Testmetriken
für die Kalkulation der Testkosten und die Bewertung der Testleistung

von Harry M. Sneed (MPA)
Software Test Engineering Consultant
Budapest, Ungarn
h.sneed@axelero.hu

Abstract: Der Test von Software verbraucht immer mehr Ressourcen und beansprucht einen wachsenden Anteil der Projektlaufzeit. Dennoch bleibt der Nutzen dieser Aktivität umstritten. Es besteht ein großer Bedarf an Meßkriterien um den Nutzen des Testens zu rechtfertigen und die Kosten davon zu planen. Dieser kurzer Beitrag stellt eine Reihe Metriken vor die der Autor in seiner Tätigkeit als Testberater bei der SDS in Wien entwickelt und angewendet hat. Er soll als Anregung für eine wissenschaftlich Vertiefung dieser aus der Praxis gewonnenen Erfahrungswerte dienen. Das Endziel wäre eine Menge international genormter und anerkannter Testmetriken, aus der Softwareproduzenten selektieren können um ihre Testprojekte zu planen und ihren Testbetrieb bewerten zu können. Außerdem, hätten Wissenschaftler dadurch die Möglichkeit diverse Testansätze auf einer objektiven Basis miteinander vergleichen zu können.

Wie Lord Kelvin es bereits im 18. Jahrhundert treffend formulierte, solange wie wir etwas in Zahlen nicht erfaßt haben, haben wir es auch nicht wirklich verstanden. Mit der Messung beginnt das Verständnis. [1] Auf den Softwaretest übertragen, heißt das ohne Testmetriken kommen wir nicht weiter, weil wir die Auswirkung unserer Bemühungen nicht bemessen können. Es gebe keine Möglichkeit zum Vergleichen und daher auch keine Möglichkeiten zur Schätzung und zur Bewertung. Solange als Software Testen nicht meßbar ist, bleibt es in den Worten von Glenford Myers eine „Art“. [2]

Als Testberater bei der SDS GmbH in Wien war dieser Autor mit der Aufgabe konfrontiert, Testaufwand zu schätzen und Testleistungen zu bewerten. Zu diesem Zweck sah er sich gezwungen einige Metriken zu sowie die dazu gehörigen Werkzeuge entwickeln. In diesem ersten Aufsatz geht es darum, die Testmetriken kurz dar zu stellen. Sie werden in einem späteren Fachartikel ausführlich mit Praxisbeispielen erläutert.

Zunächst werden die Testmetriken in vier Kategorien aufgeteilt:
- Metriken für die Schätzung der Testkosten
- Metriken für die Bewertung der Testfälle
- Metriken für die Messung der Testüberdeckung
- Metriken für die Bewertung der Testeffektivität

Für die Schätzung der Testkosten werden Metriken gebraucht die den Umfang und die Komplexität des Tests voraussagen und diese in Testaufwand umsetzen. Für die Bewertung der Testfälle sind Metriken erforderlich die, die Quantität, Komplexität und Qualität der Testfälle in Maßzahlen erfassen. Für die Messung der Testüberdeckung braucht der Testbetrieb Koeffizienten für einen Soll/Ist Abgleich der Testprofile auf allen Teststufen. Für die Beurteilung der Testeffektivität braucht das Qualitätsmanagement schließlich Metriken die den Nutzen des tests zum Ausdruck bringen.

1. Metriken für die Schätzung der Testkosten

Für die Kalkulation der Testkosten sind drei Maße ausschlaggebend:
- Die Zahl der erforderlichen Testfälle
- Die Testbarkeit des Testobjektes
- Die bisherige Testproduktivität

Die Zahl der Testfälle ergibt sich aus der statischen Analyse des Konzeptes für den Systemtest und aus der statischen Analyse des Codes für den Modultest. Im Code wird die Mindestanzahl Pfade identifiziert, die ausgeführt werden müssen um alle Zweige zu erreichen. Jeder Pfad entspricht einem codebezogenen Testfall. Dies ist ein Ergebnis des Werkzeuges CPP- bzw. JAVAnal. Im Konzept werden alle Usecase Ausprägungen, Prüfregel, logische Bedingungen, Unterfunktionsaufrufe und Objektzustände gezählt. Die Schnittmenge jener Konzeptelemente mal zwei ergibt die Anzahl konzeptbezogener Testfälle. Dies ist ein Ergebnis des Werkzeuges CMFAnal.

