

Anforderungsgetriebene Entwicklung testbarer Software

Vortrag für die Fachgruppe "Software-Test" des



am 28. März 2006 im IGZ, Erlangen

Dr. Stefan Jungmayr
Teradyne Diagnostic Solutions

Sprecher des GI-Arbeitskreises "Testen objektorientierter Programme"
www.testbarkeit.de

© 2006, Stefan Jungmayr

TERADYNE



Assembly Test Division
Circuit Board
Inspection & Test

Semiconductor Test Division
Device & Wafer
Test

Diagnostic Solutions (DS) Division
Automotive Test
& Diagnostics

Broadband Test Division
Telecommunication
Systems Test

weltweit führender Lieferant von Diagnose- und Informationslösungen

Umsatz 2005 (gesamt): US\$ 1,08 Milliarden

Mitarbeiter (gesamt): 4100

Mitarbeiter (DS): 390

Kunden (DS): BMW, Claas, DaimlerChrysler, Ford, GM, Honda, Jaguar
Knorr Bremse, MAN, Saab, Volvo

Website: www.teradyne.com

© 2006, Stefan Jungmayr

Testprobleme (I)

- ▶ Einleitung
 - ▶ **Testprobleme**
 - ▶ Definition Testbarkeit
 - ▶ Ist-Situation
 - ▶ Lösungsansätze
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

“Früher konnte man sicher sein,
dass der Kindersitz ein abgetrenntes Modul ist und
der Nebelscheinwerfer ein anderes.

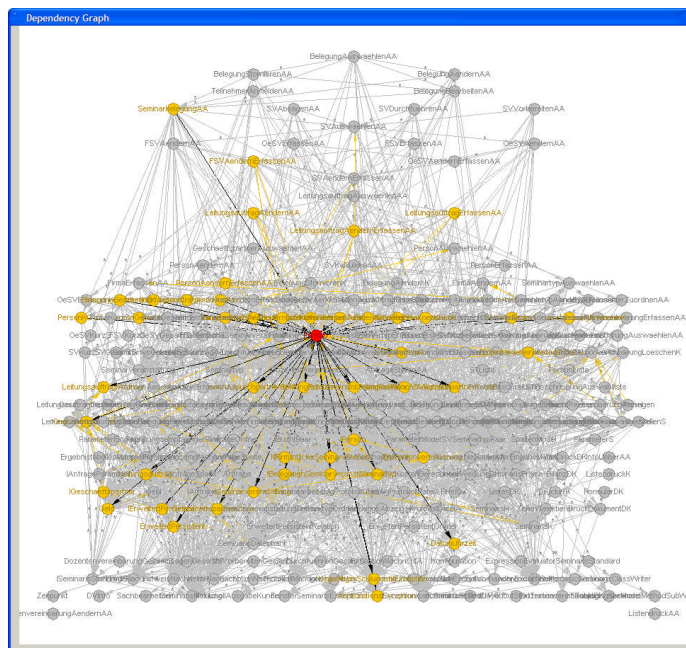
Seit aber der Sitz einen Einlullmotor hat, interferieren
beide mit der Elektronik. Wenn jetzt
das Baby Milchsäure ausspuckt, beginnen
Wechselwirkungen im ganzen Auto.

[...]

Dann suchen Sie einmal den Fehler!” [Duec05]

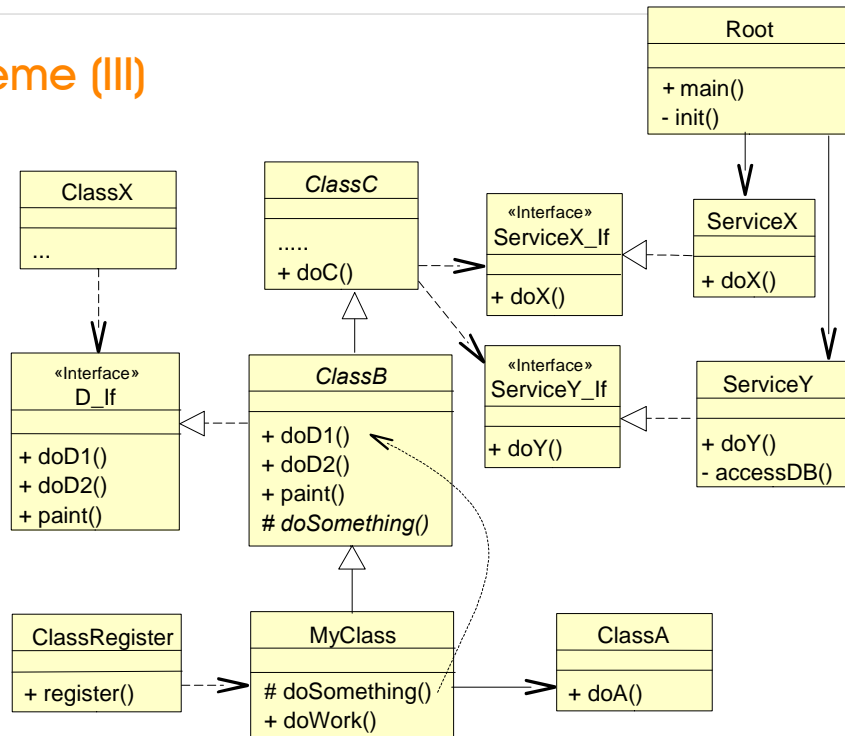
Testprobleme (II)

