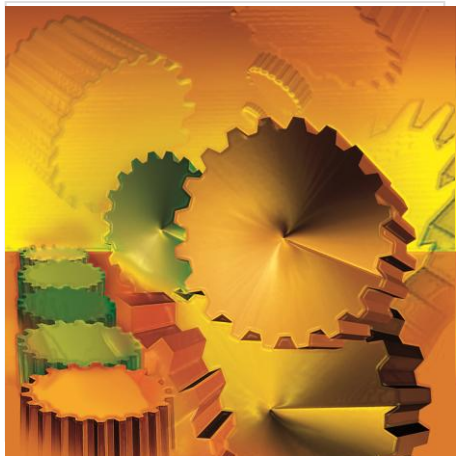
Software Engineering und Projektmanagement 2.0 VO



Vorlesung Software Testen

2015W

Wolfgang Gruber, Roland Breiteneder

www.inso.tuwien.ac.at

INSO - Industrial Software
Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien

Der Therac-25 Fall

Prominentes Literaturbeispiel zum „Versagen des Tests“

Strahlentherapiegerät für die Krebstherapie

SW-Fehler verursachte Überbestrahlung:

- Normale Therapie 1 Gray
- Tödlich ab 10 Gray
- SW-Fehler verursachte 40-200 Gray

Mind. 3 bestätigte Todesfälle

Entwicklung und Test von nur einer Person

→ Notwendigkeit der Unabhängigkeit des Tests bei entsprechendem Risiko

Agenda

1. Definition
2. Teststufen
3. Testmethoden und -abdeckung
4. Nichtfunktionale Tests
5. Testautomatisierung
5. Testmanagement

Stellenwert von Softwaretests

- Keine Software ist frei von Fehlern
- Testen von Software gehört zu den wichtigsten Aktivitäten des Software-Entwicklungsprozesses
- Systematisches Testen notwendig
- Eine gut geplante Testphase ist die Grundlage für ein erfolgreiches Projekt
- Ziel ist es, Fehler so früh wie möglich zu finden
- Softwaretests sind eine dynamische und produktorientierte Qualitätssicherungstechnik
- Aufgrund wachsender Komplexität und hoher Abhängigkeiten von Softwaresystemen, wird der Softwaretest strategisch immer bedeutender

Definition Softwaretests

Definition nach IEEE 610.12, 1990 :

„Software testing is a formal process carried out by a specialized testing team in which a software unit, several integrated software units or an entire software package are examined by running the programs on a computer. All the associated tests are performed according to approved test procedures on approved test cases.“

Ziele des Softwaretests

1. Ziel ist es, Testfälle zu identifizieren, mit denen die höchste Wahrscheinlichkeit gegeben ist, festzustellen, ob das Softwaresystem korrekt funktioniert
2. Ein guter Test ist jener, der eine möglichst hohe funktionale oder nichtfunktionale Abdeckung hat
3. Beim Softwaretest sollen Fehler von Programmen identifiziert werden.
4. Wenn ein Test einen Fehler gefunden hat, war er erfolgreich

Grundsätze des Softwaretests (nach ISTQB)

1. Grundsatz: Testen zeigt die Anwesenheit von Fehlern
2. Grundsatz: Vollständiges Testen ist nicht möglich
3. Grundsatz: Mit dem Testen frühzeitig beginnen
4. Grundsatz: Häufung von Fehlern
5. Grundsatz: Wiederholungen haben keine Wirksamkeit
6. Grundsatz: Testen ist abhängig vom Umfeld
7. Grundsatz: Trugschluss: Keine Fehler bedeutet ein brauchbares System

Begriff „Fehler“

Failure:

An incorrect result [in a system]

Error:

The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition

Fault:

An incorrect step, process, or data definition in a computer program

Mistake:

A human action that produces an incorrect result

Validierung vs. Verifikation

Definitionen nach IEEE 610.12-1990:

Validation:

The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.

Verification:

The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

ISO/IEC/IEEE 29119 Software Testing

- ISO/IEC 29119-1: Concepts & Definitions
- ISO/IEC 29119-2: Test Processes
- ISO/IEC 29119-3: Test Documentation

In Vorbereitung:

- ISO/IEC 29119-4: Test Techniques
- ISO/IEC 29119-5: Keyword Driven Testing

Test- und Integrationsstufen

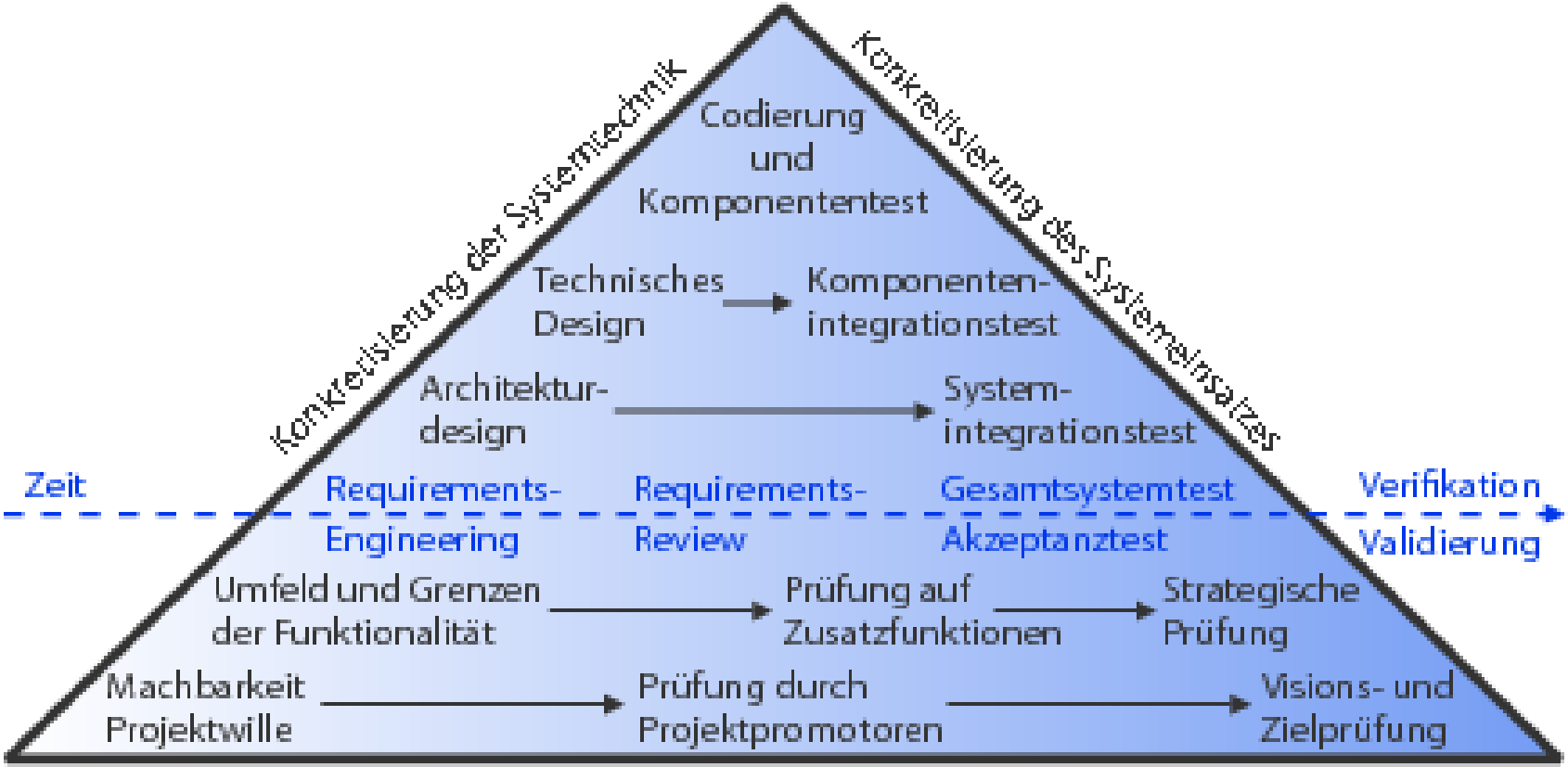
Strategie der Zerlegung der gestellten Aufgabe in beherrschbare Teile bei der Herstellung von Softwaresystemen

Zerlegung in Teststufen ermöglicht eine frühe Prüfung von unterschiedliche Teilen des zu entwickelnden Systems

Teststufen können durch folgenden Aspekte charakterisiert werden:

- allgemeine Ziele
- Arbeitsergebnisse, als Grundlage für daraus abgeleitete Testfälle (Testbasis)
- das eigentliche Testobjekt (Testgegenstand)
- auftretende Fehlerwirkungen und Zustände, die identifiziert werden sollten
- Anforderungen an den Testrahmen; Werkzeugunterstützung;
- spezifische Ansätze und Verantwortlichkeiten

