

IT Quality and Software Test

Lesson 5 Test Design Techniques Dynamic Testing I V1.1

Uwe Gühl



Winter 2011/ 2012



Contents

- Test Design Techniques – Dynamic Testing I
 - Test Development Process
 - Categories of Test Design Techniques
 - Black-box Techniques
(or Specification-based Techniques)
 - Equivalence Partitioning
 - Boundary Value Analysis
 - Decision Table Testing
 - State Transition Testing
 - Use Case Testing
 - Comparison Black-box / White-box Techniques



Test Development Process

- The level of formality of the test development process depends on the context of the testing, including
 - maturity of testing and development processes,
 - time constraints,
 - safety or regulatory requirements,
 - people involved.



Test Development Process

Test analysis

- Analysis of test basis documentation,
- determine what to test,
- identify test conditions.



Test Development Process

- Definition: A test condition is an item or event that could be verified by one or more test cases; e. g.
 - function,
 - transaction,
 - quality characteristic,
 - structural element.
- Requested: Bidirectional traceability between Test conditions \Leftrightarrow Specifications and requirements
 - for impact analysis when requirements change,
 - to determine requirements coverage for a set of tests.



Test Development Process

Test design: Creation and specification of test cases and test data

- A test case consists of a set of
 - input values,
 - execution preconditions,
 - **expected results,** and
 - execution postconditions,to cover a certain test objective(s) or test condition(s).
- The 'Standard for Software Test Documentation' (IEEE STD 829-1998) describes the content of test design specifications (containing test conditions) and test case specifications.



Test Development Process

Expected results

- Description of expected results should include
 - outputs,
 - changes to data and states,
 - any other consequences of the test.
- If expected results are not defined, then a plausible, but erroneous, result may be interpreted as the correct one.
- Expected results should ideally be defined before tests get executed.



Test Development Process

Test implementation

Activities (IEEE STD 829-1998):

- Test Cases are
 - developed,
 - implemented,
 - prioritized and
 - organized in the test procedure specification.
- Test Execution Schedules are
 - defined,
 - executed.



Test Development Process

- Test execution schedule
 - contents and defines the execution order of
 - test procedures
 - ... specifies the sequence of actions for a test execution
 - automated test procedures (automated test scripts)
 - ... if a test automation tool is used, contents sequence of actions
 - takes into account factors like
 - regression tests,
 - prioritization,
 - technical dependencies,
 - logical dependencies.



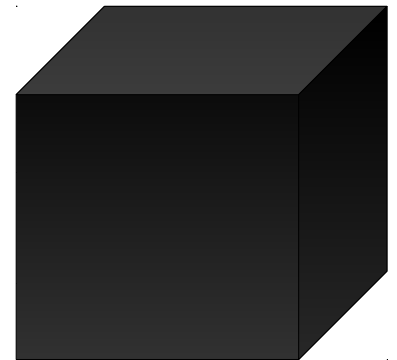
Test Design Techniques

- Purpose of a test design technique is to identify
 - test conditions,
 - test cases, and
 - test data.



Test Design Techniques

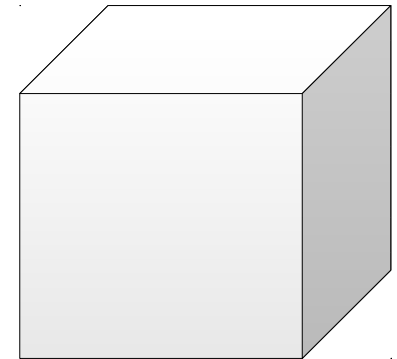
- Black-box test design techniques (also called specification-based techniques)
 - are based on an analysis of the test basis documentation,
 - include both functional and non-functional testing.
- Black-box testing, by definition, does **not use** any information regarding the internal structure of the component or system to be tested





Test Design Techniques

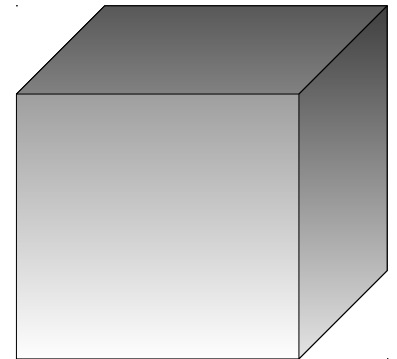
- White-box test design techniques (also called structural or structure-based techniques) are based on an analysis of the structure of the component or system.
- White-box testing does **use** any information regarding the internal structure of the component or system to be tested





Test Design Techniques

- Greybox test design techniques
 - Test Strategy based partly on internals of a software, involves knowledge of internal data structures and algorithms for purposes of designing tests,
 - execute defined tests at the user, or black-box level.
 - Idea: If you know something about the inside, you can test it better from outside
 - Important with web applications



http://en.wikipedia.org/wiki/Software_testing



Test Design Techniques

- Black-box and white-box testing may also be combined with experience-based techniques to leverage the experience of developers, testers and users to determine what should be tested.



Test Design Techniques

Specification-based test design techniques

- Models, either formal or informal, are used for
 - the specification of the problem to be solved,
 - the software, or
 - the software components.
- Test cases can be derived systematically from these models



Test Design Techniques

Structure-based test design techniques

- Information about how the software is constructed is used to derive the test cases (e.g., code and detailed design information).
- The extent of coverage of the software can be measured for existing test cases, and further test cases can be derived systematically to increase coverage.



