Testbarkeit in den frühen Projektphasen

Stefan Jungmayr
FernUniversität Hagen, Lehrgebiet Praktische Informatik III
Feithstraße 142, D 58084 Hagen
stefan.jungmayr@fernuni-hagen.de
http://www.informatik.fernuni-hagen.de/import/pi3/stefan/

Zusammenfassung:

Der Testaufwand in einem Entwicklungs- oder Wartungsprojekt hängt wesentlich vom Grad der Testbarkeit der Software ab. Versäumnisse hinsichtlich der Testbarkeit können in späteren Projektphasen nur schlecht oder mit größerem Aufwand nachgeholt werden. Daher ist es sinnvoll, bereits in den frühen Projektphasen systematisch Testbarkeit im Produkt zu verwirklichen. Dieser Beitrag behandelt Möglichkeiten der Bewertung und Verbesserung von Testbarkeit in den Entwicklungsaktivitäten Anforderungsermittlung und Analyse.

Schlüsselwörter: Analyse, Anforderung, Anforderungsermittlung, Anwendungsfall, Testbarkeit, Testen

1 Einleitung

Testbarkeit ist ein bestimmender Faktor für den Testaufwand in der Entwicklung und Wartung von Software. Eine gezielte Verwirklichung von Testbarkeit ist aber in der Praxis kaum anzutreffen, u.a. aufgrund des mangelnden Knowhows im Entwicklungsbereich [6].

1.1 Begriffsbestimmung

Unter *Testbarkeit* verstehen wir den Grad, zu dem ein Softwareentwicklungsprodukt in einem bestimmten Kontext das Testen ermöglicht.

Als *Softwareentwicklungsprodukte* bezeichnen wir die in der Anforderungsermittlung, in der Analyse, im Entwurf, in der Implementierung oder im Test erstellten Dokumente bzw. Software-Komponenten.

Testbarkeit ist keine absolute Größe eines Softwareentwicklungsprodukts sondern nur im *Kontext* von Qualitätsanforderungen, Testendekriterien, eingesetzten Testwerkzeugen etc. zu bewerten.

Testbarkeit beschreibt also den Zusammenhang zwischen Eigenschaften eines Softwareentwicklungsprodukts und dem Testaufwand bzw. dem Testergebnis im Kontext eines bestimmten Entwicklungsprojekts.

1.2 Testbarkeit im Software-Lebenszyklus

Ein hoher Grad an Testbarkeit ist in späten Projektphasen nur schlecht oder mit größerem Aufwand erzielbar, da die hierzu meist notwendigen Änderungen am Entwurf oder der Implementierung aufwendig oder im Projektdruck nicht möglich sind. Daher ist es sinnvoll, bereits in den frühen Projektphasen systematisch Testbarkeit im Produkt zu verwirklichen. Die frühen Projektphasen sind hierbei ein Synonym für die Aktivitäten *Anforderungsermittlung* und *Analyse*.

2 Testbarkeit in der Anforderungsermittlung

In der Anforderungsermittlung werden die Erwartungen des Kunden an die Software als einzelne Sätze oder in Form von Anwendungsfällen (use cases) formuliert und niedergeschrieben. Folgende produkt- und prozeßbezogenen Problembereiche bestimmen die Testbarkeit der Anforderungen in ihrer Gesamtheit:

- Überflüssige Anforderungen - sie steigern unnötig die Größe und Komplexität der zu testenden Anwendung.
- Unklare Anforderungen, z.B. verursacht durch unverständliche, unvollständige oder nicht eindeutige Formulierungen - sie verhindern die Definition von Testfällen.
- Untestbare Anforderungen, z.B. bedingt durch nicht-quantitative Formulierungen.
- Schwierig zu testende Anforderungen die z.B. mit nicht-deterministischem Verhalten oder komplexen Berechnungen verbunden sind.
- Fehlende *Testbarkeitsanforderungen*, z.B. betreffend Test-Schnittstellen und Log-Funktionalität.
- Mangelnde Aktualität und Stabilität von Anforderungen.

2.1 Bewertung der Testbarkeit

Zunächst sollten die Anforderungen priorisiert werden (falls dies noch nicht durchgeführt wurde). So kann u.a. die Bewertung der Testbarkeit auf die wichtigen Anforderungen beschränkt werden, wenn die Bewertung *aller* Anforderungen aus Zeitgründen nicht möglich ist.

Zur Bewertung der Testbarkeit in der Anforderungsermittlung stehen folgende Techniken und Hilfsmittel zur Verfügung:

- Reviews - basierend auf Checklisten [5] oder Perspektiven [7].
- Testplanerstellung während der Anforderungsermittlung.
- Entwurf der Abnahme- bzw. Systemtestfälle.
- Sprachanalyse (automatisiert) [2][3].

Alle wichtigen aber unklaren bzw. untestbaren Anforderungen müssen entweder neu formuliert oder verworfen werden. Die Anforderungen können entsprechend dem

Grad ihrer Testbarkeit Kategorien zugeordnet werden wie z.B. A - "nicht testbar", B - "schwer testbar", C - "leicht testbar".

2.2 Verbesserung der Testbarkeit

Die Verbesserung der Testbarkeit in der Anforderungsermittlung sollte sich besonders auf jene Anforderungen konzentrieren, die sowohl für den Projekterfolg kritisch als auch schwer bzw. nicht testbar sind. In der hier vorgestellten Testbarkeitsmatrix (Abb. 1) befinden sich diese Anforderungen im oberen, rechten Quadrant.