- ▶ Einleitung
 - ▶ **Testprobleme**
 - ▶ Definition Testbarkeit
 - ▶ Ist-Situation
 - ▶ Lösungsansätze
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung



Testprobleme (III)

- ▶ Einleitung
 - ▶ Testprobleme
 - ▶ Definition Testbarkeit
 - ▶ Ist-Situation
 - ▶ Lösungsansätze
 - ▶ Problemanalyse
 - ▶ Anforderungsgetriebene Entwicklung testbarer SW
 - ▶ Zusammenfassung



© 2006, Stefan Jungmayr

Definition Testbarkeit

- ▶ Einleitung
 - ▶ Testprobleme
 - ▶ Definition Testbarkeit
 - ▶ Ist-Situation
 - ▶ Lösungsansätze
- ▶ Problemanalyse
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

Testbarkeit ist der Grad, zu dem ein Software-Artefakt den Test in einem bestimmten Testkontext ermöglicht.

- Software-Artefakt:
 - Anforderung
 - SW-Architektur
 - Entwurf
 - ausführbarer Code
- Testkontext:
 - Testziele
 - Testprozess
 - Testwerkzeuge, Testumgebung

© 2006, Stefan Jungmayr

Wo stehen Sie?

- ▶ Einleitung
 - ▶ Testprobleme
 - ▶ Definition Testbarkeit
 - ▶ **Ist-Situation**
 - ▶ Lösungsansätze
- ▶ Problemanalyse
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

- Wie hoch ist Ihr Testaufwand?
 - in Prozent des gesamten Entwicklungsaufwands
 - inkl. Fehlersuche / Debugging
 - Einzelnennungen der Teilnehmer: 40%, 50%, 30-40%
- Welcher Anteil Ihres Testaufwands ist auf Testprobleme zurückzuführen?

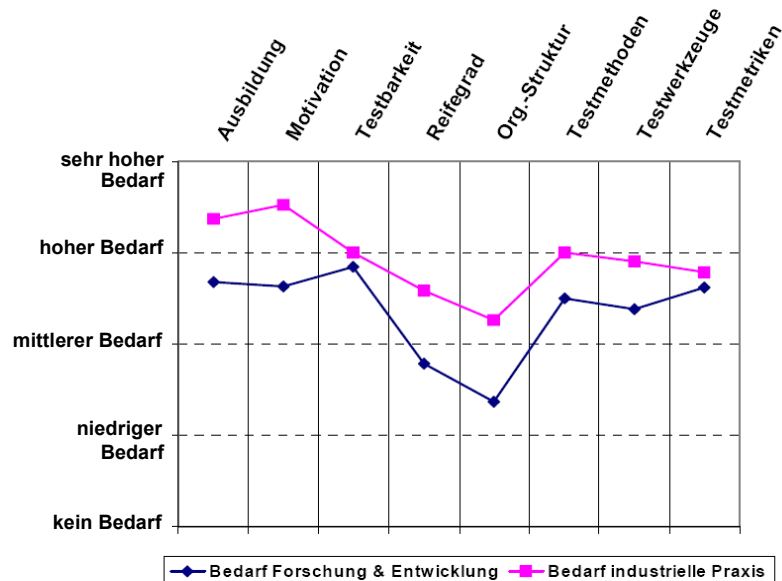
Nennungen der Teilnehmer:

- 0%: 0 .
- 6%: 0 .
- 12%: 1 *
- 25%: 5 *****
- 50%: 18 ***** ***** ***** ***
- 75%: 2 **

© 2006, Stefan Jungmayr

Bedeutung der Testbarkeit

- ▶ Einleitung
 - ▶ Testprobleme
 - ▶ Definition Testbarkeit
 - ▶ **Ist-Situation**
 - ▶ Lösungsansätze
- ▶ Problemanalyse
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung



Umfrage unter 21 Teilnehmern einer Test-Fachtagung [Jung03]

© 2006, Stefan Jungmayr

Was tun Sie?

- ▶ Einleitung
 - ▶ Testprobleme
 - ▶ Definition Testbarkeit
 - ▶ **Ist-Situation**
 - ▶ Lösungsansätze
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

- Was tun Sie für eine ausreichende Testbarkeit?