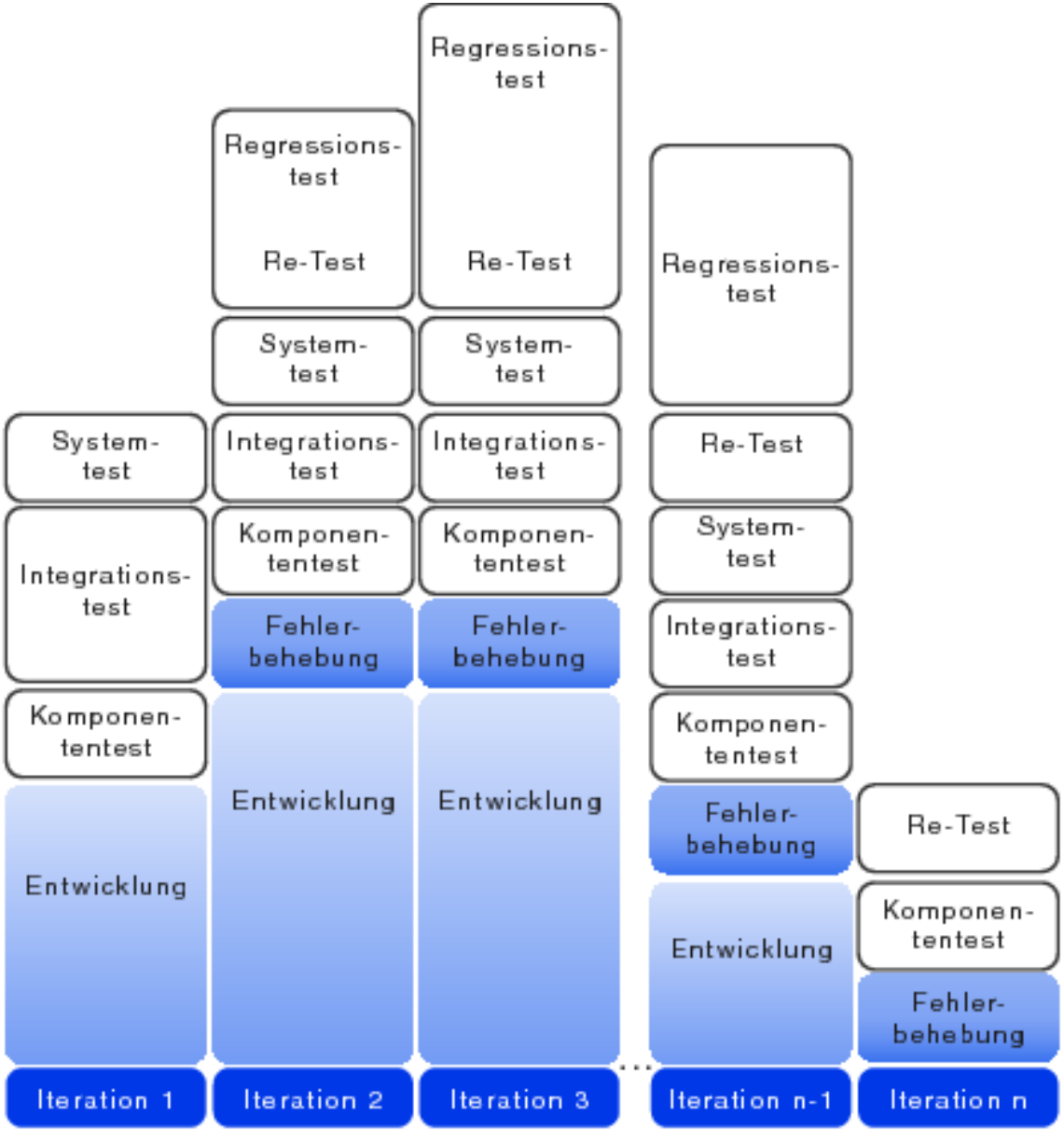
Teststufenkonzept → A-Symmetriemodell



Teststufen

- Unit-Tests
- Komponenten-Tests
- Integrations-Tests
- System-Tests
- Akzeptanz-Tests

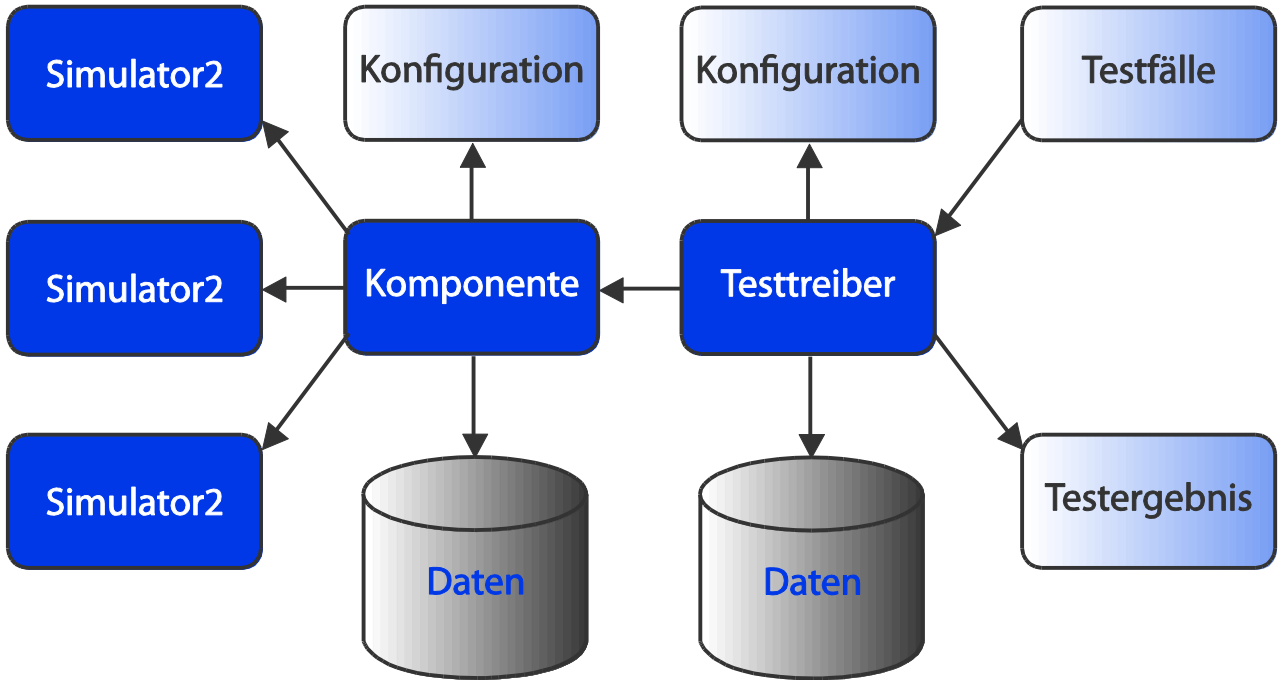
Testen in der iterativen Entwicklung - Typische Teststufen



Komponententest

Prüfung von separat testbaren Komponenten (z.B. Modulen, Programmen, Objekten, Klassen) auf vorhandene Fehler

Isolation der einzelnen Komponenten vom Rest des Systems mit Hilfe von Platzhaltern (Stubs) bzw. Simulatoren und Testtreiber

