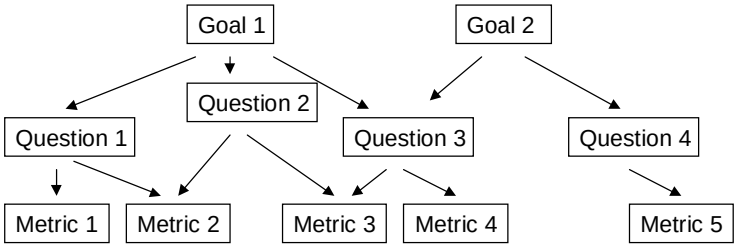
Speziell bei die Messung von Fehlern in größeren Entwicklergruppen ist der Erfolg von der Unterstützung alle Mitarbeiter abhängig. Bei der Einführung von Software Metriken müssen alle das damit beabsichtige Ziel unterstützen. Insbesondere dürfen Metriken nicht **gegen** die Entwickler selbst gerichtet werden.

Die Sammlung von (Fehler-)Daten kann zum Beispiel auch über den GQM-Ansatz aus den Unternehmens- und Entwicklerzielen, wie Produktion fehlerfreier Software oder Verbesserung des Software Prozesses vermittelt werden.

Qoal Question Metric (GQM)

nach **Basili und Weiss** (1984):

1) Definiere die grundlegenden Ziele für die geplante Aktivität. (Goal)
2) Finde möglichst viele Fragen, die helfen die Ziele zu erreichen. (Question)
3) Definiere und ermittle Maße, die benötigt werden um die Fragen zu beantworten. (Metric)



Fehler-Metriken und SPI Marc Schanne Juli 1999

Der GQM-Ansatz stellt eine sehr allgemeine Möglichkeit dar, Entwicklungsziele zu verfolgen.

Das Finden möglichst vieler relevanter Fragestellungen erleichtert auch die Entwicklung der für ihre Beantwortung nötigen Maße. Die Frage bietet einen Maßstab zu ihrer Beurteilung und erleichtert das Überprüfen von Validität, Genauigkeit und Zuverlässigkeit.

GQM-Ansatz: Umgang mit Fehlern

- Wie oft treten solche Fehler auf?
- Wie „teuer“ ist es, solche Fehler zu beheben?
- Welche Komponenten sind anfällig für solche Fehler?
- Mit welcher Prozess-Veränderung wird der Fehler entdeckt oder vermieden?

Fehler-Metriken und SPI Marc Schanne Juli 1999

Ausgehend vom Ziel, die Anzahl von Fehlern zu reduzieren, kann man über Fragen wie „Wie ‚teuer‘ ist es, solche Fehler zu beheben?“, „Wie oft treten solche Fehler auf?“ grundlegende Fehler im Entwurf oder der Implemetation der Software bewerten.

Nicht nur am konkreten Produkt können somit Verbesserungen vorgenommen werden, sondern durch Fehler, die durch den mangelnden Entwicklungsprozess nicht gefunden und behoben werden, wird auch ein Hinweis auf die Verbesserung des Prozesses gegeben.

Software Prozess Verbesserung (SPI)

Haupteigenschaften eines Software Prozesses:

- Wohldefiniertheit
- Quantitatives Verstehen
- Kontinuierliche Änderung

Der existierende Software Prozess läßt sich durch Veränderungen in diesen drei Merkmale verbessern.

SPI ist kein kurzfristiger Prozess!

Ein bestehender Software Prozess kann in folgenden 3 Merkmalen verbessert werden:

Wohldefiniertheit: Regeln müssen besonders in einem größeren Team allen Mitarbeitern bekannt sein. Wenn der Software Prozess nicht klar definiert ist, besteht sonst die Gefahr, dass Arbeitsschritte unvollständig, falsch oder doppelt erledigt werden. Außerdem sind Veränderungen besser auf den Prozess anwendbar.

Quantitatives Verstehen: Z.B. muss im Prozess quantitativ nachvollziehbar sein, welche Ressourcen für welchen Prozessschritt benötigt werden. Dies ist direkte Grundlage für Prozessverbesserungen.