Nennungen der Teilnehmer:

- “Testprogramme schreiben”
- “SE-Methoden einsetzen”
- “Testbarkeitsanforderungen formulieren”
- “Testbares Design”
- “Tester bei Reviews des Design involvieren”
- “Testwerkzeuge einsetzen”

© 2006, Stefan Jungmayr

Bekannte Ansätze für mehr Testbarkeit

- ▶ Einleitung
 - ▶ Testprobleme
 - ▶ Definition Testbarkeit
 - ▶ Ist-Situation
 - ▶ **Lösungsansätze**
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

- Ad-hoc-Ansatz:
 - an das Gute im Entwickler appellieren
 - anerkannten Entwurfsrichtlinien folgen (guter Entwurf)
 - objektorientiert entwickeln
- Entwurf für Testbarkeit:
 - Entwickler und Tester in testbarem Entwurf trainieren
 - Entwurfsrichtlinien definieren
 - Testbarkeit in Reviews berücksichtigen
- Testgetriebene Entwicklung

© 2006, Stefan Jungmayr

Bekannte Ansätze für mehr Testbarkeit

- ▶ Einleitung
 - ▶ Testprobleme
 - ▶ Definition Testbarkeit
 - ▶ Ist-Situation
 - ▶ **Lösungsansätze**
- ▶ Problemanalyse
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

- **Ad-hoc-Ansatz:**
 - an das Gute im Entwickler appellieren
 - anerkannten Entwurfsrichtlinien folgen (guter Entwurf)
 - objektorientiert entwickeln
- **Entwurf für Testbarkeit:**
 - Entwickler und Tester in testbarem Entwurf trainieren
 - Entwurfsrichtlinien definieren
 - Testbarkeit in Reviews berücksichtigen
- **Testgetriebene Entwicklung**

© 2006, Stefan Jungmayr

Testgetriebene Entwicklung

- ▶ Einleitung
 - ▶ Testprobleme
 - ▶ Definition Testbarkeit
 - ▶ Ist-Situation
 - ▶ **Lösungsansätze**
- ▶ Problemanalyse
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

- **Testcode vor Funktionalität codiert**
 - eine Technik aus dem XP-Umfeld
 - auch anwendbar in konventionellen Prozessen
- **Vorteil:**
 - früh Kontrolle der Testbarkeit der Anforderungen
 - Testfall zu jeder Funktionalität (Regressionstest)
 - verbesserte Testbarkeit des Codes, da Testprobleme frühzeitig erkannt
- **Nachteil:**
 - Testbarkeit wird nicht systematisch entwickelt
 - Trial-and-Error-Prozess
 - beruht auf Erfahrung/Einschätzung des Entwicklers
 - erforderliche Testbarkeit und Erreichung unsicher
 - Mangel an Wiederholbarkeit
 - Effektivität ?

© 2006, Stefan Jungmayr

Frage ...

- ▶ Einleitung
 - ▶ Testprobleme
 - ▶ Definition Testbarkeit
 - ▶ Ist-Situation
- ▶ **Lösungsansätze**
- ▶ Problemanalyse
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

- Sind die bekannten Ansätze ausreichend?
- → Was ist das Kernproblem, das es zu lösen gilt?

© 2006, Stefan Jungmayr

Unterschied Implementierungs- versus Test-Sicht

- ▶ Einleitung
- ▶ **Problemanalyse**
 - ▶ Anforderungen
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

Implementierung

- indirekte Dienstleisterklassen verborgen durch Information-Hiding
- Klasse wird eingebunden in System/Rahmenwerk
- Code wird bei Bedarf angepasst
- Priorität: es läuft und stürzt nicht ab, gute Performanz
- Komplexität und Dokumentation sekundär
- Beobachtung über Debugger möglich
- Fehler-Isolation z.T. intuitiv

Test

- indirekte Dienstleisterklassen für Test relevant (insbesondere bei Instanziierung, Initialisierung)
- Klasse muss isoliert werden, Zustand soll möglichst direkt gesetzt werden
- Code kann nicht verändert werden
- Priorität: Wiederholbarkeit der Testergebnisse
- Verständnis von fremden Code erforderlich und schwierig
- Beobachtung nur über Schnittstelle möglich, ggf. in Betriebsumgebung
- Fehlernachweis schwierig

© 2006, Stefan Jungmayr

Anforderungen von Testern

- ▶ Einleitung
- ▶ Problemanalyse
 - ▶ **Anforderungen**
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

- **produktbezogene Anforderungen:**
 - angemessene Komplexität
 - isolierter Test von Komponente möglich
 - Zustand der Komponenten setzbar, beobachtbar
 - Test reproduzierbar, automatisierbar, robust, schnell
 - Fehler lässt sich Komponente zuordnen
- **prozessbezogene Anforderungen:**
 - Produkthanforderung verfügbar, aktuell und stabil
 - Produkthanforderungen testbar und verfolgbar

© 2006, Stefan Jungmayr

Anforderungen von Entwicklern (Wartung)

- ▶ Einleitung
- ▶ Problemanalyse
 - ▶ **Anforderungen**
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

- **Unterstützung der Fehler-Isolation:**
 - Fehlermeldungen und Auslöser eindeutig
 - Konfiguration des Systems über Fehlermeldung bzw. UI nachvollziehbar
 - insbesondere, wenn SW durch Kunde installiert
 - Test im laufenden Betrieb möglich
 - vernachlässigbare Wirkung auf Systemverhalten
 - Systemzustand zugänglich
 - Log-Informationen konfigurierbar
 - selbstdefinierte Abfragen an das System möglich

