Individuelle IT-Konzepte
und Softwarelösungen

Gesellschaft für
Informations- und
Kommunikationssysteme mbH

Motivation und Überblick

iks-Thementag : „Wer testet, ist feige“

24.06.2009

Autor:
Christoph Schmidt-Casdorff
Carsten Schädel

Agenda

- **Einführung**
- **Auf welcher Ebene wird getestet – *testing level***
- **Was wird getestet– *testing categories***
- **Wie wird getestet – *testing methods and tools***
- **Alles zusammen – *test execution***

Fehler in Software

- **Mythos: Software ist fehlerfrei**
  - ist nicht nachweisbar
  - Testen findet Fehler, weist nicht deren Abwesenheit nach
- **Mythos: Qualität meint a priori für Jeden dasselbe**
  - es gibt unterschiedliche Dimensionen
    - Funktionalität, Performance, …
  - Qualität muss festgelegt werden
  - Qualität ist keine Eigenschaft, sondern eine Metrik
- **Mythos: Qualität ‚geschieht' von selbst**
  - das Ziel ‚*Qualität*' muss Bestandteil des Entwicklungsprozess sein

Fehler in Software

- **Je später ein Fehler entdeckt wird, desto teurer wird dessen Behebung**

		Time Detected				
		Requirements	Architecture	Construction	System Test	Post-Release
Time Introduced	Requirements	1×	3×	5–10×	10×	10–100×
	Architecture	-	1×	10×	15×	25–100×
	Construction	-	-	1×	10×	10–25×

Quelle: [McConnell2004]

- **63 % der Fehler stammen aus Analyse/Design [Perry2000]**
  - viele Fehler sind früh zu erkennen

Fehler in Software: Quintessenz

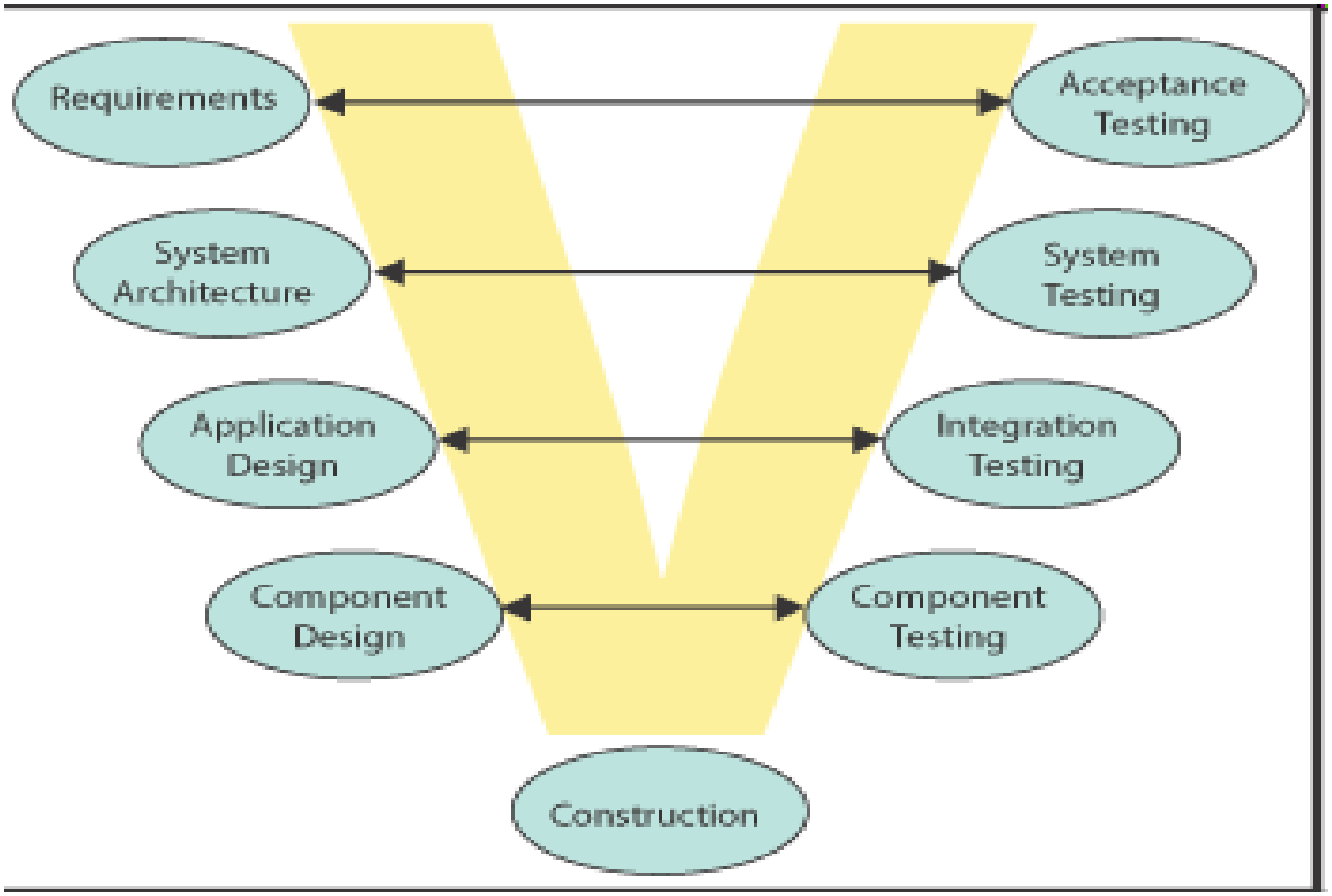
- **Es muss getestet werden**
- **… so früh wie möglich**
- **Qualitätskriterien müssen festgelegt werden**
- **Qualität ist Bestandteil des Entwicklungsprozesses**