Die Testbarkeit der Software ist ein komplexes Maß bestehend aus mehreren Teilmetriken. Dazu gehört Modularität, Datenbanknutzung, Schnittstellenbreite und Testpfadkomplexität. Die Modularität ist eine Frage der Modulgröße relativ zur Modulkopplung und zur Schnittstellenanzahl.

$$\text{Modularität} = 1 - \left\{ \frac{\text{Module x Modulschnittstellen x Modulbeziehungen}}{\text{Modulanweisungen}} \right\}$$

Die Datenbanknutzung ist eine einfache Relation zwischen den benutzen datenbanktabellen und den Testobjekten. Je mehr Datenbanken für den Test aufbereitet müssen, desto aufwendiger der Test.

$$\text{Datenbanknutzung} = 1 - \{\text{Testobjekte} + \frac{\text{Testobjekte}}{\text{Datenbanktabellen}}\}$$

Die Schnittstellenbreite mißt die Anzahl und die Länge der einzelnen Schnittstellen die, der Tester zu bedienen hat, relativ zur optimalen Anzahl und Länge. Die Schnittstellenlänge wird durch die Anzahl der Parameter, bzw. bei Oberflächen durch die Anzahl Oberflächenfelder, bestimmt.

$$\text{Schnittstellenbreite} = 1 - \{\text{Testobjekte x } \frac{\text{(Sollparameter/Sollschnittstellen)}}{\text{[Istparameter/Istschnittstellen)}}\}$$

Die Testpfadkomplexität folgt aus der Vereinigung zweier Relationen – die Anzahl Zweige relativ zur Anzahl Anweisungen und die Anzahl Prädikate relativ zur Anzahl Parameter. Denn je höher die Zweigdichte, um so großer die Pfadanzahl und je mehr Prädikate, bzw. Bedingungsvariabel, um so mehr Parameter müssen vom Tester gezielt gesetzt werden.

$$\text{Pfadkomplexität} = \frac{\text{Zweige}}{\text{Anweisungen}} \times \frac{\text{Prädikate}}{\text{Parameter}}$$

Der Testbarkeitsfaktor ist 2 minus die gewichtete Summe der vier Untermetriken dividiert durch 2. Das Ergebnis ist ein Multiplikationsfaktor im Bereich von 0,2 bis 2, d.h. die Testbarkeit des Testobjektes kann den Testaufwand verdoppeln oder bis zu 80% reduzieren.

Die bisherige Testproduktivität ist die Anzahl der Testfälle die ein Tester bisher durchziehen konnte und zwar von der Konzeption über die Spezifikation und Ausführung bis zur Auswertung. Es wäre auch möglich die Anzahl getesteter Fälle einfach durch die von der Testmannschaft gebuchten Personentage zu dividieren. Am Ende dürfte eine Zahl wie 4 oder 5,5 herauskommen.

Jetzt kommt es darauf an, nach der COCOMO-II Formel von Boehm [3], den wahrscheinlichen Testaufwand wie folgt zu kalkulieren:

$$\text{Testaufwand} = ST \times \{\frac{\text{Testfälle}}{\text{Testproduktivität}}\}^{SE} \times \text{Testbarkeitsfaktor}$$

Wobei ST = Systemtypfaktor (0,5:4)
Und SE = Skalierungsexponente (0,91:1,23)

2. Metriken für die Bewertung der Testfälle

Testfälle haben genau so wie Konzepte und Code eine Quantität, eine Komplexität und eine Qualität. Die Quantität ist aber hier mehr als nur die nackte Zahl der Testfälle. Sie ist die tatsächliche Anzahl Testfälle relativ zu der Anzahl Testfälle die laut statischer Analyse gebraucht werden. So betrachtet ist die Testquantität

$$\frac{\text{Ist\_Testfälle}}{\text{Soll\_Testfälle}}$$

Die Komplexität der Testfälle ist der Mittelwert vier einzelner Komplexitätsmetriken:

- Testdatenkomplexität
- Testdatendichte
- Testdatenvolumina
- Testintensität

Die Testdatenkomplexität ist im Sinne der Halstead Sprachmetrik das Verhältnis der Testparametertypen zur Anzahl Testparameter pro Schnittstelle [4].

$$\frac{\text{Parametertypen}}{\text{(Parameter/Schnittstellen)}}$$

Die Testdatendichte ist in Anlehnung an der Chapin Q_Metrik das Verhältnis der einzelnen Testdatenwerte zur Anzahl Testparameter [5]. Je mehr Werte die Parameter annehmen können, um so komplexer der Testfall.

$$1 - \frac{\text{Parameter}}{\text{Parameterwerte}}$$

Die Testdatenvolumina druckt sich in dem Verhältnis zwischen Testdaten und Testfällen aus. Je mehr Testparameter pro Testfall, desto hoher der Volumina.

$$1 - \frac{\text{Testfälle}}{\text{Testparameter}}$$

Die Testintensität ergibt sich aus der Anzahl der Zielfunktionen, die von den Testfällen angesteuert werden, relativ zur Anzahl Testfälle. Je mehr Testfälle gebraucht werden um die Funktionen zu erreichen, desto höher die Testintensität.

$$\frac{\text{Testfälle}}{\text{Zielfunktionen}}$$

Die Qualität der Testfälle ist der Mittelwert folgender Qualitätsmaße:

- Testfalleffizienz
- Testfallwiederverwendbarkeit
- Testfallüberdeckungsgrad
- Testfallkomformität

Die Testfalleffizenz läßt sich an der Anzahl der Zielfunktionen die von den Testfällen ausgelöst werden abmessen. Je mehr Funktionen ein Testfall erwischt, desto effizienter ist der Fall.

$$1 - \frac{\text{Testfälle}}{\text{Zielfunktionen}}$$

Die Testfallwiederverwendung ist eine Frage der Wiederholbarkeit der Testfälle. Nur automatisierte Testfälle, also jene die nicht von einem Menschen sondern von einem Werkzeug aus, gestartet werden, gelten als uneingeschränkt wiederholbar:

$$\frac{\text{Automatisierte Testfälle}}{\text{Testfälle}}$$

Die Testüberdeckung läßt sich zumindest formal an Hand der Anzahl spezifizierter, bzw. codierter Funktionen, relativ zur Anzahl Zielfunktionen die von den Testfällen explizit angesteuert werden, ermitteln.

$$\frac{\text{Zielfunktionen}}{\text{Spezifizierte Funktionen}}$$

Schließlich haben sich auch Testfälle nach einer Konvention zu richten. Bestimmte Attribute wie Zielfunktion, Version und Typ sollten richtig angegeben werden. Wenn nicht, handelt es sich um ein Mangel. Die Konformität ist demnach

$$\frac{\text{Ist-Testfallattribute}}{\text{Soll-Testfallattribute}}$$

Die Größe, Komplexität und Qualität der Testfälle sind von einem Testfallanalyse Tool, wie z.B. CTFAnal, automatisch zu messen. Benutzt werden diese Metriken um den Istzustand der Testdaten mit einem propagierten Sollzustand zu vergleichen.

3. Metriken für die Messung der Testüberdeckung

Testüberdeckungsmaße sind schon lange bekannt, auch wenn sie nur selten angewendet werden. Der bekannteste ist der C1 Maß für die Zweigüberdeckung des Codes [6]. Es geht aber nicht nur darum den Code zu überdecken. Konzepte und Benutzerhandbücher sind auch Kandidaten für die Überdeckungsmessung.

Konzepte bestehen aus zahlreichen testbaren Artifakte. In UML sind es die UseCases, die Assoziationen, die Interaktionen und die Objektzustandsübergänge. In CMF sind es die Vorgänge, Regel, Bedingungen und Objektzustände, die mehr oder weniger ausführlich spezifiziert sind. In dem Maße wie sie spezifiziert sind, gilt es sie zu testen.

$$\text{Konzeptüberdeckung} = \frac{\text{Getestete Konzeptelemente}}{\text{Spezifizierte Konzeptelemente}}$$

Auch das Benutzerhandbuch bietet eine Referenz an, gegen die getestet werden kann. Es obliegt den Testern die Funktionen daraus abzuleiten.

