Testbarkeitsanforderungen an die Software

Stefan Jungmayr, 71254 Ditzingen
www.testbarkeit.de

Abstract: Testprobleme und die Suche nach Software-Fehlern verschlingen einen wesentlichen Anteil des Projektbudgets. Um bei knapp kalkulierten Projektressourcen noch handlungsfähig zu bleiben sollte daher dieser Anteil durch konstruktive Maßnahmen zur Verbesserung der Testbarkeit unter Kontrolle gehalten werden. Dieser Artikel beschreibt zahlreiche funktionale und nicht-funktionale Anforderungen für die Testbarkeit von Software-Systemen aus der Sicht unterschiedlicher Projektrollen.

1 Einleitung

Testbudgets wachsen in der Regel nicht mit der Komplexität und Kritikalität der Software-Systeme 1-zu-1 mit. Steigender Zeitdruck in der Software-Entwicklung tut sein übriges, dass der Test und die anschließende Fehlersuche immer herausfordernder werden. Daher sollte ein Software-System den Test seiner Funktionalität und die Fehlersuche unterstützen - tut es aber häufig nicht! Im Gegenteil: der allzu oft anzutreffende Mangel an Testbarkeit gefährdet den Projekterfolg, da ein beachtlicher Anteil der Entwicklungszeit durch Testprobleme und Fehlersuche aufgezerrt wird[1].

Was ist zu tun? Wie kann eine ausreichende Testbarkeit eines Software-Systems sichergestellt werden? Indem die Testbarkeitsanforderungen an die Software explizit formuliert und im Softwareprozess berücksichtigt werden. Dies setzt voraus, dass die Anforderungsmanager die Stakeholder der Testbarkeit kennen und diese Art von Anforderungen gezielt erheben können.

Leider ist das Wissen um kritische Aspekte der Testbarkeit nur unter Testern vorhanden, kaum aber bei Entwicklern, Designern, Systemanalytikern und Projektmanagern. Erschwerend kommt hinzu, dass die Tester ihrerseits wenige bis keine konkreten Vorschläge zur Verbesserung der Testbarkeit während der Entwurfsphase einbringen – sei es aus Mangel an Entwurfswissen oder sei es aus fehlender Einbeziehung in den Anforderungs- bzw. Entwurfsprozess.

Dieser Artikel beschreibt funktionale und nicht-funktionale Testbarkeitsanforderungen an die Software mit der Intention, dass Sie sich als Tester, Entwickler, Designer oder Systemanalytiker aus diesem Fundus bedienen können, um konkrete Testbarkeitsanforderungen in Ihrem Software-Projekt zu formulieren.

[1] In Umfragen des Autors unter ca. 50 Personen im Testumfeld wurde der durch Testprobleme verursachte Anteil am Testaufwand mit 30% bis 50% geschätzt.

2 Anforderungen von Testern

Welche Testbarkeitsanforderungen stellen Tester an ein Software-Produkt? Folgende Punkte sind aus Sicht des Testens wichtig:

- Verständlichkeit: Der Code sollte einfach verständlich sein und notwendige Kommentare enthalten.[2]
- Angemessene Komplexität: Die Komplexität der Software soll nur so groß sein, wie es der Bewältigung des Anwendungsproblems entspricht, nicht größer.
- Kontrollierbarkeit (Steuerbarkeit): Der für den Test einer Komponente erforderliche Test-Anfangszustand soll sich einfach herstellen lassen.
- Beobachtbarkeit: Das zu testende System soll ausreichend Rückmeldung über Zwischen- und Endergebnisse bzw. den Erfolg einer (Benutzer-)Aktion geben.
- Isolierbarkeit: Ist ein Modultest vorgesehen, so sollen die Module isoliert, d.h. abgetrennt vom Rest des Systems getestet werden können. Dies bedingt z.B., dass dem Einsatz von Stub- oder Mock-Objekten keine Hindernisse wie fest-verdrahtete Abhängigkeiten o.Ä. entgegenstehen.
- Automatisierbarkeit: Die Software soll den Einsatz von Testwerkzeugen unterstützen. Beispielsweise sollen GUI-Elemente durch ein Testwerkzeug eindeutig identifizierbar sein oder das Testwerkzeug benachrichtigt werden, wenn ein GUI-Fenster vollständig aufgebaut wurde.
- Robustheit (der Software): Wenn ein Testfall aufgrund eines schweren Softwarefehlers (z.B. Auftritt einer Exception) fehlschlägt, so sollen nachfolgende Testfälle trotzdem ausführbar sein (insbesondere bei automatisierten Testläufen).
- Reproduzierbarkeit: Testläufe und -Ergebnisse sollen reproduzierbar sein, um den Regressionstest sowie die Fehlersuche zu unterstützen.
- Lokalisierbarkeit: Fehler sollen sich möglichst einfach einer Komponente zuordnen lassen.
- Anonymisierbarkeit: Personenbezogene oder andere sensible Daten müssen anonymisierbar sein, um sie zur Reproduzierung von Fehlern in eine Testumgebung übertragen zu können (insbesondere im Falle von Test-Outsourcing).