Kontinuierliche Änderung: Ohne diese Voraussetzung sich kontinuierlich an geänderte Rahmenbedingungen (Werkzeuge, Anwendungsgebiete, ...) anpassen zu können, wird der Prozess auf absehbare Zeit ungeeignet oder passt sich selbst unkontrolliert an und verliert die Wohldefiniertheit.

Insbesondere der letzte Punkt bewirkt, dass die Verbesserung kein punktuelles Ereignis darstellt, sondern durch sich verändernde Rahmenbedingungen eine dauerhafte Herausforderung an Unternehmen darstellt. Gute Software-organisationen haben deshalb oft eine Gruppe von Leuten, die sich nur mit der Weiterentwicklung des Softwareprozesses befasst, die Software Engineering Process Group (SEPG). [Pre97, S.117]

Fehler-Analyse

Einführungs- und Entdeckungsphase
in einer Skala nach Bearbeitungszeit/Einführphase
´´ /Entdeckungsphase

Fehler-Klassifizierung, z.B.:

nach Kategorien (Entwurf, Implementation, System, ...)

nach Gewichtigkeit (Auswirkung auf Entwurf, ...)

Um eine spätere Weiterverarbeitung der Fehlerdaten zu gewährleisten, ist es notwendig, jeden aufgetretenen Fehler genau zu spezifizieren.

Eine klare Bestimmung der Einschleppungs- und Entdeckungsphase ist für die Anpassung des Entwicklungsprozesses nötig. Signifikant auffällige Ergebnisse deuten an, dass der Prozess hier näher untersucht werden sollte und evtl. korregiert werden muss.

Außerdem lassen sich Fehler nach Arten unterscheiden. Der PSP verwendet hierfür ein einfaches Schema mit 10 Klassen zur Kategorisierung der Fehler:

Documentation; **Syntax**; **Build, Package** (Bibliotheken, Versionskontrolle), **Assignment** (Deklaration, doppelte Bezeichner, Scope), **Interface** (Prozedur-aufrufe, I/O), **Checking** (Fehlermeldungen, unzureichende Checks), **Data** (Strukturen, Inhalt), **Function** (Logik, Berechnungen), **System** (Konfiguration, Timing, Speicher), **Environment** (Hilfssysteme)

PSP & Fehler-Metriken

Kein Mangel an Übertragbarkeit

Einzelperson => Konsistenz, Vergleichbarkeit

Metriken sind wohldefiniert und aussagekräftig und tatsächliche Grundlage für Verbesserung.

Sie basieren im Wesentlichen auf:

Phase, Zeit, KLOC

Fehler-Metriken und SPI Marc Schanne Juli 1999

Die Fehlersammlung ist im PSP Grundlage für die Korrektur des Prozesses. Insbesondere die Einführung neuer Prozess-Phasen können mit dem Ziel der Fehlerreduktion und Qualitätsverbesserung motiviert werden.

Beispiele:

Produktivität - LOC/Stunde :
Entdeckungsquote - % der frühzeitig entfernten Fehler

(Optimierung der frühzeitigen Fehlerentdeckungsphasen verbessert die Produktivität.)

Übersetzungsfehler : Testfehler

(Es besteht Korrelation! Annahme: Jeder Fehler (auch ein trivialer) ist schädlich.)

Fehler/Stunde : Fehlerfindungsmethode

(Nachweis der Effizienz verschiedener Fehlerfindungsstrategien.)

Durchsichten / Inspektionen

Durchsichtteam

1 Entwickler, 3-5 Gutachter

Entwickler erklärt den Gutachtern die Arbeit, Gutachter suchen die Fehler.

Inspektionsteam

(Fehler werden anhand von Checklisten gesucht.)