Wo sind wir

- Einführung
- **Auf welcher Ebene wird getestet – *testing level***
- Was wird getestet– *testing categories*
- Wie wird getestet – *testing methods and tools*
- Alles zusammen – *test execution*

Testebene

- **Granularität des zu testenden Systems**
  - (*target-of-test*)
- **in Reihenfolge in abnehmender Granularität**
  - Unit testing
  - Integration testing
  - System testing
  - Acceptance testing



[http://www.information-management.com/news/1026146-1.html]

Unit / Component testing

- ***Unit* ist die kleinste zu testende Softwareeinheit**
  - kann kompiliert, gelinkt, geladen werden …
  - Klassen, Interfaces, Prozeduren, ….
- **wird durch Entwicklung erstellt**
  - *buddy testing*
    - Testen und Entwicklung durch verschiedene Personen

Integrationstest I

- **fasst Units und Module zu sinnvollen Testgruppen zusammen**
  - setzt sich aus unterschiedlichen Quellen zusammen
    - Libraries, Subsysteme, Module, …
- **testet Kollaboration**
- **Integrationstests verdichten schrittweise**
  - bis das Gesamtsystem abgedeckt ist

Integrationstest II

- **wird durch Entwicklung durchgeführt**
  - Buddy testing

- **Testmethoden vergleichbar zu *Unit testing***

Systemtest I

- **testet vollständiges System**
  - vollständige Kollaboration aller Komponenten
  - Integration mit externen Systemen
- **beinhaltet nicht-funktionale Tests**
  - Performance, Last, …..

Systemtest II

- **verifiziert Umsetzung**
  - Abgleich mit Designspezifikation
- **wird durch Testteam durchgeführt**
- **wird auf spezieller Umgebung durchgeführt**

Abnahmetest I

- **…. auch als *Acceptancetest* bezeichnet**
- **testet, ob das System die Anforderungen erfüllt,**
  - aus Sicht des Kunden / Auftraggebers
  - entscheidender Teil der Abnahme
- **umfasst i.d.R. nicht-funktionale Tests**
  - Performance, Last , ….
- **umfasst das Gesamtprodukt**
  - inkl. Dokumentation, …

Abnahmetest II

- **wird durch Kunde / Auftraggeber durchgeführt**
- **wird in produktionsnaher/-identischer Umgebung ausgeführt**