© 2006, Stefan Jungmayr

Anforderungen von Benutzern

- ▶ Einleitung
- ▶ Problemanalyse
 - ▶ **Anforderungen**
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
- ▶ Zusammenfassung

- **Unterstützung der Fehlererkennung:**
 - Systemzustand zugänglich
 - Systemzustand & Fehlermeldungen nachvollziehbar
 - Prüfung der Integrität durch Selbst-Test
- **Unterstützung der Fehlermeldung:**
 - Fehlermeldung sowie Systemzustand bei Fehlereintritt dokumentierbar
 - Weiterleitung der Fehlermeldung an Entwickler automatisiert

© 2006, Stefan Jungmayr

Anforderungsgetriebene Testbarkeitsentwicklung

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ **Anforderungsgetriebene Entwicklung testbarer SW**
 - ▶ Modell der Testbarkeit
 - ▶ Anforderungsermittlung
 - ▶ Architekturentwurf
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

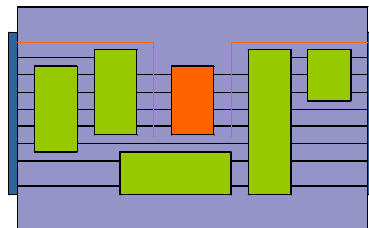
- **Anforderungen an die Testbarkeit definieren**
 - explizit formulieren
 - priorisieren und bewerten (ggf. Trade-Offs notwendig)
- **Ausgangspunkt ist ein Modell der Testbarkeit**
- **systematische Umsetzung im Entwicklungsprozess**
 - wie bei allen anderen Arten von Anforderungen

© 2006, Stefan Jungmayr

Exkurs: Vergleich mit Hardware-Testbarkeit

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ **Anforderungsgetriebene Entwicklung testbarer SW**
 - ▶ Modell der Testbarkeit
 - ▶ Anforderungsermittlung
 - ▶ Architekturentwurf
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

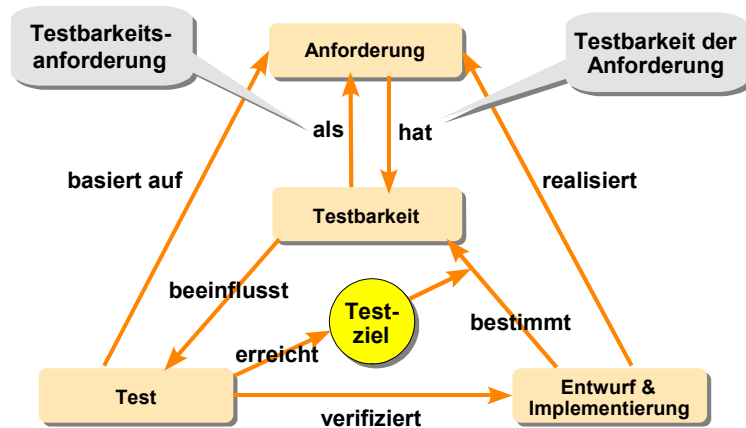
- **Elektronische Ausrüstung:**
 - MIL-STD-2165 (1985), u.a. Testbarkeitsanforderungen
- **U-Boot-Systeme:**
 - Testability Analysis Handbook (1992)
u.a. Testbarkeitsanforderungen
- **VLSI:**
 - Anforderung: jedes IC muss getestet werden
 - Problem: Mangel an Steuerbarkeit / Beobachtbarkeit
 - Lösung:
Boundary-Scan-Architektur (IEEE 1149.1)



© 2006, Stefan Jungmayr

Modell der Testbarkeit (I)

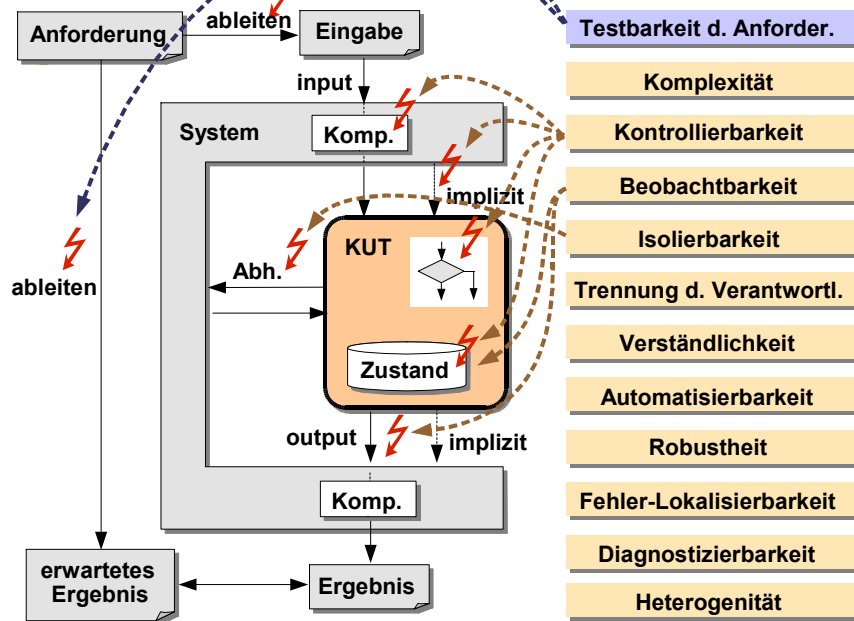
- ▶ Einleitung
- ▶ Problemanalyse
- ▶ **Anforderungsgetriebene Entwicklung testbarer SW**
 - ▶ **Modell der Testbarkeit**
 - ▶ Anforderungsermittlung
 - ▶ Architekturentwurf
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung



© 2006, Stefan Jungmayr

Modell der Testbarkeit (II)

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
 - ▶ Modell der Testbarkeit
 - ▶ Anforderungsermittlung
 - ▶ Architekturentwurf
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

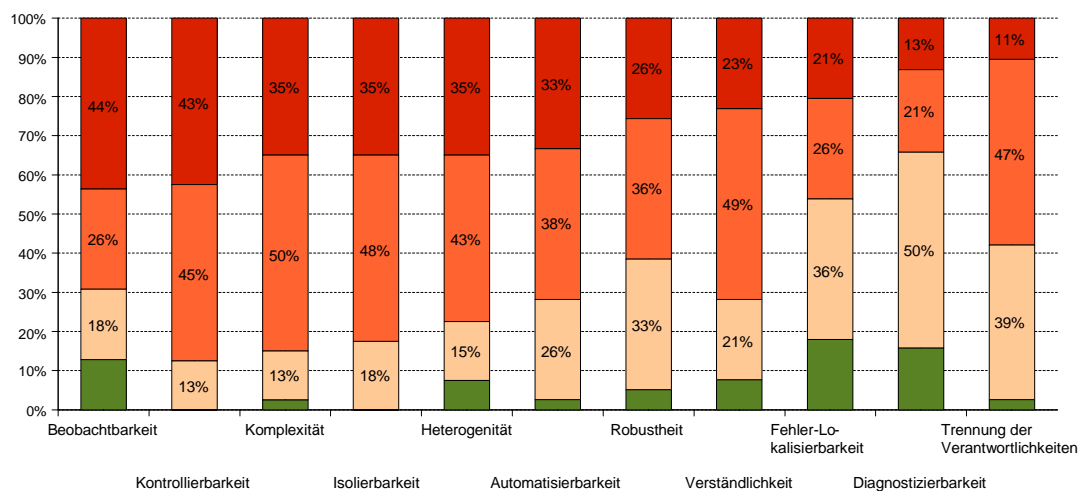


© 2006, Stefan Jungmayr

Priorität der Anforderungen

Auswirkung auf den Testaufwand

- hoch
- mittel
- niedrig
- keine



© 2006, Stefan Jungmayr

Exkurs: Testbarkeit und andere Qualitätskriterien

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
 - ▶ **Modell der Testbarkeit**
 - ▶ Anforderungsermittlung
 - ▶ Architekturentwurf
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

- **Wartbarkeit:**
 - Testbarkeit ist für Regressionstest wichtig
- **Debuggbarkeit**
 - Testen ≠ Debuggen (Fehler-Isolation und -Behebung)
 - aber: Testbarkeit und Fehler-Isolierbarkeit überlappen stark
- **Performanz**
 - Laufzeitoptimierung reduziert häufig Testbarkeit
 - Testbarkeit kann Performanz reduzieren
- **Nachvollziehbarkeit von Ergebnissen**
 - bei kritischen, verteilten, nicht-determinist. Systemen

© 2006, Stefan Jungmayr

Testbarkeit in Anforderungsermittlung + Analyse

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
 - ▶ Modell der Testbarkeit
 - ▶ **Anforderungsermittlung**
 - ▶ Architekturentwurf
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

- **Testbarkeit der Anforderungen erhöhen:**
 - Abhängigkeiten von Anwendungsfällen reduzieren
 - Zugriff auf gemeinsame Klassen reduzieren
 - Vorbedingungen lockern
 - zeitliche Abhängigkeiten reduzieren
 - untere Multiplizitätsgrenzen lockern
 - Anforderungen quantitativ formulieren
 - schwer testbare Anforderungen identifizieren
- **testkritische Domänenklassen identifizieren:**
 - enthalten kritische Anwendungslogik
 - sind schwer beobachtbar/automatisierbar
 - enthalten Zugriff auf langsame Ressourcen
- **Testbarkeitsanforderungen definieren**
 - funktional / nicht-funktional

© 2006, Stefan Jungmayr

Beispiel Testbarkeitsanforderung, nicht-funktional

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
 - ▶ Modell der Testbarkeit
 - ▶ **Anforderungsermittlung**
 - ▶ Architekturentwurf
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