[2] Dies unterstützt beim Review, beim Entwurf von Whitebox-Testfällen und bei der Fehlerlokalisierung.

Die oben genannten Testbarkeitsanforderungen sind für den erfahrenen Tester keine Überraschung. Trotzdem ist es noch keine gängige Praxis, diese Anforderungen auch in Anforderungsdokumenten zu verankern bzw. angemessen zu berücksichtigen.

3 Anforderungen von Entwicklern

Entwickler müssen Fehlermeldungen interpretieren sowie Fehler möglichst rasch lokalisieren und beheben können. Entsprechend fallen die Anforderungen der Entwickler an die Testbarkeit aus.

- Ausgabe von Fehlermeldungen: Bei Fehlern muss eine entsprechende Meldung (zumindest im Log) ausgegeben werden. Ein „Verschlucken“ von Exceptions ist nicht zulässig.
- Nachvollziehbare Fehlermeldungen: Fehlermeldungen des Systems, die z.B. von Benutzern an den Entwickler gemeldet werden, sollen eindeutig und verständlich formuliert sein sowie (falls möglich) Aufschluss über die Fehlerursache geben.
- Nachvollziehbarer Systemzustand: Die vor oder beim Eintritt des Fehlers vorliegende Systemkonfiguration soll durch den Entwickler nachvollziehbar sein. Entsprechende Daten sollen der Fehlermeldung beigefügt bzw. vom Entwickler nach Eingang der Fehlermeldung abfragbar sein. Die Systemkonfiguration bezieht sich u.a. auf den Versionsstand der betroffenen Komponenten. Der Systemzustand bezieht sich auf zentrale Datenstrukturen und Datenwerte (ggf. auch Zähler- und Zeitwerte), welche für das Reproduzieren des Fehlers relevant sind.
- Konfigurierbares Logging: Die Log-Informationen sollen zur Laufzeit konfigurierbar sein, d.h. der Umfang bzw. die Art der Log-Ausgaben soll an den Informationsbedarf angepasst werden können.
- Selbstdefinierte Abfragen: Der Entwickler soll in der Lage sein, nicht nur vordefinierte Log-Informationen einzusehen, sondern auch zur Laufzeit Systeminformationen nach Bedarf abzufragen und zu filtern.
- Fehlerstatistiken: Für Rückschlüsse auf die Fehlerursache sollen dem Entwickler Statistiken über Fehler(arten) zur Verfügung stehen.
- Trennung von Fehlermanagement und Geschäftslogik: Code-Anteile für Fehlermanagement und Geschäftslogik sollen sauber voneinander getrennt werden. Auch hier gilt "Separation of Concerns".
- Einfache Fehlermanagement-Logik: Zur Fehlerbehandlung soll möglichst wenig zusätzliche Programmlogik erforderlich sein, die durch Teilausfälle des Systems möglichst wenig beeinträchtigt werden soll.
- Offene System-Architektur: Die Code-Infrastruktur zur Behandlung von Fehlern sollte auf offenen Standards und Rahmenwerken aufsetzen.

4 Anforderungen von Administratoren

System-Administratoren müssen den Betrieb eines Software-Systems erhalten bzw. wiederherstellen. Ein System-Administrator hat in der Regel keine detaillierten Kenntnisse über die technischen Details des Systems. Treten im Betrieb Fehler auf, so soll der Administrator durch gute Fehlermeldungen bei der Lokalisierung und Behandelung von Fehlern unterstützt werden. Der Administrator erwartet somit u.a. Informationen zu den folgenden Punkten [2]:

- Fehlerart: Eindeutige Fehlernummer sowie Fehlerattribute zur Charakterisierung des Fehlers.
- Fehlerort: Angabe des Moduls bzw. Komponente, welche die Fehlermeldung abgesetzt hat.
- Fehlerkontext: Welches Datum sollte transformiert werden? In welchem Anwendungsfall ist der Fehler aufgetreten? Mit welchen Benutzerdaten (Datenkontext) ist der Fehler aufgetreten?
- Fehlerursache: Hinweise auf die mögliche Fehlerursache, sofern verfügbar.
- Fehlerkritikalität: Wie kritisch ist der Fehler für den Systembetrieb?

