

Software-Qualität messen und bewerten: Theorie und Empirie

Prof. Dr.-Ing. habil. Peter Liggesmeyer

Lehrstuhl Software Engineering: Dependability
TU Kaiserslautern

Direktor Fraunhofer Institut für Experimentelles Software Engineering (IESE),
Kaiserslautern

- Motivation
- Einfache Messung von Qualität
- Software-Qualitätsexperimente
- Software-Qualitätsmessung
- Schlussfolgerungen

Motivation Kleine Ursache – große Wirkung



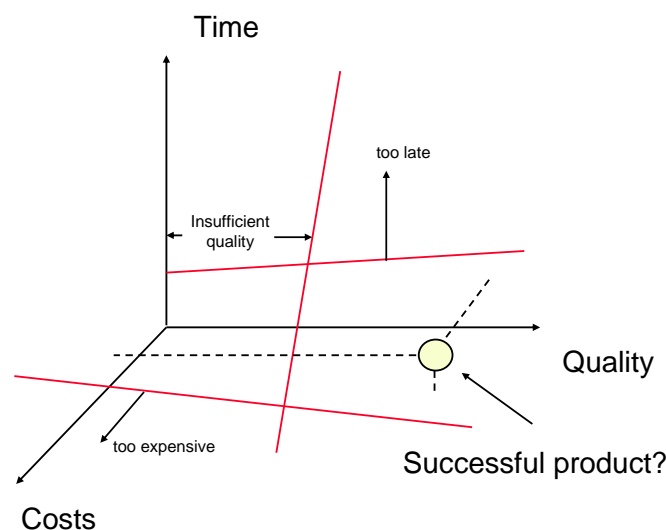
Motivation

Qualität: Qualitativ vs. quantitativ

- Ersetzung qualitativer – oft intuitiver – Aussagen über Software durch quantitative, reproduzierbare Aussagen
- Beispiel:
 - Qualitativ, intuitiv:
 - Der Entwickler sagt: „Ich habe mein Softwaremodul ausgetestet“
 - „Wir wollen Null-Fehler-Software machen!“
 - Quantitativ, reproduzierbar:
 - „Mein Testwerkzeug zeigt zur Zeit eine Zweigüberdeckungsrate von **57% (70 von 123 Zweigen)** an. In unserer Firma gelten Module als ausreichend getestet, wenn die Zweigüberdeckungsrate mindestens **95%** beträgt. Ich muss daher noch mindestens **47** Zweige testen und erwarte aufgrund der Erfahrung mit vergleichbaren Modulen, dass ich dafür etwa **1,5 Mitarbeitertage** zusätzlichen Aufwand benötige.“
 - „Das Restrisiko beträgt 5 FIT.“

Motivation

Der Zielkonflikt für erfolgreiche Produkte



Einfache Messung von Qualität

Notwendige, minimale Anforderungen an Tests

- Absolut notwendig entspr. aller maßgeblichen Standards:
 - Funktionsorientierte Testplanung für alle Testphasen
 - Reproduzierbarkeit von Testergebnissen => automatische Regressionstests nach Software-Modifikationen
- Weitgehender Konsens:
 - Ergänzende strukturorientierte Abdeckung (mindestens Zweigüberdeckungstest)
 - In kritischen Anwendungsbereichen – z. B. der Avionik – werden darüber hinaus durch Standards explizit gründlichere strukturorientierte Tests gefordert
 - Durchführung möglichst in der ersten Testphase nach Fertigstellung des Codes (Modultest)
 - Zusätzlich Leistungs- und Stresstests insbesondere in technischen Anwendungsbereichen

Einfache Messung von Qualität

Eine einfache praxisgeeignete Teststrategie

- Modultest
 - Funktionsorientierter Test der Module unter Verwendung eines Zweigüberdeckungstestwerkzeugs
 - Funktionsorientierte Testfallerzeugung (z.B. funktionale Äquivalenzklassenbildung)
 - Vorbereitung - d.h. Instrumentierung - der zu testenden Module zur Kontrolle der erreichten Zweigüberdeckung
 - Vollständige Durchführung der funktionsorientierten Tests
 - Kontrolle der auf diese Weise erreichten Zweigüberdeckung (erfahrungsgemäß ca. 70% - 80%)
 - Strukturorientierter Test der Module unter Verwendung des Zweigüberdeckungstestwerkzeugs
 - Ursachenanalyse für die Nichtausführung von Zweigen
 - Erzeugung von Testfällen für die noch nicht durchlaufenen Zweige
- Integrations- und Systemtest
 - Funktionsorientierter Test