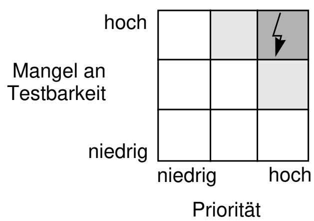

Abb. 1: Testbarkeitsmatrix

Die Verbesserung der Testbarkeit bezieht sich auf die in Abschnitt 2.1 genannten Problembereiche. Im Fall von nicht-quantitativen Formulierungen (wie z.B. "rasche Antwortzeit" oder "hohe Benutzbarkeit") bedeutet dies z.B. die Neuformulierung der Anforderung unter Angabe von exakten Maßen, Minima, Maxima oder statistischen Maßen. Fehlende Testbarkeitsanforderungen können z.B. durch die Aufnahme der Akteure "Wartungspersonal" und "Testwerkzeug" sowie durch die Einführung von Anwendungsfällen wie "Wartungstest durchführen" und "Fehler lokalisieren" berücksichtigt werden.

3 Testbarkeit in der Analyse

In der Aktivität Analyse werden die Anforderungen in eine interne Sicht der Anwendung übersetzt. Ergebnisse der Analyse sind u.a. Analyse-Klassendiagramme, Analyse-Pakete und Szenarien (vergl. use-case realizations) [4]. Folgende Problembereiche bestimmen die Testbarkeit dieser Entwicklungsprodukte:

- Fehlende Verfolgbarkeit zurück zu den Anforderungen - die Analyseergebnisse sind nur eingeschränkt zur Testfalldefinition verwendbar.
- Fehlende Berücksichtigung der Testbarkeitsanforderungen - dies führt zu mangelnder Testunterstützung während des Testens.
- Komplexe Abläufe von Anwendungsfällen - diese erhöhen die Anzahl der Testfälle.
- Starke Abhängigkeiten zwischen Anwendungsfällen - diese führen zu Abhängigkeiten in den Testfällen.
- Undokumentierte Abhängigkeiten zwischen Anwendungsfällen - diese erhöhen die Gefahr, wesentliche Testfälle zu übersehen.
- Starke Kopplung zwischen den Analyse-Klassen sowie starke Abhängigkeiten zwischen den Analyse-Paketen - diese erhöhen die Testkomplexität und führen zu Abhängigkeiten in den Testfällen.

3.1 Bewertung der Testbarkeit

Auch in der Analyse sollte die Testbarkeit zumindest für die wichtigen Anwendungsfälle bestimmt werden. Grundlage hierfür sind die Abhängigkeiten zwischen den Anwendungsfällen die z.B. durch lesende oder schreibende Zugriffe auf Analyse-Klassen verursacht werden (siehe *use case object model matrix* [1]). Weiters ist die Umsetzung der Testbarkeitsanforderungen aus der Anforderungsermittlung zu bewerten.

3.2 Verbesserung der Testbarkeit

Abhängigkeiten zwischen Anwendungsfällen können u.a. durch eine Abschwächung ihrer Vorbedingungen reduziert werden. Dies läßt sich z.B. durch eine Abschwächung von Assoziations-Multiplizitäten im Analyse-Klassenmodell erreichen. Die Funktionalität der Anwendungsfälle sollte so auf Operationen von Analyse-Klassen abgebildet werden, dass die Abhängigkeiten zwischen den Analyse-Paketen reduziert werden.

4 Fazit

Mit der zunehmenden Komplexität und Kritikalität von Softwareanwendungen gewinnt Testbarkeit stark an Bedeutung und muss bereits in den frühen Projektphasen systematisch verwirklicht werden. Dazu dient u.a. die Konzentration auf Anforderungen von hoher Priorität, die Berücksichtigung von Testaktivitäten als eigene Anwendungsfälle sowie die Bewertung und Verbesserung der Testbarkeit von Anwendungsfällen im Sinne ihrer wechselseitigen Abhängigkeiten. Voraussetzung für einen systematischen Ansatz zur Verbesserung der Testbarkeit ist die geeignete Unterstützung der Entwickler und Tester durch das Management im Rahmen von Trainingsmöglichkeiten, Anpassung der Entwicklungsprozesse (falls notwendig) sowie entsprechender Verteilung der Resourcen.

5 Quellenverzeichnis

[1] Armour Frank und Miller Granville. *Advanced use case modeling: software systems*. Addison Wesley, 2000. ISBN 0-201-61592-4.

[2] Automated requirement measurement (ARM) tool, Software Assurance Technology Center (SATC), NASA Goddard Space Flight Center. http://satc.gsfc.nasa.gov/tools/arm/.

[3] Fabbrini F., Fusani M., Gnesi S., Lami G. An Automatic Quality Evaluation for Natural Language Requirements. *Seventh International Workshop on Requirements Engineering: Foundation for Software Quality*, Interlaken, Schweitz, 4.-5. Juni 2001.

[4] Jacobson Ivar, Booch Grady and Rumbaugh James. *The Unified Software Development Process*. Addison-Wesley, 1999. ISBN 0-201-57169-2.

[5] Jungmayr Stefan. Reviewing Software Artifacts for Testability. *Proceedings of EuroSTAR99*, Barcelona, Spanien, 8.-12. November 1999.

[6] Linz Tilo. Berufsbild "Tester" - Wünsche und Anregungen zur Ausbildung. *Softwaretechnik-Trends* 21:1, Februar 2001.

[7] Laitenberger Oliver und DeBaud Jean-Marc. Perspective-based Reading of Code Documents at Robert Bosch GmbH. *Information and Software Technology*, vol. 39, November 1997.