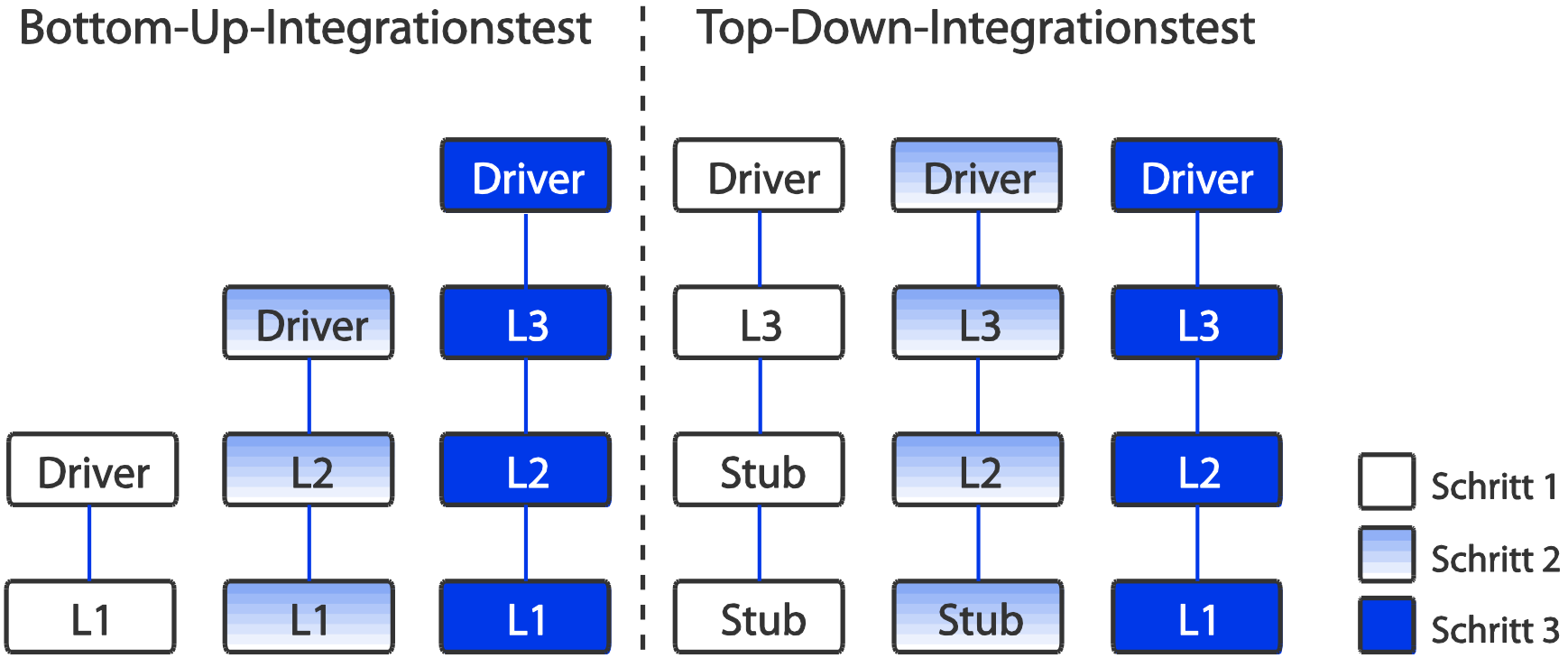


Integrationstest

- Prüft die Schnittstellen zwischen Komponenten
- Es können mehrere Integrationsstufen zum Einsatz gelangen, wobei diese Testobjekte unterschiedlichster Größe betreffen können
- Mit der Größe des Integrationsumfangs wächst auch die Schwierigkeit der Isolation von Fehlerwirkungen in Komponenten oder Systemen
- Inkrementelle Integrationsstrategien vs. Big-Bang-Strategie
- Können auf der Systemarchitektur, auf funktionalen Aufgaben, Transaktionsverarbeitungssequenzen oder weiteren Aspekten des Systems oder seiner Komponenten basieren

Integrationstest cont.

Bottom-Up-Integrationstests vs. Top-Down-Integrationstests



Systemtest

Fokus liegt im spezifizierten Verhalten eines Gesamtsystems oder eines Produktes

Systemtests sollten funktionale und nichtfunktionale Anforderungen an das System abdecken

Systemtests basieren auf:

- Anforderungsspezifikationen
- Anwendungsfällen oder sonstigen Beschreibungen eines Systems
- Geschäftsprozessen
- Risikoanalysen
- Erfahrungen im Produktionsumfeld
- Systemressourcen

Akzeptanztest

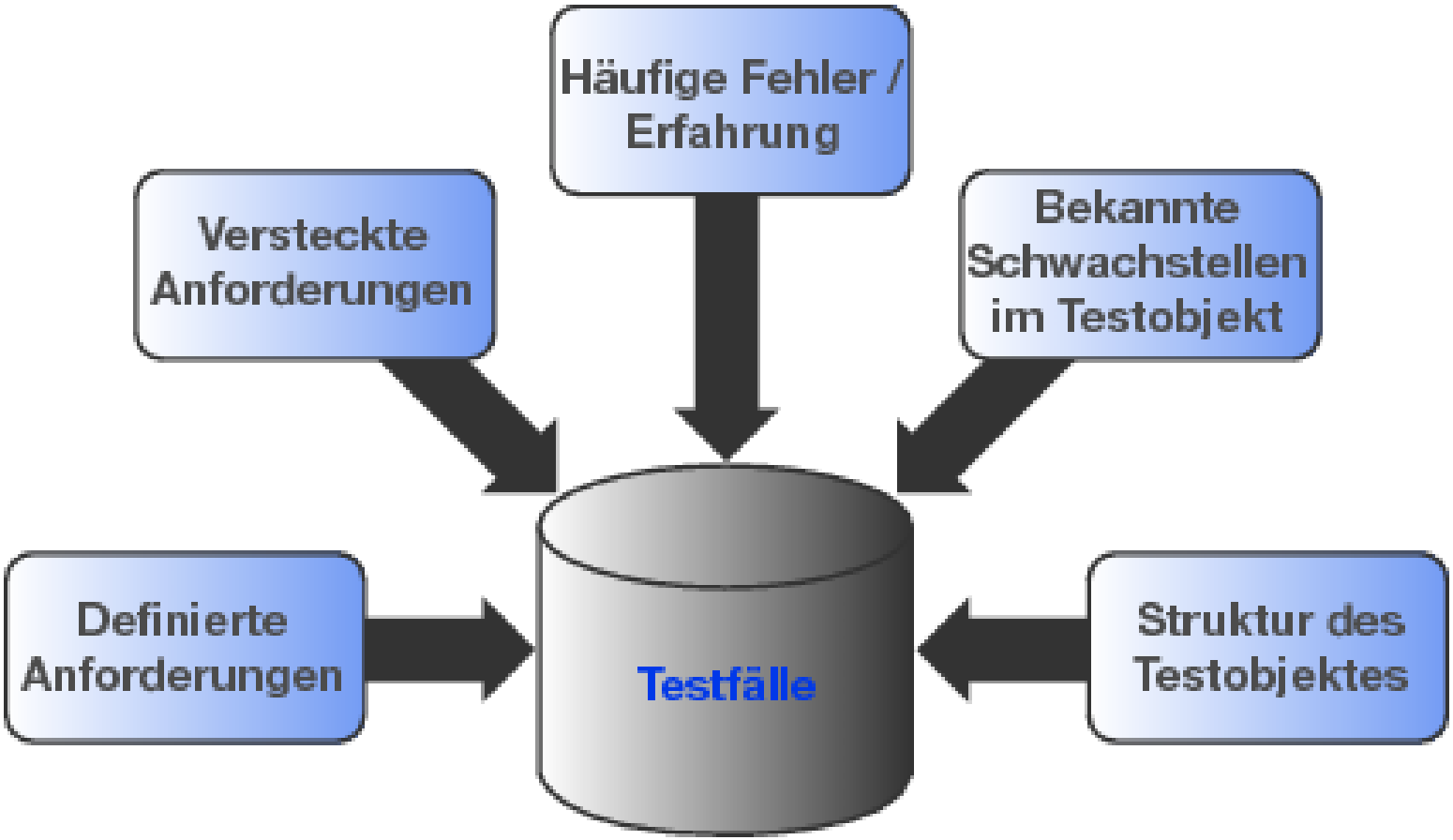
Fokus des Abnahmetests liegt darin, die Erbringung der vom Auftraggeber und Auftragnehmer vereinbarten Leistungen nachzuweisen

Wird meist von Kunden oder Benutzern eines Systems durchgeführt

Arten von Abnahmetests:

- Anwender-Abnahmetest
- Betrieblicher Abnahmetest
- Regulatorischer und vertraglicher Abnahmetest
- Alpha- und Beta-Test (oder Feldtest)

Typische Quellen für Testfälle

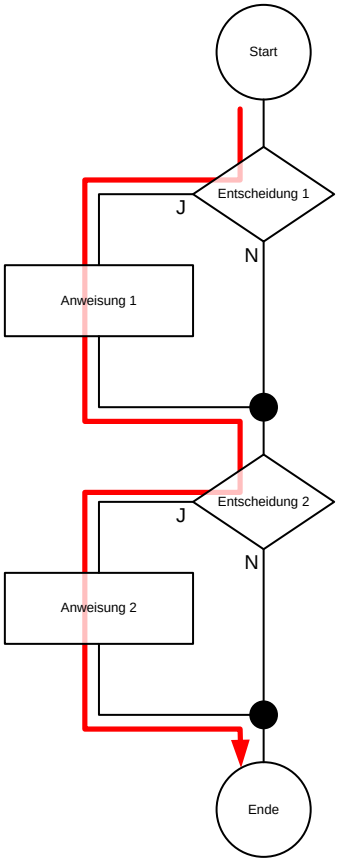


Testabdeckung

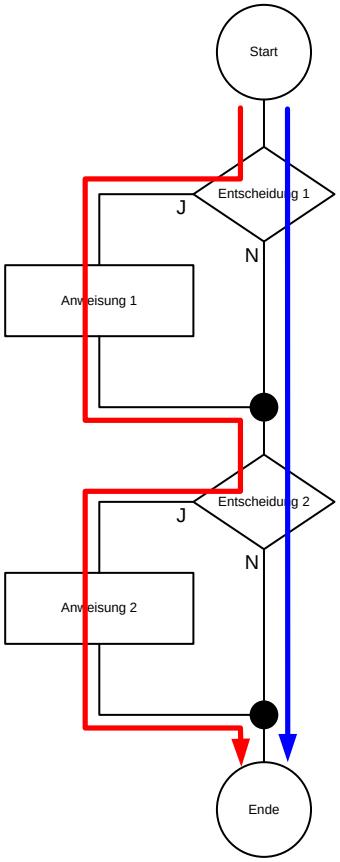
- Ziel der Verifikation von Softwaresystemen ist eine möglichst hohe Abdeckung (Coverage) des Systems mit den entsprechenden Testfällen
- Testfallselektion notwendig, da eine vollständige taxative Aufzählung aller möglichen Systemzustände aufgrund der Komplexität von Systemen unmöglich ist
- Ziel ist es, mittels formeller und deterministischer Verfahren die Menge an Testfällen zu identifizieren, die mit höchster Wahrscheinlichkeit Fehler finden bzw. die größtmögliche Abdeckung erreichen
- Zwei Ansätze von Abdeckung: strukturelle Abdeckung (White-Box-Tests) und funktionale Abdeckung (Black-Box-Tests)