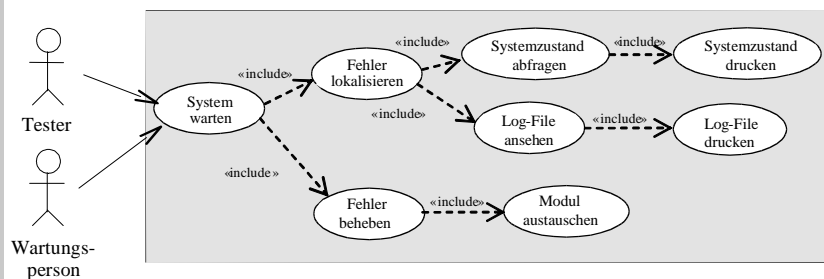
- **Isolierbarkeit:**
 - “Für 90% aller Klassen erfordert JUnit-Testsetup die Instantiierung von Objekten aus weniger als 7 Dienstleisterklassen.”
- **Fehler-Lokalisierbarkeit:**
 - “Durchschnittlich < 30 Minuten erforderlich, um einen kritischen Fehler zu isolieren.”
- **Beobachtbarkeit:**
 - “Abstrakter (d.h. fachlicher) Zustand testkritischer Komponenten ist über eine Testschnittstelle abrufbar.”

© 2006, Stefan Jungmayr

Beispiel Testbarkeitsanforderung, funktional (I)

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
 - ▶ Modell der Testbarkeit
 - ▶ **Anforderungsermittlung**
 - ▶ Architekturentwurf
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

- **Anwendungsfälle im Kontext Systemwartung:**



© 2006, Stefan Jungmayr

Beispiel Testbarkeitsanforderung - funktional (II)

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
 - ▶ Modell der Testbarkeit
 - ▶ **Anforderungsermittlung**
 - ▶ Architektorentwurf
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

Anwendungsfall Systemzustand abfragen

Akteur: Tester, Wartungsperson

Ablauf: Der Akteur fragt den Systemzustand ab. Der Systemzustand beinhaltet Ereignisse (Ein- und Ausgabe-Operationen, Fehlerereignisse) und Systemkennzahlen (CPU-Nutzung, Speichernutzung, Anzahl der Objekt-Instanzen, Anzahl der Threads). Die Abfrage des Systemzustands ist passwortgeschützt. Der Akteur kann die Ausgaben sortieren und filtern (nach Zeitpunkt und Kritikalität). Der Akteur kann Ereignisse und Systemkennzahlen auswählen und drucken (**include** „Systemzustand drucken“).

Vorbedingung: keine (soweit möglich, soll der Anwendungsfall in allen möglichen Systemzuständen ablauffähig sein)

Nachbedingung: Der aktuelle Systemzustand wurde ausgegeben.

Ausnahmefälle: keine

© 2006, Stefan Jungmayr

Testbarkeit im Architektorentwurf (I)

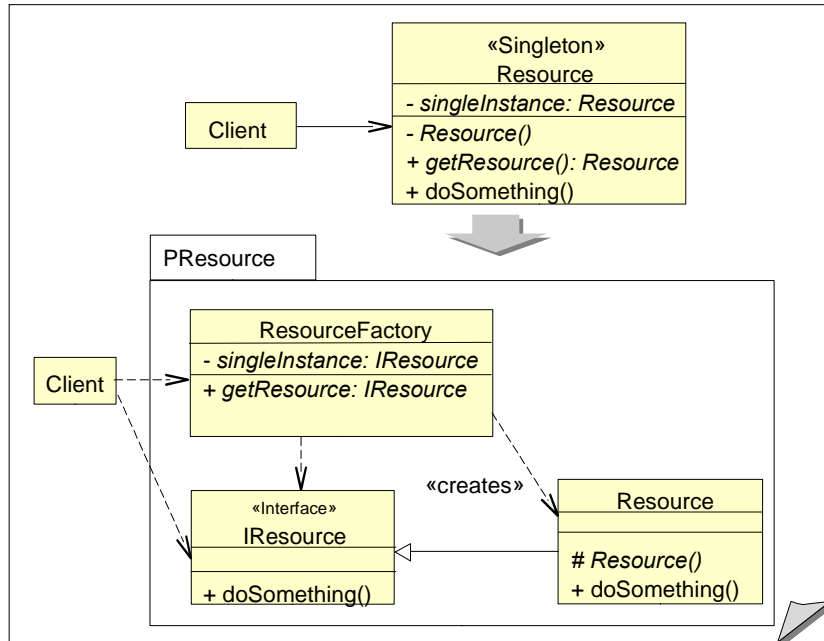
- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
 - ▶ Modell der Testbarkeit
 - ▶ Anforderungsermittlung
 - ▶ **Architektorentwurf**
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

- Komplexität reduzieren
- Verantwortlichkeiten trennen
- Isolierbarkeit sicherstellen
 - “Durchstich” für Test in Isolation
 - Mock-Objekte für systemweite Dienste
 - vergl. Mock-Plattform [Völt05]
 - Objektfabriken
 - Singleton-Muster vermeiden
 - Dependency Injection