Motivation

Qualitätsmessung in der Softwareentwicklung

- Software wird heute vielfach in Anwendungsbereichen eingesetzt in denen quantitative Aussagen üblich oder notwendig sind:
 - Vertragsgestaltung: „Wir vereinbaren eine Mindestverfügbarkeit des Systems von 99,8%!“
 - Sicherheitsnachweis eines Bahnsystems beim Eisenbahnbundesamt: „Wie hoch ist das Restrisiko durch Softwarefehler?“
 - Ist die zu erwartende Anzahl an Restfehlern hinreichend klein für die Freigabe?
 - Ist die Wahrscheinlichkeit hinreichend gering, dass Softwarefehler in Steuergeräten einen Ausfall unserer Oberklassenlimousine verursachen?
 - Wir benötigen eine ausfallfreie Missionszeit von 4 Wochen. Wird das gelingen?
- Viele Unternehmen haben definierte Prozesse installiert: Der nächste Schritt ist deren quantitative Steuerung

Motivation

Qualitätsmessung in der Softwareentwicklung: Probleme

- Die meisten Qualitätseigenschaften sind nicht direkt messbar!
 - Fehlergehalt
 - Verfügbarkeit
 - Zuverlässigkeit
 - Sicherheit
 - ...
- Man versucht die Qualitätseigenschaften ...
 - ... experimentell zu ermitteln (z.B. Zuverlässigkeit) oder
 - ... aus direkt messbaren Eigenschaften zu schließen (z.B. Fehlergehalt).