Strukturelle Methoden

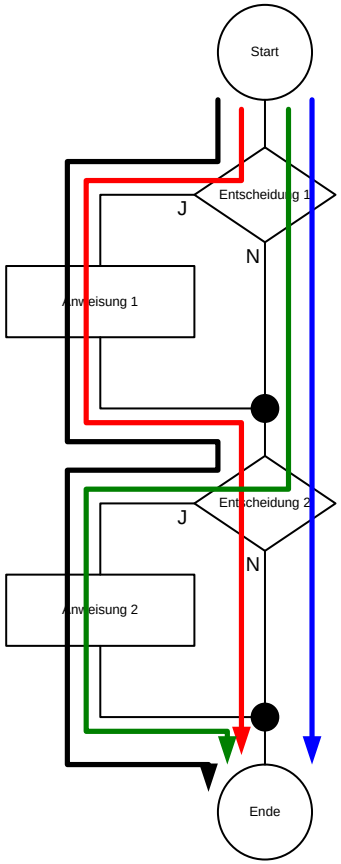
Anweisungsabdeckung



Bedingungsabdeckung



Pfadabdeckung



Funktionale Methoden

Äquivalenzklassenanalyse

- Einteilung möglicher Ein- und Ausgabewerte in Klassen gleichem Systemverhalten
- Es werden Werte von Klassen gewählt, bei denen angenommen wird, dass Fehler auftreten, die auch bei allen anderen möglichen Werten der Klasse auftreten können

Grenzwertanalyse

- Spezialfall der Äquivalenzklassenanalyse
- Tatsache, dass Fehler besonders oft an den Grenzen der Äquivalenzklassen auftreten

Beispiel Äquivalenzklassen- und Grenzwertanalyse

- One of the fields on a form contains a text box which accepts numeric values in the range of 18 to 25. Identify the invalid Equivalence class.

- a) 17
  b) 19
  c) 24
  d) 21

Beispiel Äquivalenzklassen- und Grenzwertanalyse

- **Solution:**

- **Class I: values < 18 => invalid class**
  Class II: 18 to 25 => valid class
  Class III: values > 25 => invalid class

- **17 fall under invalid class. 19, 24 and 21 fall under valid class. So answer is ‘A’**

Beispiel Äquivalenzklassen- und Grenzwertanalyse

- **Question 1**
  In an Examination a candidate has to score minimum of 24 marks in order to clear the exam. The maximum that he can score is 40 marks. Identify the Valid Equivalence values if the student clears the exam.

- **a) 22,23,26**
  b) 21,39,40
  c) 29,30,31
  d) 0,15,22

Equivalence partitioning

- **Solution**
  The classes will be as follows:
  Class I: values < 24 => invalid class
  Class II: 24 to 40 => valid class
  Class III: values > 40 => invalid class
- **We have to indentify Valid Equivalence values. Valid Equivalence values will be there in Valid Equivalence class. All the values should be in Class II. So answer is 'C'**

Equivalence partitioning

- **Question 2**
  One of the fields on a form contains a text box which accepts alpha numeric values. Identify the Valid Equivalence class

- **a) BOOK**
  b) Book
  c) Boo02k
  d) Book

Equivalence partitioning

- Solution
- Alpha numeric is combination of alphabets and numbers. Hence we have to choose an option which has both of these. A valid equivalence class will consist of both alphabets and numbers. Option ‘c’ contains both alphabets and numbers. So answer is ‘C’

Equivalence partitioning

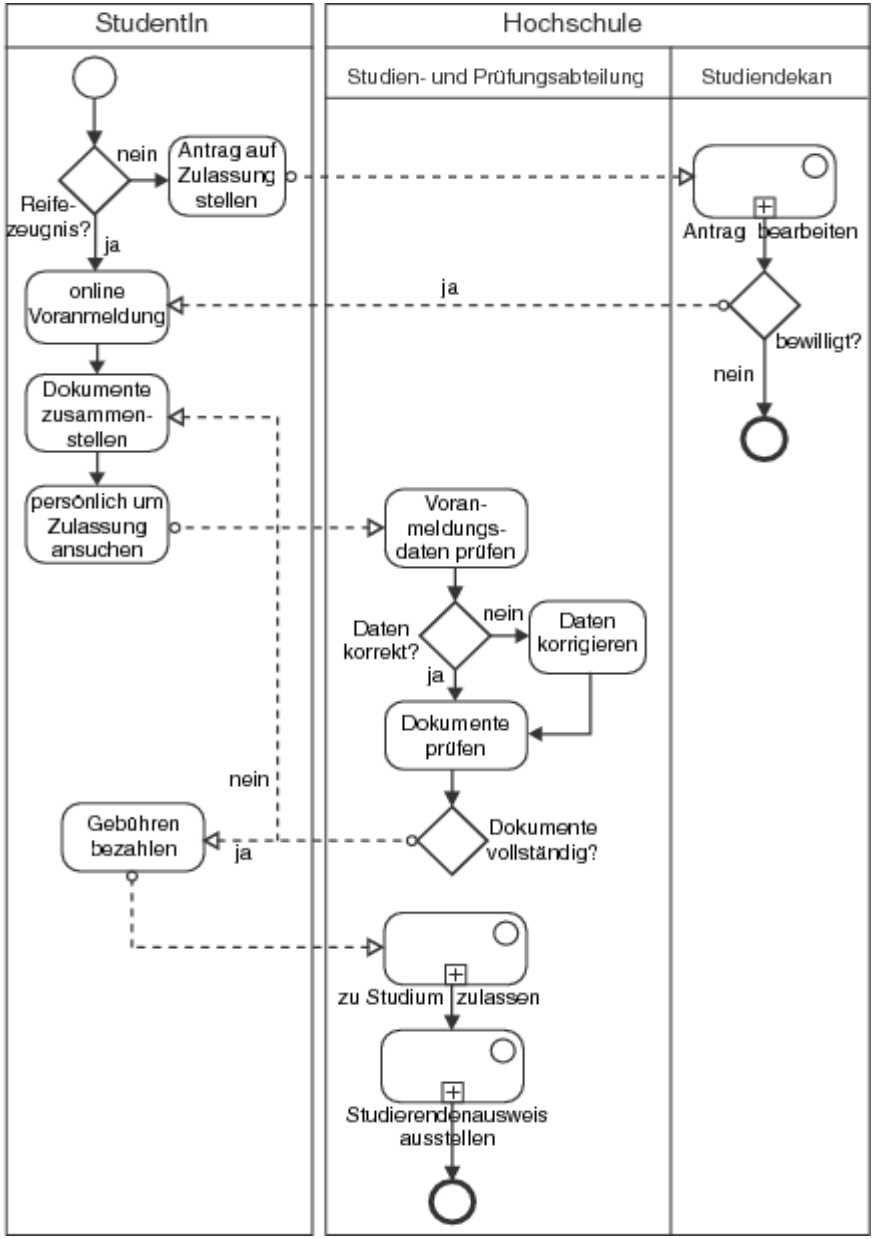
- **Question 3**
  The Switch is switched off once the temperature falls below 18 and then it is turned on when the temperature is more than 21.
- **Identify the Equivalence values which belong to the same class.**
- **a) 12,16,22**
  b) 24,27,17
  c) 22,23,24
  d) 14,15,19

Equivalence partitioning

- **Solution**
  We have to choose values from same class (it can be valid or invalid class). The classes will be as follows:
- **Class I: less than 18 (switch turned off)**
  Class II: 18 to 21
  Class III: above 21 (switch turned on)
- **Only in Option ‘c’ all values are from one class. Hence the answer is ‘C’. (Please note that the question does not talk about valid or invalid classes. It is only about values in same class)**