$$\text{Benutzerhandbuchüberdeckung} = \frac{\text{Getestete Funktionen}}{\text{Dokumentierte Funktionen}}$$

Das Ziel im Modultest ist von dem Code möglichst viele der Anweisungen, bzw. Zweige, auszuführen.

$$\text{Codeüberdeckung} = \frac{\text{Getestete Anweisungen, bzw. Zweige}}{\text{Anweisungen, bzw. Zweige}}$$

Schließlich sind die Testfälle selbst Kandidaten für die Überdeckungsmessung. Im Regressionstest werden immer nur einen Teil der Testfälle ausgeführt. Dieser Anteil entspricht der Testfallüberdeckung.

$$\text{Testfallüberdeckung} = \frac{\text{Ausgeführte Testfälle}}{\text{Testfälle}}$$

Es hängt von der Art des Systems und der Qualitätsanforderungen ab, welchen Grad an Testüberdeckung anzustreben ist.

4. Metriken für die Bewertung der Testeffektivität

Laut Spillner und Linz ist der Test eine Dienstleistung mit zwei Ziele

- Fehler zu finden (zumindest ehe der Anwender sie findet)
- Vertrauen im System aufzubauen [7]

Um die Erfüllung dieser beiden Ziele zu messen bieten sich zwei Metriken an, die beide erst im nachhinein berechnet werden können.

Die eine Metrik ist die Testqualitätsmetrik. Sie vergleicht die Anzahl nach deren Schwere gewichteter Fehler die im Testbetrieb aufgedeckt wurden mit der Anzahl der gewichteten Fehler die in der Produktion auftreten. Die Fehlermeldungen müssen dafür in zwei Kategorien geteilt werden – Fehlermeldungen von den eigenen Testern und Fehlermeldungen von den Anwendern. Beide zusammen ergeben die Summe aller Fehlermeldungen. Die Qualität des Tests ist

$$\frac{\text{Gewichtete Fehlermeldungen aus dem Testbetrieb}}{\text{Summe aller gewichteten Fehlermeldungen}}$$

Die andere Metrik mißt den Grad des Vertrauens im System. Vertrauen in einem System wächst mit einer sinkenden Fehlerrate trotz zunehmender Testaktivität gekoppelt mit einer steigenden Überdeckung. Die Anzahl ausgeführter Testfälle ist ein Zeichen zunehmender Testaktivität. Die Metrik für den Vertrauensgrad wäre demnach

$$1 - \frac{\text{Gewichtete Fehler}}{\{\text{ausgeführte Testfälle }\} \times \text{Überdeckungsgrad}}$$

Diese beiden Metriken erfassen letztendlich den Nutzen des Testbetriebes und beantworten die Frage ob es sich lohnt mehr in den Testbetrieb zu investieren. Man könnte natürlich noch versuchen den Schaden der durch Systemfehler in der Produktion verursacht wird, z.B. in ausgefallenen Arbeitszeiten und verlorenen Geschäfte, zu ermitteln, aber das würde den Rahmen dieser Studie sprengen. Hier geht es darum die direkten Kosten und Leistungen einer Testgruppe und deren Arbeitsergebnisse – die Testfälle ,Testprotokolle und Fehlermeldungen – messbar zu machen.

Literaturhinweise:

1] Hughes, B. Practical Software Measuremenr, McGraw-Hill, Maidenhead, G.B., 1999, s. 3
2] Myers, G.: The Art of Software Testing, John Wiley & Sons, New York, 1979, s. 11
3] Boehm, B. u.a.: Software Cost Estimation with COCOMO-II, Prentice-Hall, Englewood Cliffs, 1999, s.29
4] Halstead, M.: Elements of Software Science, Elsevier North-Holland, New York, 1978, s 11
5] Chapin, N.: "A Measure of Software Complexity" in Proc. of National Computer Conf., Chicago, 1979
6] Miller, E.: „Structured Techniques of Program Validation", Proc. of IEEE COMPCON, Chicago, 1974, s. 169
7] Spillner, A./ Linz, T.: Basiswissen Softwaretest, dpunkt Verlag, Heidelberg, 2003, s. 10