Regressionstest

- **testet, ob eine Fehlerbehebung unerwartete Auswirkungen hat**
- **ist ein andauernder Prozess**
  - über den gesamten Testlebenszyklus
  - umfasst alle Testebenen
- **wird bei jeder Programmänderung durchgeführt**

Smoke Test I

- **ist ein Gesundheitscheck des System**
  - angestoßen i.d.R. durch den Bau des Systems (***system build***)
  - Synonym: *build verification test*
- **ist Verfahren während der Entwicklung**
  - evtl. Fehler werden sofort behoben
- **ist ein Typ eines Regressionstests**

Smoke Test II

- **beinhaltet oft eine aussagekräftige Teilmenge aller Tests**
  - alle Tests sprengen das mögliche Zeitfenster
- **gängige Verfahren sind**
  - *daily build*
  - *build on check-in*

Continuous Integration

- **im Team wird Software häufig integriert**
  - mindestens 1x täglich
  - jede Integration wird durch *Build* verifiziert
- **bedarf Werkzeugunterstützung**
- **ist Entwicklungsverfahren**
- **Verfahren für *Smoke Test* bei *build on check-in***

Wo sind wir

- Einführung
- Auf welcher Ebene wird getestet – *testing level*
- **Was wird getestet– *testing categories***
- Wie wird getestet – *testing methods and tools*
- Alles zusammen – *test execution*

Dimensionen der Qualität

- **Funktionalität (*functionality*)**
- **Benutzungsfreundlichkeit (*usability*)**
- **Zuverlässigkeit (*reliability*)**
- **Performance**
- **Pflegbarkeit (*supportability / maintainability*)**

Tests für Funktionalität

- **Fachliche Korrektheit**
- **Sicherheit**
- **Anforderungen an Mengengerüste**

Tests für Benutzungsfreundlichkeit

- Interaktion „Mensch-System“
- Ästhetische Aspekte
- Ergonomische Aspekte
- Online- und context-sensitive Hilfe
- Benutzerdokumentation

Tests für Zuverlässigkeit

- **Integritätstest**
  - Kompatibilität
- **Strukturtest auf**
  - Quellcode
  - Webseiten
- **Robustheitstest**
  - Wiederanlauffähigkeit

Tests für Performance

- **Benchmarking**
- **Test auf konkurrierender Nutzung**
  - Anzahl der Nutzer
- **Lasttests**
- **Stresstest**
  - Verhalten des Systems unter Extrembedingungen

Tests für Pflegbarkeit

- **Konfigurationstest**
  - Unterstützung unterschiedlicher Laufzeitumgebungen
- **Installationstest**

Wo sind wir

- Einführung
- Auf welcher Ebene wird getestet – *testing level*
- Was wird getestet– *testing categories*
- **Wie wird getestet – *testing methods and tools***
- Alles zusammen – *test execution*

Testmethoden

- **White Box Testing**
  - Interne Strukturen des Testziels sind gläsern
  - Tests nutzen interne Strukturen
  - Unittesting, Codeanalyse, Codeabdeckung, …
- **Black Box Testing**
  - Interne Strukturen des Testziels sind verborgen
  - Anforderungs- und Funktionalitätstests
- **Gray Box Testing**
  - wie Black Box Testing, nutzt aber innere Kenntnisse z.B. über Datenbankstrukturen

Testwerkzeuge I

- **lassen Test erstellen**
- **sind Spezialisten für**
  - einen Testtyp (oft noch spezialisierter)
  - eine Technologie
- **lassen sich nach Testmethoden klassifizieren**

Testwerkzeuge II

- **Beispiele**
  - JUnit für funktionale Unit- /Integrationstests
  - Capture & Replay für Webanwendungen
  - Grinder, HttpUnit für Web-Lasttests

- **OpenSource Testing Tools**
  - http://java-source.net/open-source/testing-tools

Wo sind wir

- Einführung
- Auf welcher Ebene wird getestet – *testing level*
- Was wird getestet– *testing categories*
- Wie wird getestet – *testing methods and tools*
- **Alles zusammen – *test execution***