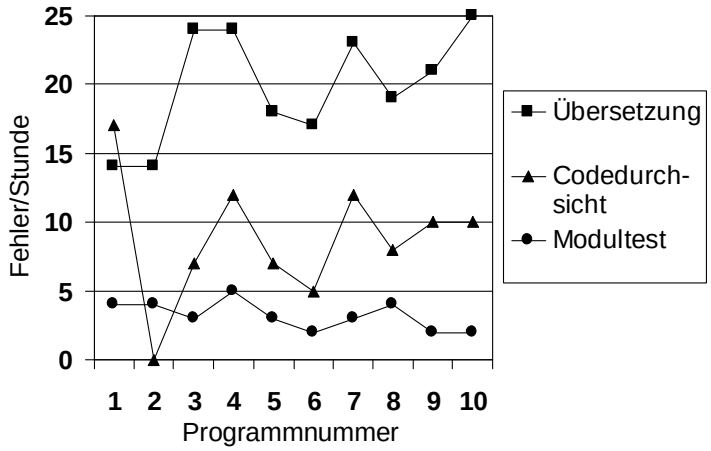
- Moderator: verteilt Arbeit, leitet Diskussion
- Entwickler: liest vor und erklärt das Produkt
- Tester
- Entwerfer

Inspektionen ähneln Durchsichten. Die Überprüfung erfolgt jedoch anhand von Checklisten.

Im PSP wird ebenfalls durch Einführung von persönlichen Checklisten versucht die Durchsichten effizienter zu gestalten.
Für Fehler, die häufig von Durchsichten nicht entdeckt und erst in der Übersetzungs- oder Testphase beseitigt werden konnten, wird durch Ergänzung der Durchsicht um eine spezielle Entdeckungs-Teilphase für diesen Fehlertyp die Ausbeute der Durchsicht verbessert werden.

Als Maß, zur Beurteilung ob die Codebegutachtung zu ausführlich oder zu flüchtig vorgenommen wurde, verwendet der PSP die Begutachtungs-geschwindigkeit in KLOC/Stunde.

Durchsichten / Inspektionen (2)



Fehler-Metriken und SPI Marc Schanne Juli 1999

In Durchsichten werden Fehler direkt gefunden; beim Test nur Symptome, deshalb muss noch ein erheblicher Zeitaufwand in das Finden des Fehlers selbst, passend zum Symptom, gesteckt werden (Debugging).

Während der Durchsicht durch den Programm-Text entwickelt der Programmierer ein gedankliches System von logischen Beziehungen und Konstrukten, die stufenweise die Funktionalität des Programms ergeben. Wenn das Programm eine unerwartete Funktion ausführt, kann der Begutachter durch noch genauere Untersuchung entweder erkennen, dass das Programm hier korrekt ist oder direkt einen Fehler im Programm finden.

Beim Debugging ist die Situation anders. Der Software-Tester beginnt mit dem unerwarteten Systemverhalten des Programms. Sicher kann so mit der entsprechenden Unterstützung durch Debug-Software der Fehler schnell gefunden werden, aber oft läßt sich das gezeigte Symptom nicht so einfach auf den tatsächlichen Programm-Fehler beziehen.

Der tatsächliche Vorteil eines Debuggers besteht darin, dass er den Entwickler Schritt für Schritt durch die Programmlogik führt und die wichtigen Parameter-Werte anzeigt. Dieser Prozess ist aber nur dann effektiv, wenn der Entwickler weiß, welche Funktion die Parameter haben. Wenn der Entwickler die Quellcode hierfür sowieso inspizieren muss, dann ist es nur noch ein kleiner Schritt hin zu einer vollständigen Durchsicht.

Ausbeute (yield)

Maß für Qualität des Prozesses:

$$\textbf{yield} = 100 * \frac{\textbf{entfernt vor Übersetzung}}{\textbf{eingefügt vor Übersetzung}}$$

Die tatsächliche Ausbeute, kann nur geschätzt werden, da die Zahl der ausgelieferten Fehler unbekannt ist, aber eigentlich in die Berechnung eingehen müsste.
Aber auch diese vereinfachte Form (nur bis einschließlich Test-Phase) ist aussagekräftig.