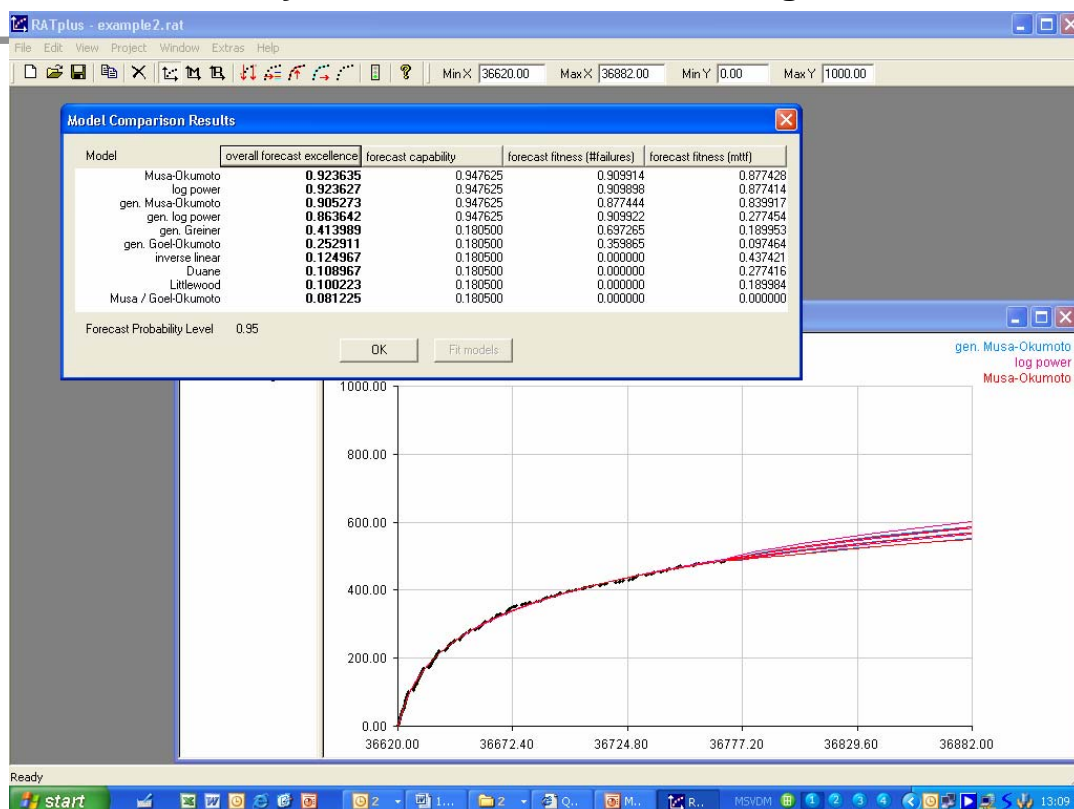
Software-Qualitätsexperimente

Stochastische Analyse von Software-Zuverlässigkeit: Situation

- Eigenständige Forschungsrichtung seit ca. 30 Jahren
- Wenig Einfluss auf die Softwareentwicklung in der Praxis
- Mathematische Basis zum Teil kompliziert
- Sehr viele unterschiedliche stochastische Zuverlässigkeitsmodelle
- A priori Auswahl eines Modells nicht möglich
- Bestimmung von Modellparametern notwendig
- Anwendung der Theorie in der Praxis erfordert leistungsfähige Werkzeugunterstützung

Software-Qualitätsexperimente

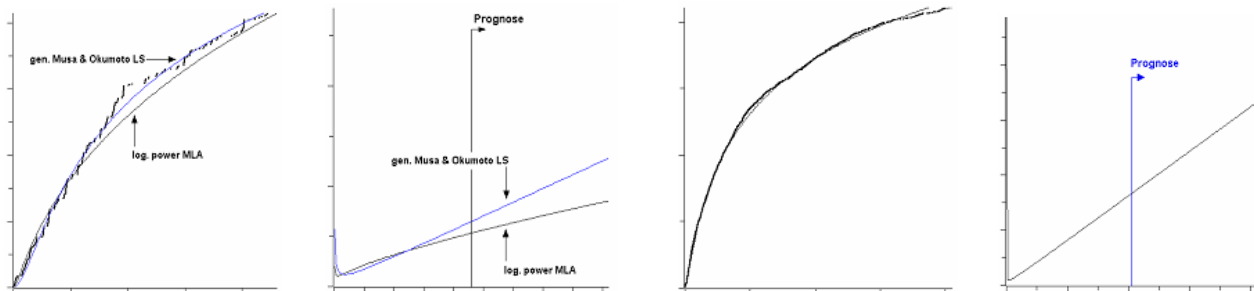
Stochastische Analyse von Software-Zuverlässigkeit