Genauso wichtig sind für den System-Administrator Informationen darüber, wie mit Fehlern umzugehen ist:

- Eskalation: Welche Eskalationsstufe ist zu wählen?
- Benachrichtigungen: Wer soll über was wie (per E-mail, SMS, etc.) benachrichtigt werden?
- Support-Hinweise: Wer kann bei Fehlern helfen? Wer hat welchen Teil der SW installiert? Wer ist Produktverantwortlicher?

Falls sich der Fehler nicht sofort beheben lässt, soll das System es ermöglichen, zumindest einen eingeschränkten Betrieb aufrechtzuerhalten und dem Administrator entsprechend Hilfestellung dazu geben:

- Information über Handlungsspielraum: Die Fehlermeldung soll Informationen über mögliche Ausweichstrategien (Workarounds) enthalten.
- Lastverteilung: Bei verteilten Systemen soll es möglich sein, Aufgaben des betroffenen Subsystems auf andere Subsysteme zu verteilen.
- Rekonfiguration: Eine Rekonfiguration des Systems soll möglich sein, ggf. im laufenden Betrieb (d.h. ohne Neustart). Fehlerhafte Subsysteme sollen einzeln (de)aktivierbar sein
- Automatische Fehlerbehandlung: Falls sinnvoll und möglich, soll das System selbst Maßnahmen zur Fehlerbehandlung einleiten, z.B. Datenbank-Verbindungen neu aufbauen, Module re-initialisieren, oder Problembehaftete Module beim nächsten Neustart deaktivieren.
- Konfigurierbares Logging: Auch der Administrator soll zur Fehlerlokalisierung den Log-Level zur Laufzeit anpassen können. Ggf. soll das Softwaresystem selbst den Log-Level für eine Komponente automa-

tisch anpassen, wenn in dieser Komponente häufig Fehler auftreten.

- Leitstellen-Darstellung: Status- und Fehlerinformationen sollen in einer Leitstelle graphisch und übersichtlich dargestellt werden. Gegebenenfalls soll das System die Schnittstelle einer vorhandenen Leitstelle beschicken. Detailinformationen zu Fehlern und den betroffenen Modulen sollen auf Anfrage hin vom System bereitgestellt werden.
- Systemkennzahlen: Wichtige Systemkennzahlen zur Unterscheidung von normalem, kritischem und fehlerhaftem Verhalten sollen dem Administrator zur Verfügung stehen. Dazu zählen die CPU-Auslastung und Antwortzeiten, die Datenmenge an Schnittstellen (in/out) sowie der verfügbarer Festplattenspeicher. Ebenso soll der Administrator Informationen darüber erhalten, ob noch alle Betriebssystem- bzw. Anwendungsprozesse aktiv sind (z.B. durch Ping-Tests bei Netzwerkkommunikation oder Watch-Dogs in Multi-Threading-Umgebungen).
- Alarme: Nicht zuletzt sollte der System-Administrator bei Auftreten von Fehlern oder Grenzwertüberschreitungen alarmiert werden.

5 Anforderungen von Benutzern

Auch die Benutzer haben Anforderungen bzgl. der Fehlerhandhabung. Zunächst muss der Benutzer bei der Fehlererkennung unterstützt werden:

- zugänglicher Systemzustand: Der aktuelle Zustand des Systems (d.h. wichtige Kennzahlen) sollte für den Benutzer abfragbar sein.
- verständlicher Systemzustand & Fehlermeldung: Angaben zum Systemzustand bzw. zu Fehlern sollten auch vom Benutzer einfach und unmissverständlich interpretiert werden können. Meldungen, die sich auf System-Interna beziehen, kryptische Detailangaben oder Fachbegriffe sind zu vermeiden.
- Unterscheidung Benutzerfehler/Systemfehler: Für den Benutzer soll aus der Fehlermeldung ersichtlich sein, ob das Problem durch eine Fehlbedienung (und wenn ja – durch welche) hervorgerufen wurde.
- Selbst-Test: Zur Prüfung der Integrität des Systems sollten Selbst-Tests verfügbar sein.
- frühzeitige Fehlererkennung: Fehlermeldungen sollten möglichst rasch den Benutzer erreichen, sodass dieser noch ausreichend Zeit für die Reaktion auf die Fehlermeldung hat, bevor das System ausfällt. D.h. bei Auftreten einer Fehlermeldung sollte gleichzeitig eine ggf. reduzierte Systemfunktionalität so lange wie möglich zur Verfügung stehen. Gleichzeitig unterstützt eine frühzeitige Fehlerwarnung die Fehlerlokalisierung.
- Fehlerprognosen: Falls technisch möglich sind daher auch Fehlerprognosen für den Benutzer wichtig. Fehlerprognosen können z.B. für Netzwerke (Überlast) oder den Zugriff auf Peripherie-Geräte erstellt werden.