Zustandsbasierte Methoden

- **Zustandsbasierte Testmethoden**
  - Basieren auf Zustandsautomaten, die oft als UML-Zustandsdiagramme oder Prozessgrafiken dargestellt werden
  - Coverage:
    - State-Coverage,
    - Transition-Coverage,
    - Event-Coverage,
    - Path-Coverage

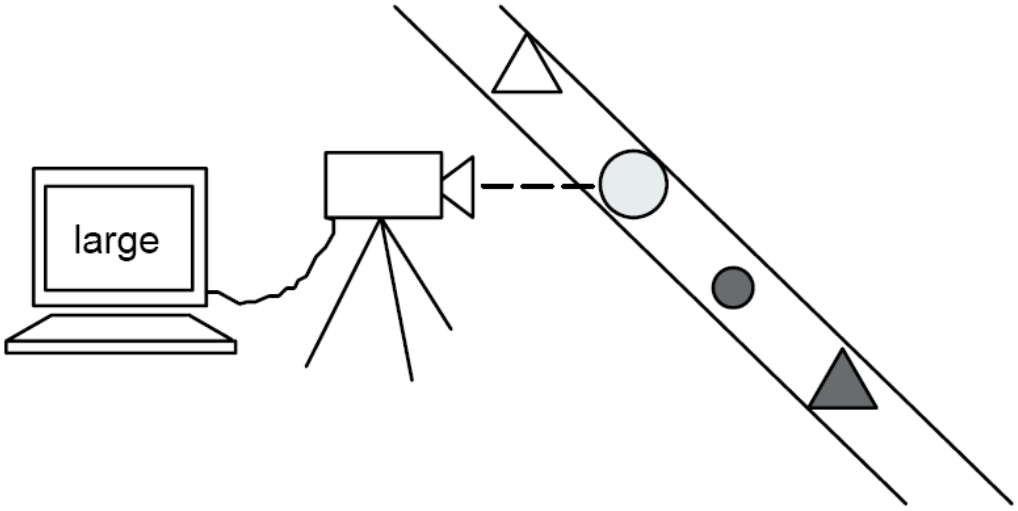


Funktionale Methoden

Klassifikationsbaummethode

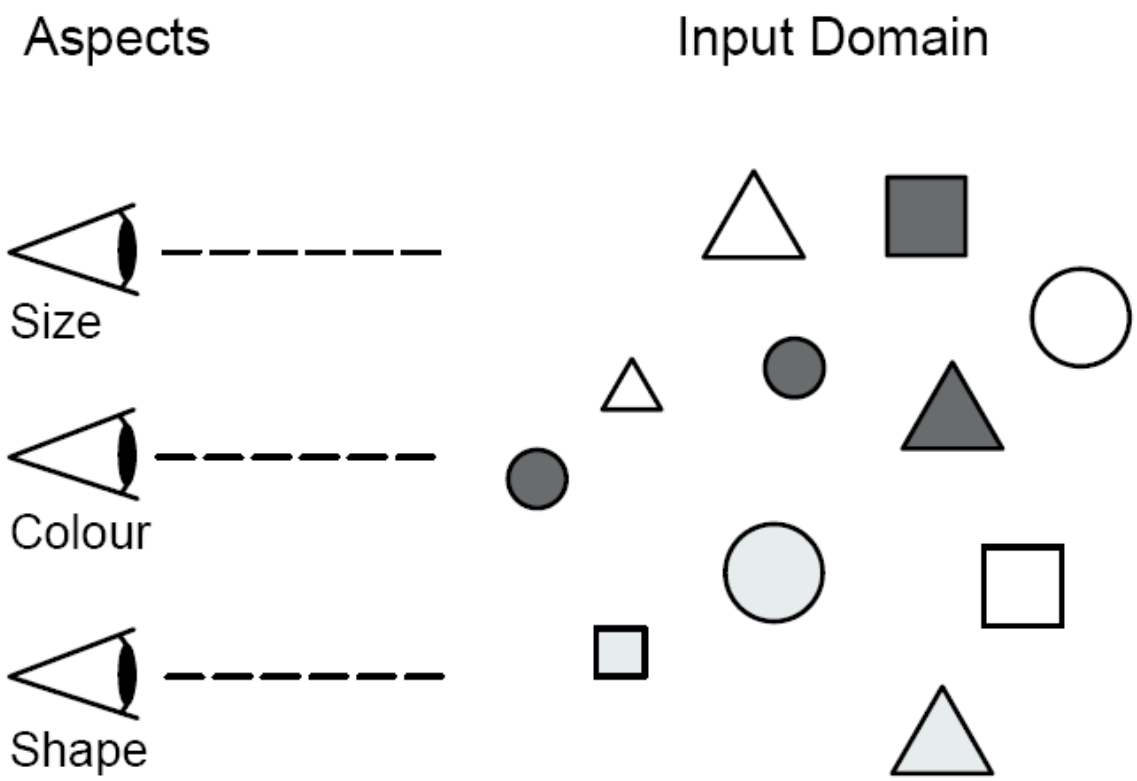
- Die Klassifikationsbaummethode wird zur systematischen Unterstützung von Blackbox-Tests angewandt
- Es werden die Testfälle anhand möglicher Eingabebereiche oder Systemzustände in sogenannten Klassifikationen eingeteilt
- Klassifikationen werden wiederum in disjunkte Teilmengen zerlegt
- Testfälle werden durch Kombination verschiedener Klassen erstellt, wobei von jeder Klassifikation nur eine Klasse gewählt wird
- Folgende Kombinationen von Klassen sind bei der Erstellung von Testfällen möglich:
  - **Minimale Kombination**: Jede Klasse jeder Klassifikation wird mindestens einmal verwendet.
  - **Maximale Kombination**: Jede Klasse wird mit jeder Klasse aus anderen Klassifikationen kombiniert.
  - **Paarweise Kombination**: Jeweils zwei Klassifikationen werden vollständig miteinander kombiniert.
  - **Tripelweise Kombination**: Analog zur paarweisen Kombination werden jeweils drei Klassifikationen miteinander vollständig kombiniert.