Testausführung I

- **bedarf einer Ausführungsumgebung**
  - ist werkzeuggestützt

- **umfasst**
  - SCM und Build
  - Aufbau einer Testumgebung (optional)
  - Deployment des Systems (optional)
  - Deployment der Tests (optional)
  - Ausführung des Tests
  - Test Defect Management

Test Defect Management

- ***Test Defects*** **müssen systematisch behandelt werden**
  - *Defects* als Ausgang eines Tests
- ***Test Defect Management***
  - beinhaltet Reporting
  - unterstützt Fehlerbehebung
    - stellt einen Workflow bereit
  - liefert Testmetriken
    - Codeabdeckung
    - Maße für Defects
    - Reifegrad der Software

Testaktivitäten I

- **Begleitung der *requirement analyse***
  - Welche Aspekte / use cases des Systems sind testbar ?
  - Welche Szenarien sind denkbar ?
  - Rolle *test engineer*
- **Begleitung der Systemarchitektur**
  - Wie sieht eine Testumgebung aus?
  - Welche Systemaspekte sind zu testen (Performance, Durchsatz, …) ?
  - Rolle *test engineer*
- **Test planning: Teststrategie und Testplan werden aufgestellt**
  - Siehe IEEE 829
  - Rolle *test engineer*

Testaktivitäten II

- **Test development: Entwicklung der Tests**
  - *test procedures, test scenarios, test cases, test data, test scripts*
  - Rolle Entwickler (Unittest/Integrationstest) oder *test engineer*
- **Ausführung der Tests**
- **Test reporting: Auswertung der Testergebnisse**
- **Defect Analysis**
- **Retesting der behobenen Fehler**
- **Regression testing**

Zusammenfassung

- **ein Test hat 3 Dimensionen**
  - Testebene
    - In welcher Granularität wird getestet?
  - Testkategorie
    - Welche Qualitätseigenschaft wird getestet?
  - Testmethode
    - Welche Technologie wird mit welchem Werkzeug getestet?
- **Ausführungsumgebung- und -planung**
  - sorgt für die Ausführung und Ergebnisverwaltung der Tests

Ziel des Thementages

Sie erfahren heute, wie …

- **eine Testumgebung zur Ausführung von Tests aufgebaut ist / sein kann**
  - Werkzeuge
- **wie ein unterstützender Prozess aufgebaut ist / sein kann, der**
  - die Ausführung von Tests integriert
  - die Ausführung von Tests automatisiert
  - das Reporting von Testausführung integriert
- **ein solcher Prozess aus Sicht der Projektleitung eingeführt wird**

Referenzen

[Perry2000]
William E. Perry, *Effective Methods for Software Testing, 2nd Edition*, John Wiley & Sons, Inc., New York, NY, 2000, ISBN 0-471-35418-X

[McConnell2004]
McConnell, Steve (2004). *Code Complete* (2nd ed.). Microsoft Press. pp. 960. ISBN 0-7356-1967-0.

Links

http://rup.hops-fp6.org/process/workflow/test/co_tytst.htm

http://www.softwaretestingwiki.com/doku.php

http://www.nickjenkins.net/prose/testingPrimer.pdf

http://martinfowler.com/articles/continuousIntegration.html

www.iks-gmbh.com

Anhang

Integrationtesting III

- **Bottom Up Integrationstesting**
  - Low-Level Komponenten zuerst
  - Einsatz von Treibern, die die Integration auf höherer Ebene simulieren
  - Setzt entsprechende Designstrategie voraus