Software-Qualitätsexperimente

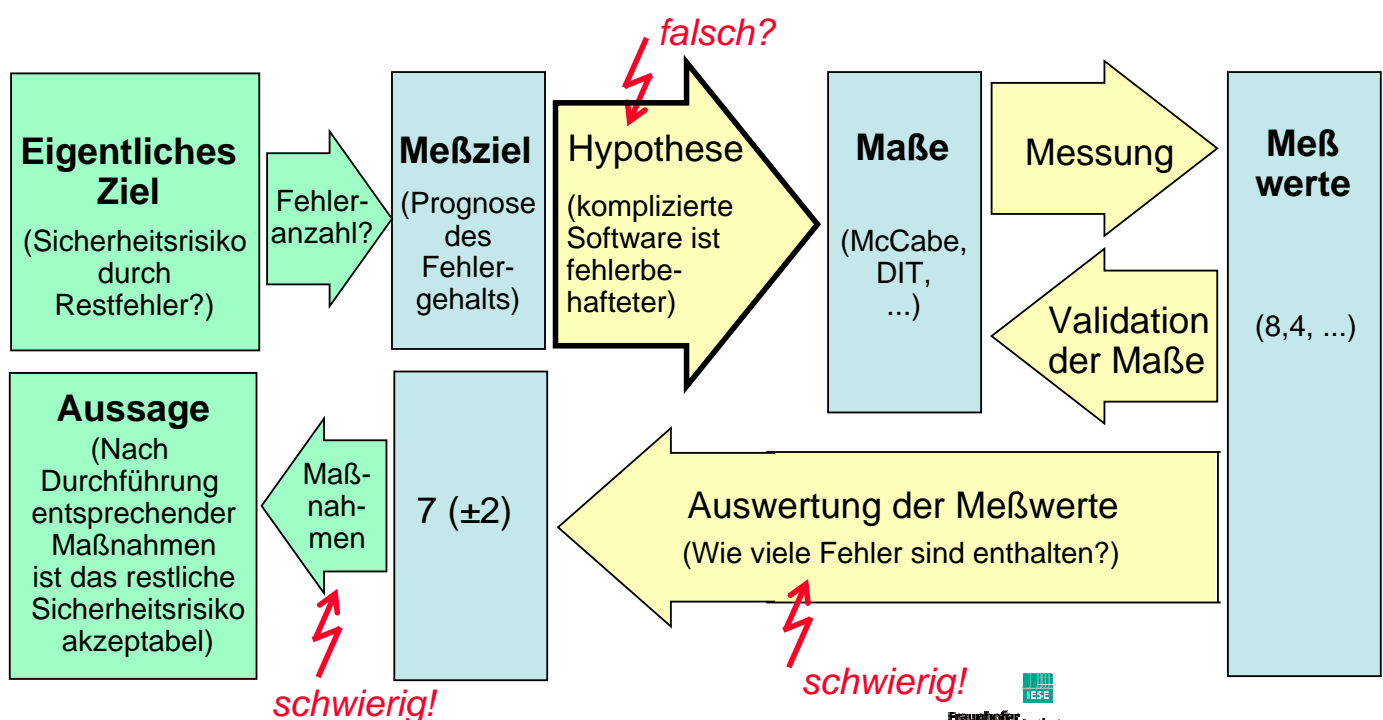
Stochastische Analyse von Software-Zuverlässigkeit: Praktischer Lösungsansatz

Zahlreiche Anwendungen (Verkehrstechnik, Medizintechnik, Telekommunikation, ...)



Software-Qualitätsmessung

Schlussfolgerungskette



Software-Qualitätsmessung

Beliebte Hypothesen in Theorie und Praxis

	/Fenton, Ohlsson 00/	/Basili, et al. 96/	/Cartwright, Shepperd 00/	/Basili, Perricone 84/	/Abreu, Melo 96/
Wenige Module enthalten die Mehrzahl der Fehler.	++	++	(+)	++	/
Wenige Module erzeugen die meisten Ausfälle.	++	/	/	/	/
Viele Fehler im Modultest bedeuten viele Fehler im Systemtest.	+	/	/	/	/
Viele Fehler im Test bedeuten viele Ausfälle im Feld.	--	/	/	/	/
Fehlerdichten korrespondierender Phasen sind über Releases hinweg konstant.	+	/	/	/	/
Umfangsmaße sind geeignet zur Fehlerprognose.	+	/	+	-	/

++: starke Bestätigung; +: schwache Bestätigung; 0: keine Aussage;
 -: schwache Ablehnung; -- starke Ablehnung; /: nicht evaluiert;
 ?: unklar

Software-Qualitätsmessung

Beliebte Hypothesen in Theorie und Praxis: Erkenntnisse I

- Fehler sind über die Module einer Software nicht gleichmäßig verteilt, sondern konzentrieren sich in einigen wenigen Modulen
- Diese Module erzeugen den größten Teil der Probleme
- Großer Modulumfang bedeutet nicht zugleich hoher Fehlergehalt
- Viele erkannte Probleme im Test bedeuten nicht, dass eine Software in der Praxis Qualitätsmängel zeigt
- Es scheint Regeln zu geben, die dafür sorgen, dass aufeinander folgende Entwicklungen ähnliche Ergebnisse liefern

Frage:

- Wie können die wenigen besonders fehlerhaften Module erkannt werden?**

Software-Qualitätsmessung

Beliebte Hypothesen in Theorie und Praxis

	/Fenton, Ohlsson 00/	/Basili, et al. 96/	/Cartwright, Shepperd 00/	/Basili, Perricone 84/	/Abreu, Melo 96/
Code-Komplexitätsmaße sind geeignete Mittel zur Fehlerprognose.	besser als Umfangsmaße: -	WMC: +	WMC: /	besser als Umfangsmaße: -	MHF: +
		DIT: ++	DIT: ++		AHF: 0
		RFC: ++	RFC: /		MIF: +
		NOC: ?	NOC: ?		AIF: (+)
		CBO: ++	CBO: /		POF: +
		LCOM: 0	LCOM: /		COF: ++