© 2006, Stefan Jungmayr

Singleton-Muster

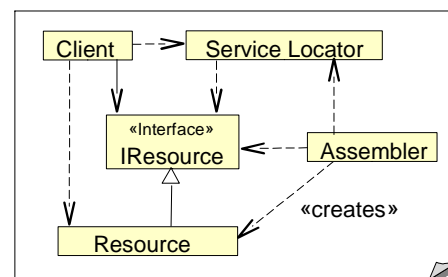
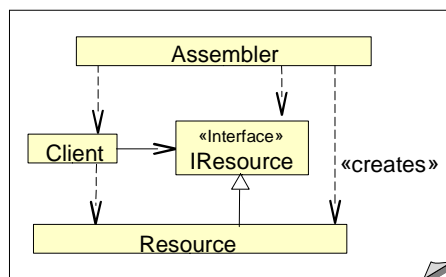
- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
 - ▶ Modell der Testbarkeit
 - ▶ Anforderungsermittlung
 - ▶ **Architekturstudium**
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung



© 2006, Stefan Jungmayr

Dependency Injection

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderungsgetriebene Entwicklung testbarer SW
 - ▶ Modell der Testbarkeit
 - ▶ Anforderungsermittlung
 - ▶ **Architekturstudium**
 - ▶ Komponententwurf
 - ▶ Implementierung
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung



© 2006, Stefan Jungmayr

Testbarkeit im Architekturentwurf (II)

- Einleitung
- Problemanalyse
- Anforderunggetriebene Entwicklung testbarer SW
 - Modell der Testbarkeit
 - Anforderungsermittlung
 - **Architekturentwurf**
 - Komponententwurf
 - Implementierung
 - Verfolgbarkeit
- Zusammenfassung

- **Beobachtbarkeit sicherstellen**
 - Testschnittstelle
 - abstrakter (d.h. fachlicher) Zustand abfragbar
 - Zugriff auf Laufzeitumgebung:
 - Anzahl der Objektinstanzen, Threads
 - Speicherverbrauch
 - Selbstauskunft über Version und Konfiguration
 - Logging auf unterschiedlichen Leveln
 - tw. Kundenanforderung
- **Automatisierbarkeit sicherstellen**
- **Fokus auf testkritische Klassen**

© 2006, Stefan Jungmayr

Testbarkeit im Komponententwurf

- Einleitung
- Problemanalyse
- Anforderunggetriebene Entwicklung testbarer SW
 - Modell der Testbarkeit
 - Anforderungsermittlung
 - Architekturentwurf
 - **Komponententwurf**
 - Implementierung
 - Verfolgbarkeit
- Zusammenfassung

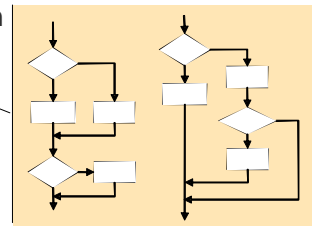
- **Komplexität reduzieren**
 - hohe Schnittstellen-Komplexität vermeiden
 - tiefe Vererbungshierarchien vermeiden
 - zyklische Klassenabhängigkeiten vermeiden
 - Abh. zu indirekten Dienstleisterklassen vermeiden
- **Fehler-Lokalisierbarkeit verbessern**
 - Yo-Yo Effekt reduzieren
 - implizite Abhängigkeiten (Seiteneffekte) vermeiden
- **Fehler-Lokalität verbessern**
 - defensive Programmierung an Subsystemgrenzen

© 2006, Stefan Jungmayr

Testbarkeit in der Implementierung

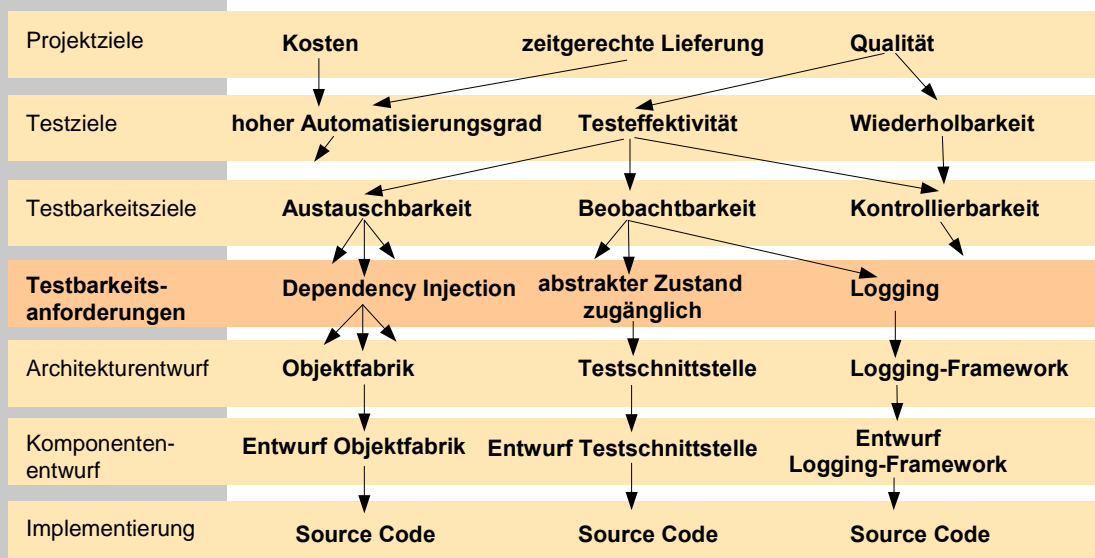
- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
 - ▶ Modell der Testbarkeit
 - ▶ Anforderungsermittlung
 - ▶ Architekturentwurf
 - ▶ Komponententwurf
 - ▶ **Implementierung**
 - ▶ Verfolgbarkeit
- ▶ Zusammenfassung