Integrationtesting III

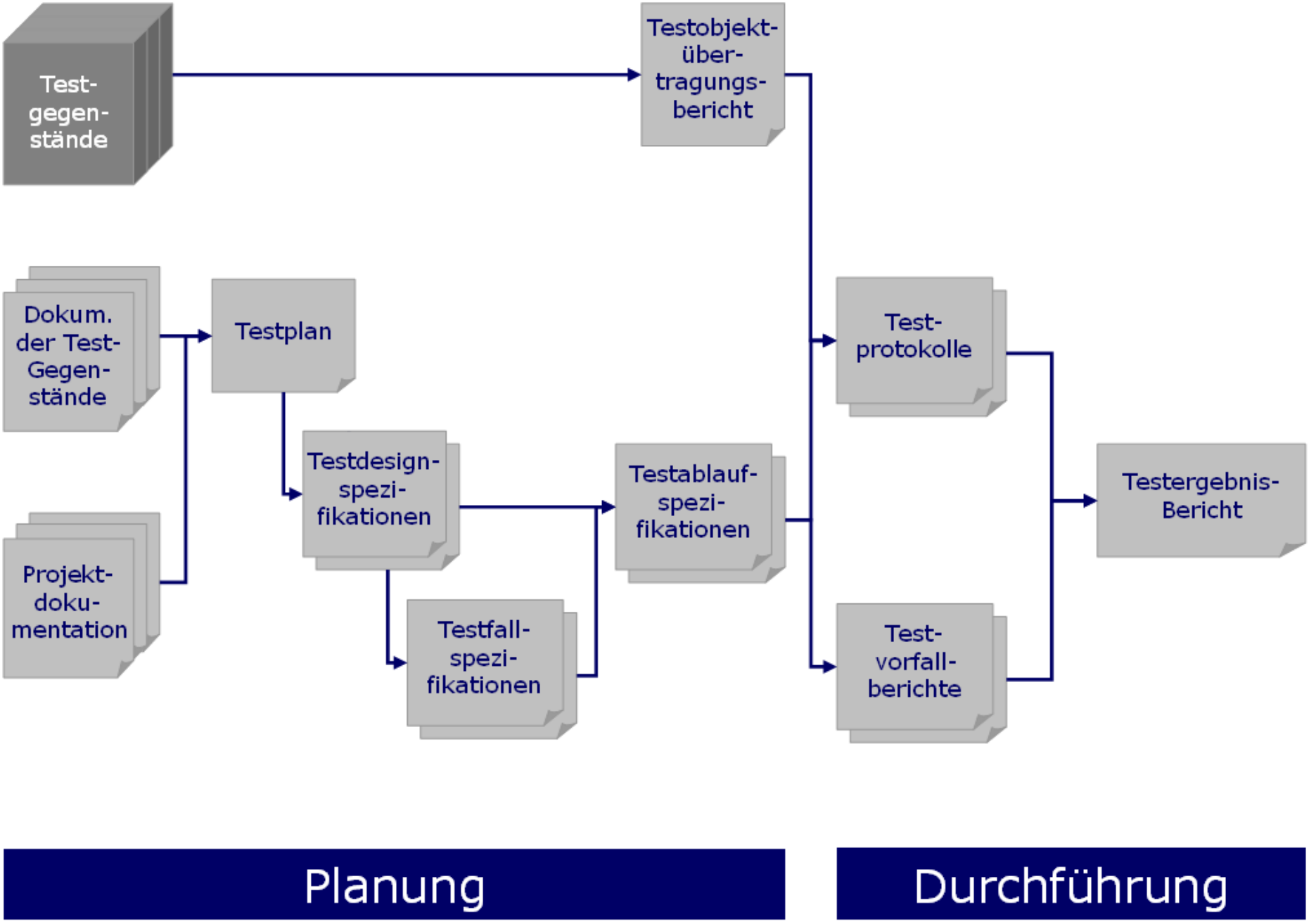
- **Top Down Integrationstest**
  - High-Level Komponenten zuerst
  - Benötigte Komponenten werden simuliert
  - Simulation mittels Stubbing/Mocking
  - Setzt entsprechende Designstrategie voraus

Intergrationstesting IV

- **Mixed-testing**
  - Button up
    - Infrastruktur Komponenten
  - Top down
    - ansonsten

Best practices

- Test Early, Test Often
- Regression vs. Retesting
- White-Box vs Black-Box testing
- Verification and Validation


Test-
gegen-
stände
Testobjekt-
über-
tragungs-
bericht
Dokum.
der Test-
Gegen-
stände
Testplan
Testdesign-
spezi-
fikationen
Testablauf-
spezi-
fikationen
Test-
protokolle
Testergebnis-
Bericht
Projekt-
doku-
mentation
Testfall-
spezi-
fikationen
Test-
vorfall-
berichte
Planung
Durchführung