Classification-Tree Method

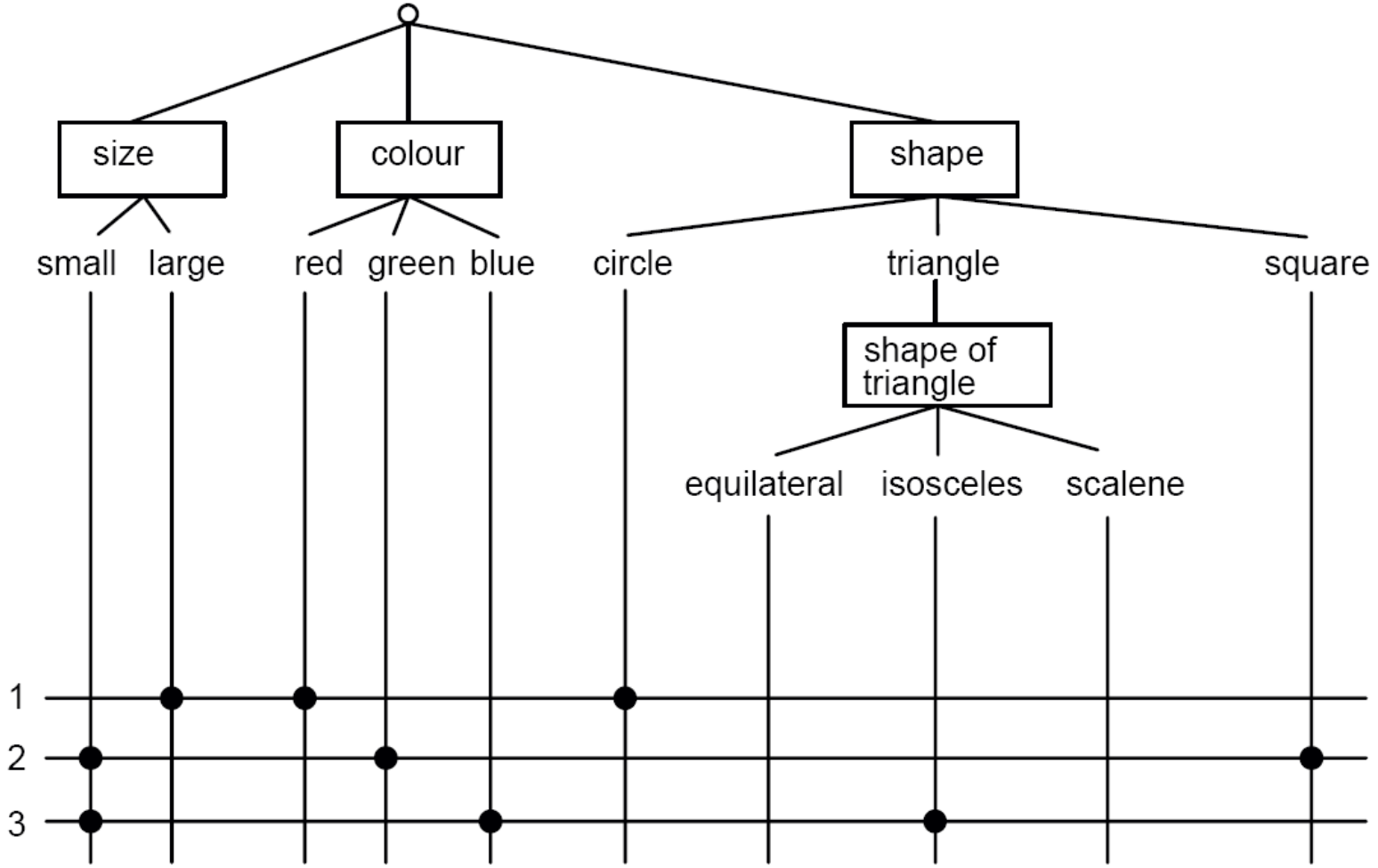


Classification-Tree Method

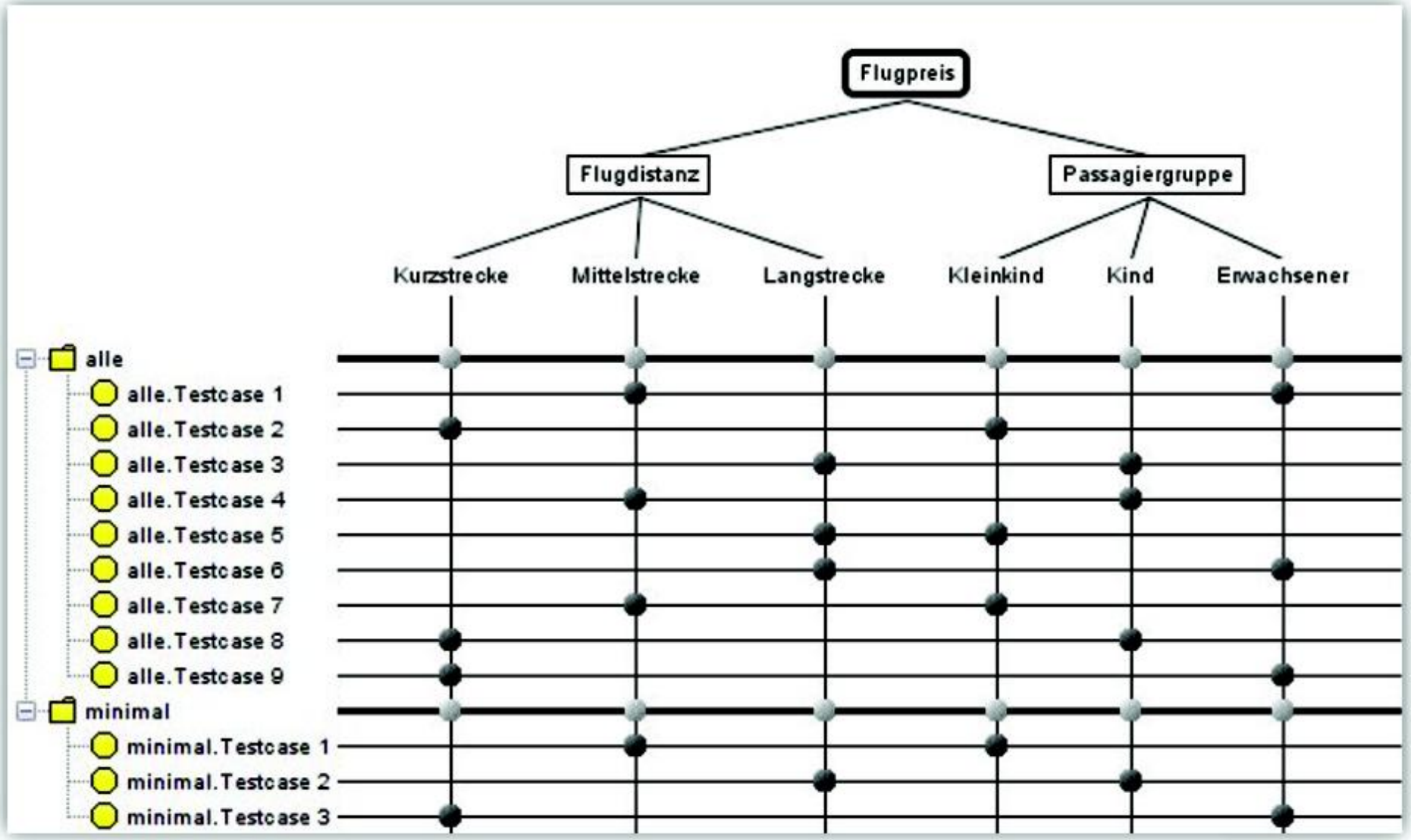
Aspects for Classification



Classification-Tree Method



Beispiel Klassifikationsbaummethode



Informelle Testmethoden

- Es wird ganz oder teilweise auf eine systematische Ableitung von Testfällen verzichtet
- Die Durchführung der Tests erfolgt mehr oder minder nach der Intuition des Testers
- Informelle Testmethoden beruhen auf der Erfahrung der Tester und sind daher nicht deterministisch und nicht exakt wiederholbar, jedoch sehr oft effektiv in der Fehlerfindung
- In der Praxis werden informelle und formelle Methoden daher häufig kombiniert
- Die Idee der explorativen Testmethode ist gleichzeitiges Lernen und Testen eines Testobjektes
- Informationen, Erfahrungen und Ergebnisse aus diesen Tests fließen anschließend direkt in neue, verbesserte Testfälle ein

Nichtfunktionale Softwaretests

Der Test für nichtfunktionale Eigenschaften ist in Abhängigkeit vom jeweils untersuchten Qualitätsattribut aufgebaut

Zu den wichtigsten nichtfunktionalen Eigenschaften, für die es auch spezialisierte Testverfahren gibt, zählen die Benutzbarkeit (Usability), die Leistungsfähigkeit (Performance) und die Sicherheit (Security) eines Systems

Der Test der nichtfunktionalen Eigenschaften erfordert mehr Expertenwissen zu den jeweiligen Testarten als funktionale

Efficiency Testing I

The efficiency quality attribute is evaluated by conducting tests focused on time and resource behavior.

Efficiency testing relating to time behavior is covered by performance, load, stress and scalability testing.

Performance Testing:

Focus on the ability of a component or system to respond to user or system inputs within a specified time and under specified conditions. Two main facilities:

- Load generation
- Measurement and analysis of system response to a given load

Measurements have to be taken between individual components so that performance “bottlenecks” can be identified.

Efficiency Testing II

Load Testing:

Focus on the ability of a system to handle increasing levels of anticipated realistic loads.

Transaction requests are generated by numbers of parallel users under different scenarios of typical use.

Two sub-types of load testing:

- Multi-user testing (with realistic numbers of users)
- Volume testing (with large numbers of users)

Stress Testing:

Focus on the ability of a system to handle peak loads at or beyond maximum capacity.

Sub-types of stress testing:

- Spike testing (sudden extreme load being placed)
- Bounce testing (applying several spikes – test claim/release of recourses)

Efficiency Testing III

Other efficiency testing methods:

Scalability testing (test the ability of a system to meet future efficiency requirements)

Test of Resource Utilization (evaluate the usage of system resources – e.g. memory space, disk capacity and network bandwidth)

Efficiency Test Specification:

Specification of tests based on *operational profiles*

OP's represent distinct forms of user behavior when interacting with an application.

Testautomatisierung

Vollautomatische Durchführung und Verifikation von Testfällen

In erster Linie eine wirtschaftliche Überlegung

Initialaufwand für eine automatisierte Durchführung von Testfällen ist in der Regel wesentlich höher als der Initialaufwand für einen manuellen Test

Im Gegenzug ist jedoch der Aufwand pro Testlauf im automatisierten Fall deutlich niedriger

Der Testvorbereitungsaufwand ist im automatisierten Fall etwa doppelt so hoch ist wie im manuellen Fall. Der Aufwand für die Testdurchführung ist jedoch sehr viel geringer im automatisierten Fall

Wartungsaufwand von automatisierten Testfällen im Allgemeinen höher ist als für manuelle Testfälle

Wirtschaftlichkeit der Testautomatisierung

	Testvorbereitung			Testdurchführung (pro Testlauf)		
	Durchschn.	Min.	Max.	Durchschn.	Min.	Max.
Automatisiert	19.2 h	10.6 h	56.0 h	0.21 h	0.1 h	1.0 h
Manuell	11.6 h	10.0 h	20.0 h	3.93 h	0.5 h	24.0 h

Tabelle 7.1: Studie von Dustin et al. (1999) zum Aufwand von manuellen und automatisierten GUI-Softwaretests