- **Verständlichkeit sicherstellen**
 - "elegante", trickreiche Lösungen vermeiden
 - erst Performanz optimieren wenn korrekt und testbar
 - implizite Kontrolllogik vermeiden
- **Kontrollierbarkeit sicherstellen**
 - komplexe Schleifenkonstrukte vermeiden
 - Trennung von Iterator-Code und "Arbeits-Code".
 - Rekursion vermeiden
 - unerreichbare Ausgabewerte vermeiden
 - unerreichbare Pfade vermeiden



© 2006, Stefan Jungmayr

Verfolgbarkeit der Testbarkeitsanforderungen



© 2006, Stefan Jungmayr

Ansätze für mehr Software-Testbarkeit

- Einleitung
- Problemanalyse
- Anforderungsgetriebene Entwicklung testbarer SW
- **Zusammenfassung**

- Ad-hoc-Ansatz
- Entwurf für Testbarkeit
- Testgetriebene Entwicklung
- Anforderungsgetriebene Testbarkeitsentwicklung

Vorteile:

- Einbindung in Standard-Entwicklungsprozess
- Verfolgbarkeit der Testbarkeitsanforderungen
- Wiederholbarkeit
- Effektivität

© 2006, Stefan Jungmayr

Wo wollen Sie in 2 Jahren sein?

- Einleitung
- Problemanalyse
- Anforderungsgetriebene Entwicklung testbarer SW
- **Zusammenfassung**

- Testbarkeitsanforderungen in Entwicklungsprozessen und -Produkten verankert.
- Know-How aufgebaut:
 - **Analytiker**: kennen konstruktive Möglichkeiten, die Testen erleichtern → können Testbarkeitsanforderungen definieren
 - **Entwickler**: können Testbarkeitsanforderungen umsetzen
 - **Projektleiter**: treffen richtige Trade-Offs zwischen Testbarkeit und anderen Anforderungen / Zielen
- Testbarkeit im Entwicklungsprozess kontrolliert.
- Kritische Testbarkeitsmängel werden vermieden bzw. frühzeitig erkannt und behoben.

© 2006, Stefan Jungmayr

Was sind Ihre nächsten Schritte?

- ▶ Einleitung
- ▶ Problemanalyse
- ▶ Anforderunggetriebene Entwicklung testbarer SW
- ▶ **Zusammenfassung**

- **Testprobleme sammeln, z.B. in Fehler-DB**
 - Testprobleme kategorisieren, Ursachen analysieren
- **Lösungsmöglichkeiten sammeln**
 - gemeinsam mit Entwicklern erarbeiten
 - externe Ideen sammeln
 - Handlungsspielraum bestimmen
- **Testbarkeitsanforderungen definieren**
- **Zusatzkosten durch Testprobleme bestimmen**
 - wieviel Aufwand durch Debugging
- **Motivieren**
 - Kunden ins Boot holen?

© 2006, Stefan Jungmayr

Anhang: Informationsquelle

- **www.testbarkeit.de**
 - Artikel zum Thema Testbarkeit
 - Literaturhinweise
 - Web-Links
 - Kontaktformular zum Vortragenden

© 2006, Stefan Jungmayr

Anhang: Referenzen

- [Duec05] G. Dueck. Aveyware. Informatik-Spektrum, August 2005, S. 313.
- [Jung03] S. Jungmayr. TAV-Testbarometer: Ein kleiner Blick in die Zukunft des Testens. Softwaretechnik-Trends, Band 23, Heft 4, Nov. 2003.
- [Jung04] S. Jungmayr. Improving testability of object-oriented systems. dissertation.de, 2004. ISBN 3-89825-781-9.
- [Jung06] Stefan Jungmayr. Testbarkeitsfaktoren und Testaufwand: Auswertung zweier Umfragen. URL: http://www.testbarkeit.de/Publikationen/Umfrage_Testaufwand.pdf
- [Lakos96] J. Lakos. Large-scale C++ software design. Addison-Wesley, 1996. ISBN 0201633620.
- [Snee06] H. Sneed, S. Jungmayr. Produkt- und Prozessmetriken für den Software Test. Informatik Spektrum, Band 29, Nr. 1, Feb. 2006, S. 23-38.
- [Völt05] M. Völter, A. Haase. Architektur ohne Hype. Javamagazin, Nov. 2005, S. 45-51.