Ausbeute: Beispiel

Beispiel:

Phase	injected	SUM(inj)	removed	SUM(rem)
Plan	1	1	0	0
Design	5	6	0	0
D-Review	0	6	3	3
Code	15	21	1	4
C-Review	0	21	8	12
Compile	0	21	6	18
Test	0	21	3	21

Ausbeute: Beispiel (2)

Mit yield läßt sich der Erfolg jeder einzelnen Prozess-Phase im Anschluss bewerten:

$$\text{Yield(Phase)} = 100 * \frac{\text{entfernt in Phase}}{\text{entfernt in Phase} + (\text{bisher eingefügt} - \text{bisher entfernt})}$$

Beispiel:

yield(Design-Review) = 100 * (3 / (3 + 6 -3)) = **50%**

yield(Code-Review) = 100 * (8 / (8 + 21 - 12)) = **47.1%**

yield(Compile) = 100 * (6 / (6 + 21 - 18)) = **66.7%**

yield = 100 * 12 / 21 = **57.1%**

yield ist zwar ein gutes Maß für die Bewertung von Prozess-Phasen, insbesondere von Durchsichten, aber der Erfolg läßt sich erst am Ende der Begutachtung berechnen.

Als einfaches und frühzeitiger zu berechnendes Maß, das auch gut mit yield korreliert, wird deshalb im PSP schon während der Begutachtung das Maß

gefundene Fehler / begutachtete KLOC

verwendet.

PSP *vs.* CMM

PSP	CMM
SW-Qualitätssicherung Projektverfolgung	Stufe 2
Prozessdefinition	Stufe 3
(Fehler-)Datensammlung SW-Qualitätsmanagement	Stufe 4
Fehlervermeidung Verwaltung	Stufe 5

Fehler-Metriken und SPI Marc Schanne Juli 1999

Die Gliederung des PSP-Kurses in 4 Stufen mit 3 Zwischenstufen ermöglicht es dem einzelnen Entwickler Merkmale der 5 CMM-Reifegrad-Stufen zu erreichen:

PSP0: Zeit- / Fehlererfassung

PSP1: Größenmessung / -schätzung, Terminplanung

PSP2: Qualitätsmanagement (Durchsichten, ...)

PSP3: inkrementelle Entwicklung

Ergebnisse

- Mit Metriken läßt sich Software bewerten, Fehler-Metriken ermöglichen die Qualitätsbewertung
- Fehler-Metriken sind einfach und prozessnah erfassbar
- Datensammlung und Analyse der Fehler ermöglichen Korrekturen am Software Prozess
- Durchsichtsphasen sind effektiv
- Viele Fehlerentfernungsphasen verbessern Produktqualität

Literatur

[Lew98] Lewis, Ted: Joe Sixpack, Larry Lemming and Ralph Nader in IEEE Computer 7/98, S. 107

[Gra97] Grady, Robert B.: Successful software process improvement. 1997

[Gra92] Grady, Robert B.: Practical software metrics for project managment and process improvement. 1992

[Fen97] Fenton, Norman E.; Pfleeger, S. L.: Software Metrics: A Rigorous and Practical Approach. 1997

[Pre97] Prechelt, Lutz: Skriptum zur Vorlesung Ausgewählte Kapitel der Softwaretechnik. 1997

[Hum95] Humphrey, Watts S.: A discipline for software engineering. 1995

[Bal98] Balzert, Helmut: Lehrbuch der Software-Technik. Software-Management, Software-Qualitätsscherung, Unternehmensmodellierung. 1998

[Gra92] ist eine einfache, praxisorientierteEinführung in die Verwendung von Software Metriken für Projekt Management und Prozess Verbesserung.

[Pre97] ist ein vorlesungsbegleitendes Skript für die Vorlesung Ausgewählte Kapitel aus der Softwaretechnik und ist in der aktuellen Version online verfügbar: http://wwwipd.ira.uka.de/~prechelt/swt2/

[Hum95] stellt einen praktischen Einführungskurs in den Persönlichen Software Prozess dar.