Automatisierte Komponententests

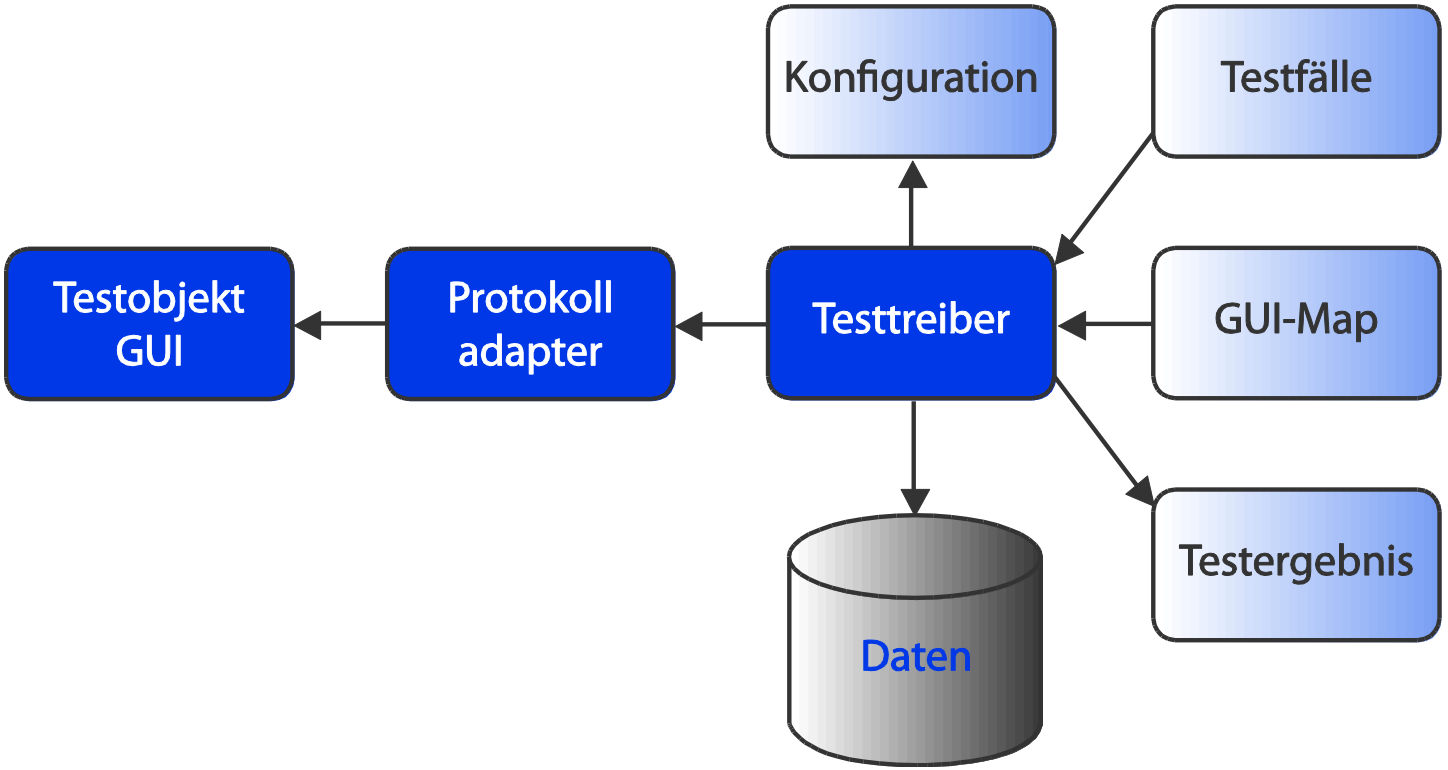
- Automatisierte Komponententests auf Quellcode- oder Schnittstellenebene werden mit speziellen Test-Frameworks durchgeführt
- Moderne Unit-Test-Frameworks bieten die Möglichkeit, einen Komponententest direkt in der Entwicklungsumgebung zu definieren und ständig parallel zur Entwicklung durchzuführen bzw. auszuwerten
- Eine wesentliche Funktion eines Unit-Test-Frameworks ist die Bereitstellung von Verifikationsmethoden (assertions), die den exakten Bereich des Testfalls definieren
- Unit-Test-Frameworks sind häufig mit Code-Coverage-Frameworks gekoppelt, die mithilfe von Instrumentierung die strukturelle Abdeckung der Unit-Tests feststellen

Automatisierte GUI-Tests

- Simulation des Benutzers über die Benutzerschnittstelle des Systems
- GUI-Tests sind dadurch definiert, dass sowohl der Point of Control (PoC) als auch der Point of Observation (PoO) die Benutzerschnittstelle des Softwaresystems ist
- Ein übliches Verfahren zur Bestimmung der Aufbauelemente eines GUI-Tests ist das Capture/Replay-Verfahren
- Automatisierte GUI-Tests können auch wie normaler Quellcode von Grund auf entwickelt werden (Scripting)

Automatisierte GUI-Tests

Die einzelnen GUI-Protokolle (z.B. Web, Win32, SAP) werden in Protokolladaptern implementiert und sind damit unabhängig vom Testtreiber und den Testfällen selbst



Testdaten

Testdaten zur Konfiguration des Testaufbaus

- Zielsysteme und Testtreiber müssen mit geeigneten Testdaten vorkonfiguriert werden

Datengetriebener Test → Trennung von Testfällen und Testdaten

- Wiederverwendbarkeit der Testfälle und Testdaten getrennt voneinander
- Die Wartbarkeit und Lesbarkeit der Testfälle und Testdaten wird erhöht.
- Testfälle und Testdatenmengen können in m:n Verwendung stehen.
- Testfälle können definiert werden, auch wenn die konkreten Testdaten noch nicht bekannt sind.
- Eine Automatisierung der Testfälle ist leichter herzustellen bzw. es ergeben sich besser zu wartende automatisierte Tests.

Strategie: Deterministisch versus Random

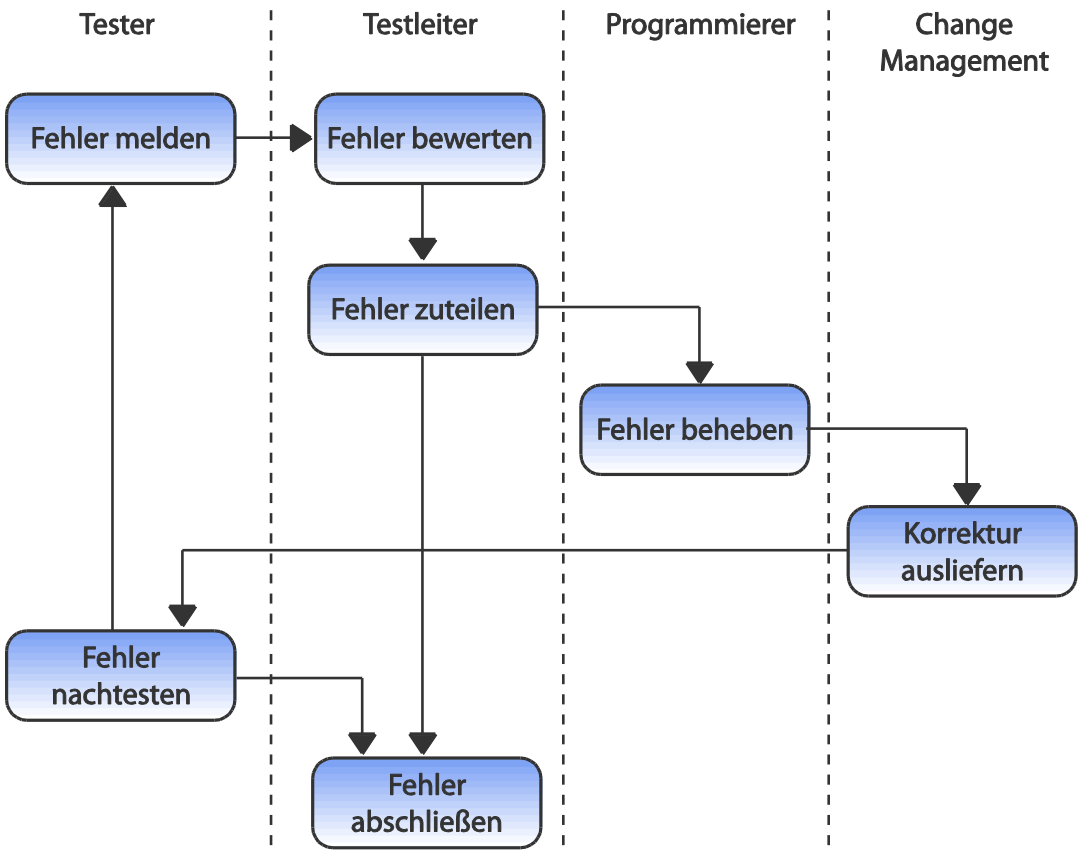
Testmanagement

- Teststrategie
- Testplanung
- Testanalyse & -design
- Testausführung & -protokollierung
- Fehleranalyse & -bewertung
- Testfortschrittsmonitoring & -überwachung
- Auswertungen