Neben der Fehlererkennung soll der Benutzer auch bei der Meldung von Fehlern an den Hersteller bzw. Entwickler unterstützt werden:

- Unterstützung bei Fehlerbeschreibung: Eine vom Softwaresystem dargestellte Fehlermeldung sowie der Systemzustand zum Zeitpunkt dieser Fehlermeldung soll vom Benutzer einfach oder automatisiert dokumentierbar sein.
- Weiterleitung: Eine automatische Weiterleitung der Fehlermeldung an den System-Hersteller / Entwickler vereinfacht für den Benutzer die Meldung von Fehlern. Dabei ist der Datenschutz zu berücksichtigen und das Einverständnis des Benutzers zur Übertragung der Daten einzuholen.
- Diagnose-Qualität: Die Güte der Diagnose ist für den Benutzer entscheidend. Weder sollen aufgetretene Fehler nicht erkannt werden (Diagnosefehler 1. Klasse), noch korrektes Systemverhalten als Fehler interpretiert werden (Diagnosefehler 2. Klasse) [1].
- Vorwarnzeit: Wichtig für den Benutzer ist die verbleibende Zeitspanne zwischen der Meldung des Fehlers und dem Eintritt des Systemausfalls bzw. der eingeschränkten Funktionalität.

6 Umsetzung der Anforderungen

Eine gute aber selten anzutreffende Praxis ist das Review der Systemanforderungen durch die Tester. Dies kann allerdings nur dazu beitragen, dass Anforderungen testbar formuliert werden – Testbarkeitsanforderungen werden dadurch kaum Einzug in die Anforderungsdokumente halten. Daher sind diese Anforderungen systematisch zu erfassen und zu dokumentieren.

Eine gute Quelle für die Erfassung von Testbarkeitsanforderungen sind Fehlerdatenbanken, die um die Fehlerkategorie „Testproblem“ erweitert werden können. Ebenso sollten Mängel bzw. positive Ansätze betreffend der Testbarkeit in Lessons-Learned-Sitzungen erfasst und damit in Folgeprojekten berücksichtigt werden.

7 Zusammenfassung und Ausblick

Dieser Artikel hat eine Vielzahl von funktionalen und nicht-funktionalen Testbarkeitsanforderungen von Software dargestellt. Das allgemeine Wissen über Anforderungen bzgl. Testbarkeit sollte allen an der Entwicklung eines komplexen Softwaresystems beteiligten Rollen zugänglich sein, damit im Projekt Konkretisierungen dieser Anforderungen explizit formuliert und umgesetzt werden können. Das Projektmanagement ist dann gefordert, den richtigen Trade-Off zwischen Testbarkeit und anderen Anforderungen z.B. betreffend Performance und Datenschutz sicherzustellen.

Eine solide Grundlage für Managemententscheidungen fehlt allerdings noch: zuverlässige Daten über jenen Aufwand, der durch Testprobleme und unnötig aufwendige Fehlersuche verursacht wird (und was davon durch konstruktive Maßnahmen eingespart werden kann). Dieser Aufwand ist derzeit z.B. in den Implementierungskosten versteckt. Ein offener Punkt ist daher die Ent-

wicklung eines Ansatzes, wie dieser versteckte Aufwand beziffert werden kann.

8 Danksagung

Vielen Dank an Andreas Schönknecht und Matthias Hamburg für ihre Review-Kommentare zu diesem Artikel.

9 Referenzen

[1] G. Vachtsevanos, F. L. Lewis, M. Roemer et.al. Intelligent Fault Diagnosis and Prognosis for Engineering Systems. Wiley, 2006. ISBN 0-471-72999-X

[2] Lars Wunderlich. Managing Java. entwickler press, 2006. ISBN 3-939084-13-1.