Objektorientierte Maße:

- WMC (*Weighted Methods per Class*)
- DIT (*Depth of Inheritance Tree*)
- NOC (*Number Of Children*)
- CBO (*Coupling Between Object-classes*)
- RFC (*Response For a Class*)
- LCOM (*Lack of Cohesion on Methods*)
- MHF: *Method Hiding Factor*
- AHF: *Attribute Hiding Factor*
- MIF: *Method Inheritance Factor*
- AIF: *Attribute Inheritance Factor*
- POF: *Polymorphism Factor*
- COF: *Coupling Factor*

Software-Qualitätsmessung

Beliebte Hypothesen in Theorie und Praxis: Erkenntnisse II

- Einzelne einfache Komplexitätsmaße (z.B. McCabe zyklomatische Zahl) funktionieren nicht besser als Umfangsmaße (z.B. LOC)
- Spezifischere Komplexitätsmaße zeigen oft eine gute Prognosequalität für den Fehlergehalt

Schlussfolgerung:

- Eine geeignete Kombination von geeigneten Komplexitätsmaßen gestattet eine gezielte Identifikation der fehlerhaften Module

Software-Qualitätsmessung

Beliebte Hypothesen in Theorie und Praxis

	/Fenton, Ohlsson 00/	/Basili, et al. 96/	/Cartwright, Shepperd 00/	/Basili, Perricone 84/	/Abreu, Melo 96/
Modellbasierte (<i>Shlaer-Mellor</i>) Maße sind geeignet zur Fehlerprognose.	/	/	Events: ++	/	/
Modellbasierte Maße sind geeignet zur Prognose des Codeumfangs.	/	/	States: ++	/	/

Software-Qualitätsmessung

Beliebte Hypothesen in Theorie und Praxis: Erkenntnisse III

- Aus Softwareentwürfen können Messwerte gewonnen werden, die es gestatten, Codeumfänge und Fehlergehalte frühzeitig zu prognostizieren

- Statistische Verfahren zur Feststellung der Softwarezuverlässigkeit sind theoretisch fundiert und praktisch anwendbar.
- Einige plausibel erscheinende Hypothesen im Bereich der Qualitätsmessung sind empirisch falsifiziert worden, aber es existiert Evidenz, dass ...
 - ... sich Fehler in wenigen Modulen konzentrieren und ...
 - ... diese Module identifiziert werden können durch Messung und Anwendung von Prognosemodellen.
- Quantifizierung von Software-Qualität ist notwendig und machbar
- Dazu sind einschlägige Testunterstützungs- und Auswertewerkzeuge unverzichtbar

Literatur

- /Abreu, Melo 96/
Abreu F., Melo W., *Evaluating the Impact of Object-Oriented Design on Software Quality*, Proc. Metrics '96, pp. 90 - 99
- /Basili et al. 96/
Basili V., Briand L.C., Melo W.L., *A Validation of Object-Oriented Design Metrics as Quality Indicators*, IEEE Transactions on Software Engineering, Vol. 22, No. 10, October 1996, pp. 751-761
- /Basili, Perricone 84/
Basili V.R., Perricone B.T., *Software Errors and Complexity: An Empirical Investigation*, Communications of the ACM, Vol. 27, No. 1, January 1984, pp. 42-52
- /Cartwright, Shepperd 00/
Cartwright M., Shepperd M., *An Empirical Investigation of an Object-Oriented Software System*, IEEE Transactions on Software Engineering, Vol. 26, No. 8, August 2000, pp. 768-796
- /Fenton, Ohlsson 00/
Fenton N., Ohlsson N., *Quantitative Analysis of Faults and Failures in a Complex Software System*, IEEE Transactions on Software Engineering, Vol. 26, No. 8, August 2000, pp. 797-814

Das trägt alles dazu bei, dass Sie ...