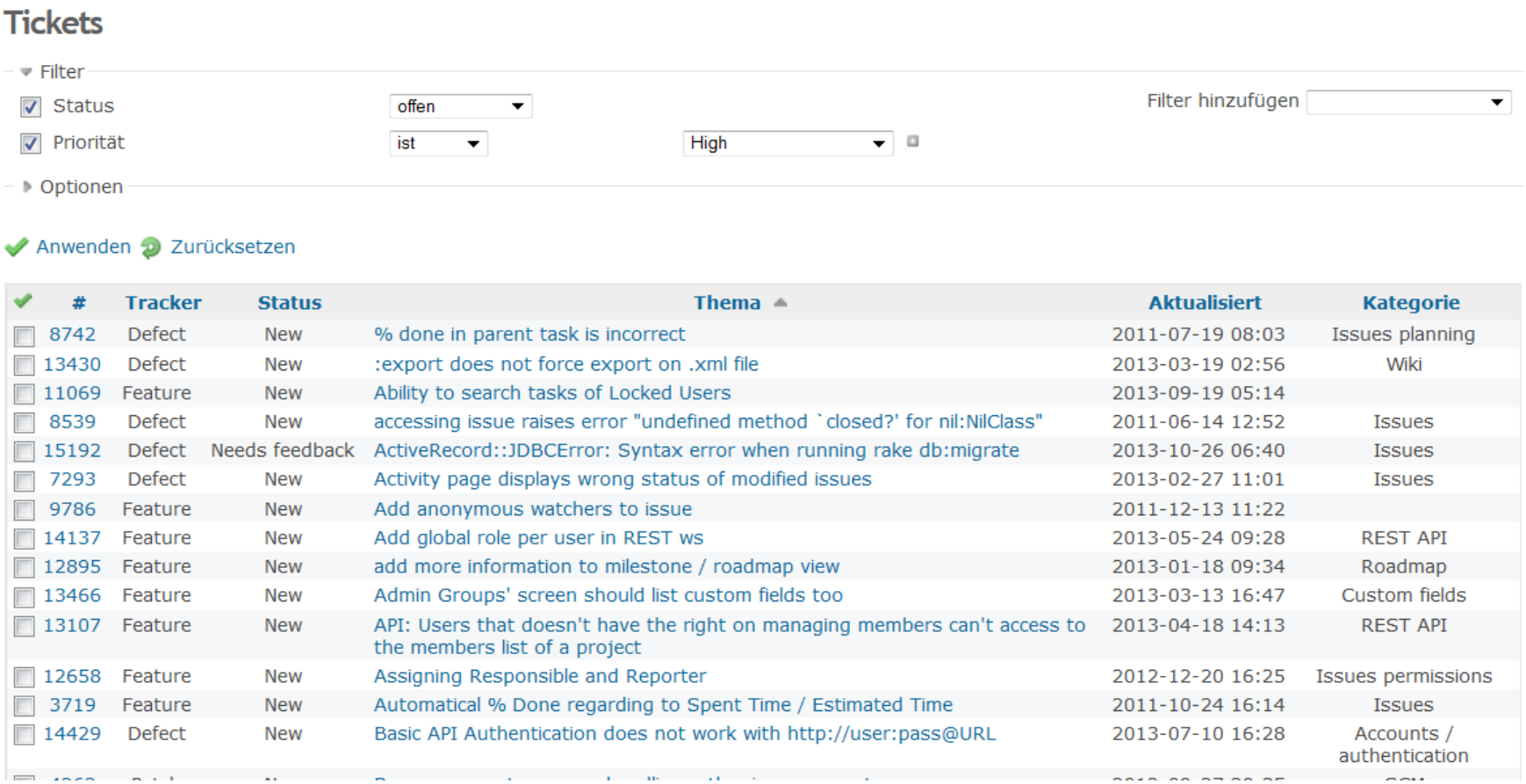
Fehlermanagement

Typischer Fehlermanagement Prozess

1. Fehler melden
2. Fehler bewerten
3. Fehler zuteilen
4. Fehler beheben
5. Korrektur ausliefern
6. Fehler nachtesten
7. Fehler abschließen



Issue bzw Bug Tracking System



Tickets

Filter

Status: offen

Priorität: ist: High

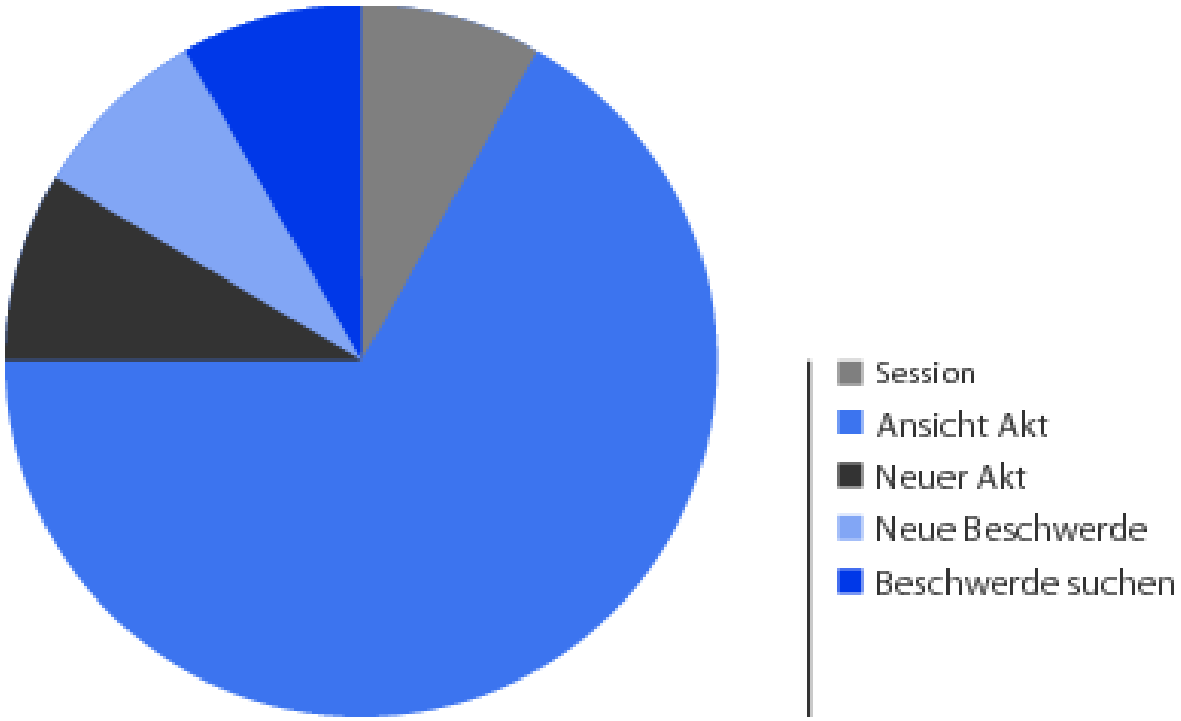
Filter hinzufügen

Optionen

Anwenden Zurücksetzen

#	Tracker	Status	Thema	Aktualisiert	Kategorie
8742	Defect	New	% done in parent task is incorrect	2011-07-19 08:03	Issues planning
13430	Defect	New	:export does not force export on .xml file	2013-03-19 02:56	Wiki
11069	Feature	New	Ability to search tasks of Locked Users	2013-09-19 05:14	
8539	Defect	New	accessing issue raises error "undefined method `closed?' for nil:NilClass"	2011-06-14 12:52	Issues
15192	Defect	Needs feedback	ActiveRecord::JDBCError: Syntax error when running rake db:migrate	2013-10-26 06:40	Issues
7293	Defect	New	Activity page displays wrong status of modified issues	2013-02-27 11:01	Issues
9786	Feature	New	Add anonymous watchers to issue	2011-12-13 11:22	
14137	Feature	New	Add global role per user in REST ws	2013-05-24 09:28	REST API
12895	Feature	New	add more information to milestone / roadmap view	2013-01-18 09:34	Roadmap
13466	Feature	New	Admin Groups' screen should list custom fields too	2013-03-13 16:47	Custom fields
13107	Feature	New	API: Users that doesn't have the right on managing members can't access to the members list of a project	2013-04-18 14:13	REST API
12658	Feature	New	Assigning Responsible and Reporter	2012-12-20 16:25	Issues permissions
3719	Feature	New	Automatical % Done regarding to Spent Time / Estimated Time	2011-10-24 16:14	Issues
14429	Defect	New	Basic API Authentication does not work with http://user:pass@URL	2013-07-10 16:28	Accounts / authentication

Beispiel für die Fehlerverteilung in Stadt21



Vielen Dank für Ihre Aufmerksamkeit!