Testen und JUnit

Stephan Karrer

Testen von Software

- Nach **International Software Testing Qualifications Board (ISTQB)**:
  Der Prozess, der aus allen Aktivitäten des Lebenszyklus besteht (sowohl statisch als auch dynamisch), die sich mit der Planung, Vorbereitung und Bewertung eines Softwareprodukts und dazugehöriger Arbeitsergebnisse befassen. Ziel des Prozesses ist sicherzustellen, dass diese allen festgelegten Anforderungen genügen, dass sie ihren Zweck erfüllen, und etwaige Fehlerzustände zu finden. [ISTQB Glossar 2011]

- Daraus folgt:
  - Statische Analysen, welche das Testobjekt nicht ausführen, sondern in Bezug auf Spezifikation, Procammcode als Text, ... analysieren.
  - Dynamische Tests, welche „ausführbare“ Testobjekte, sprich Programmcode, erfordern und diese unter kontrollierten Bedingungen mit dem Ziel ausführen, Anforderungserfüllung sicherzustellen bzw. Fehler zu finden.
  - Diese dynamischen Tests wollen wir im folgenden näher betrachten!

Dynamische Tests 1

- Dynamische Tests führen das Testobjekt mit bestimmten Eingabewerten unter bestimmten Bedingungen aus und vergleichen dessen tatsächliche Ergebnisse mit den erwarteten Ergebnissen.

- Eingabewerte inklusive der erwarteten Ergebnisse etc. für dynamische Tests werden zusammengefasst als Testfall bezeichnet und potentiell aus Dokumentation, Anforderungs- und Entwurfsspezifikationen abgeleitet.
  - Bei systematischer Ableitung der Testfälle spricht man auch von Teststrategie.

- Man unterscheidet:
  - Funktionale Tests: Prüfung, ob das Testobjekt die funktionalen Vorgaben erfüllt.
  - Nicht-funktionale Tests: Erfüllung anderer Qualitätskriterien wie z. B. Performanz oder Benutzbarkeit wird geprüft.
  - Black-Box: Keine Berücksichtigung der Implementierung, nur die spezifizierten Anforderungen werden geprüft.
  - White-Box: Man überprüft den Kontroll/Datenfluß innerhalb der Implementierung (z.B. Finden von nicht erreichbaren Code).

Dynamische Tests 2

- Erschöpfende Tests sind wegen der Fülle der möglichen Eingaben und Programmzustände selbst bei kleineren Programmen in der Regel nicht möglich.

- **Testen als stichprobenartiges Verfahren kann somit normalerweise nur die Anwesenheit von Fehlern aufzeigen, nicht aber deren Abwesenheit beweisen!!**

- Durch geschickte Auswahl von Testfällen und -daten wird das gemildert, z.B:

  - Äquivalenzklassentest:
    Die Menge der Werte und Bedingungen wird in Klassen aufgeteilt, bei denen ein gleiches Verhalten des Testobjekts erwartet wird. Dadurch muss je Äquivalenzklasse nur für wenige Stellvertreter tatsächlich getestet werden.

  - Grenzwertanalyse:
    Erfahrungsgemäß führt die nicht ausreichende Berücksichtigung der Grenzen von Wertebereichen immer wieder zu Fehlern. Man betrachtet nun genau die Grenzen der entsprechenden Wertebereiche.

Wie gut ist der Test

- Zur Planung und Steuerung des Testprozesses sind quantifizierbare Informationen über die geforderte bzw. erreichte Testgüte erforderlich.

- Als Testüberdeckung oder auch Testabdeckung bezeichnet man entsprechende Metriken:
  - Testfallbasierte Metriken,
    wie z.B. Anzahl ausgeführte Testfälle / Anzahl spezifizierter Testfälle
  - Testbasis- und testobjektbasierte Metriken,
    wie z.B. Anzahl Testfälle pro Anwendungsfall oder Anzahl Testfälle pro Funktion
  - Fehlerbasierte Metriken,

    wie z.B. Anzahl Fehler / Anzahl Programmzeilen
  - Kosten- und aufwandsbasierte Metriken,
    wie z.B. angefallene Anzahl Personentage für den Test

- Das kann durch Mutationstests ergänzt werden:
  - Gezieltes Einpflanzen von Fehlerzuständen in die Testobjekte, z. B. durch Änderung arithmetischer oder logischer Operatoren, d.h. Schaffung sogenannter Mutanten.
  - Anschließend Messung, welche und wieviele Mutanten durch die Tests aufgedeckt werden.

Teststufen

- Folgende Stufen werden üblicherweise unterschieden (und werden bei Bedarf noch weiter verfeinert):
  - Modul- bzw. Komponententest (unit test):
    - Einzelne elementare Programmbausteine werden isoliert getestet, oft vom jeweiligen Entwickler selbst.
    - Der Tester hat in der Regel Zugang zum Quellcode und somit sind sowohl Black-Box- als auch White-Box-Tests anwendbar.
  - Integrationstest:
    - Sind mehrere Programmbausteine im Zusammenspiel korrekt spezifiziert und implementiert.
    - Das kann sich sowohl auf kleine Einheiten, z.B. Klassen, als auch große Teile der Applikation beziehen.
    - Die Ausführung der größeren Integrationstests erfolgt in der Regel in speziellen Testumgebungen und wird durch entsprechende Werkzeuge zur Automatisierung unterstützt.
  - Systemtest
    - Der Systemtest soll die Funktionsfähigkeit des gesamten Systems entsprechend der Anforderungsspezifikation sicherstellen. Zum Systemtest zählen weiterhin Performanz-, Last-, Stress- und Robustheitstests.

Unit-Tests mit JUnit

"Any program feature without an automated test doesn't exist."
Kent Beck, Extreme Programming Explained: Embrace Change (Reading, MA: Addison-Wesley, 1999)

- Unit- und Integrationstests können durch den Entwickler mit Hilfe des freien Test-Frameworks erfolgen
- Ist heute (hoffentlich) übliche Praxis in der Software-Entwicklung mit Java
- Aktuell 2 syntaktisch etwas unterschiedliche Versionen eingesetzt:
  - JUnit 4
  - JUnit 5, kurz Jupiter genannt
- Wird in der Regel direkt durch die IDE bzw. Build-Tool unterstützt.
- Freies Konkurrenz-Framework TestNG hat sich weniger stark durchgesetzt
- Ergänzt durch weitere Bibliotheken, z.B. für Mocking