Test Design Techniques

Experience-based test design techniques

- The knowledge and experience of people are used to derive the test cases
- The knowledge of testers, developers, users and other stakeholders about the software, its usage and its environment is one source of information
- Knowledge about likely defects and their distribution is another source of information



Black-box Techniques

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table Testing
- State Transition Testing
- Use Case Testing



Black-box Techniques

Equivalence Partitioning (1/4)

- Inputs to the software or system are divided into groups that are expected to exhibit similar behaviour.
- Equivalence partitions (or classes) can be found for
 - valid data, i.e., values that should be accepted,
 - invalid data, i.e., values that should be rejected.
- Partitions can also be identified for
 - outputs,
 - internal values,
 - time-related values (e.g., before or after an event),
 - interface parameters (e.g., integrated components being tested during integration testing).



Black-box Techniques

Equivalence Partitioning (2/4)

- Equivalence partitioning
 - is applicable at all levels of testing.
 - can be used to achieve input and output coverage goals.
 - can be applied to
 - human input,
 - input via interfaces to a system, or
 - interface parameters in integration testing.



Black-box Techniques

Equivalence Partitioning (3/4)

- Example: Input: Day of the week
Ideas
 - Working day
 - Weekend day
- Input: Day of the week as number
(1 = Monday, ..., 7 = Friday)
Ideas:
 - Valid: ≥ 1 and ≤ 7
 - Not valid: <1 or >7
 - Test: 0, 5, 14



Black-box Techniques

Equivalence Partitioning (4/4)

- Example: Input: Month; ideas:
 - Months with 30 days
 - Months with 31 days
 - February with 28 days (non leap year)
 - February with 29 days (leap year)
- Month as number
(1 = January, ..., 12 = December); ideas:
 - Valid: ≥ 1 and ≤ 12
 - Not valid: < 1 or > 12
 - Test: 0, 5, 14

Legend:

Red values: Not valid values

Green values: Valid values



Black-box Techniques

Boundary Value Analysis (1/3)

- Behaviour at the edge of each equivalence partition is more likely to be incorrect than behaviour within the partition
⇒ Defect detection probable

- The maximum and minimum values of a partition are its boundary values.

- Boundary of a valid partition is a valid boundary value.
- Boundary of an invalid partition is an invalid boundary value.

Design corresponding **tests** for each boundary value



Black-box Techniques

Boundary Value Analysis (2/3)

- Boundary value analysis
 - can be applied at all test levels.
 - is relatively easy to apply and its defect finding capability is high.
 - Is often considered as an extension of equivalence partitioning or other black-box test design techniques.
- Detailed specifications are helpful in determining the interesting boundaries.



Black-box Techniques

Boundary Value Analysis (3/3)

- Examples for usage:
 - Equivalence classes for user input on screen
 - time ranges (e.g., time out, transactional speed requirements)
 - table ranges (e.g., table size is 512*512).
- Example: Month as number (1 = January, ..., 12 = December); ideas:
 - Valid: ≥ 1 and ≤ 12
 - Not valid: < 1 or > 12
 - Boundaries Test: 0, 1, 12, 13

Legend:

Red values: Not valid values
Green values: Valid values



Black-box Techniques

Decision Table Testing (1/7)

- Decision tables
 - to capture system requirements that contain logical conditions,
 - to document internal system design.
 - to record complex business rules that should be implemented.
- When creating decision tables, the specification is analysed, and conditions and actions of the system are identified.



Black-box Techniques

Decision Table Testing (2/7)

- The input conditions and actions are most often boolean (TRUE - FALSE).
- The decision table contains the triggering conditions, often combinations of true and false for all input conditions, and the resulting actions for each combination of conditions.
- Each column of the table corresponds to a business rule that defines a unique combination of conditions and which result in the execution of the actions associated with that rule.



Black-box Techniques

Decision Table Testing (3/7)

- The coverage standard commonly used with decision table testing is to have at least one test per column in the table, which typically involves covering all combinations of triggering conditions.



Black-box Techniques

Decision Table Testing (4/7)

- Advantage of decision table testing
 - It creates combinations of conditions that otherwise might not have been exercised during testing.
- When to use?
When the action of the software depends on several logical decisions.



Black-box Techniques

Decision Table Testing (5/7)

- Example [Dus07]: 8 Test Cases in the beginning

	Visiting rules	1	2	3	4	5	6	7	8
Conditions	Infectious sickness	yes	yes	yes	yes	no	no	no	no
	Visit during visiting time	yes	yes	no	no	yes	yes	no	no
	Fever	yes	no	yes	no	yes	no	yes	no
Actions	No visit	x	x	x	x				
	Visit with nurse					x		x	x
	Visit max. 30 minutes							x	
	Regular visit						x		

Black-box Techniques

Decision Table Testing (6/7)

- Example [Dus07] Same impact

	Visiting rules	1	2	3	4	5	6	7	8
Conditions	Infectious sickness	yes	yes	yes	yes	no	no	no	no
	Visit during visiting time	yes	yes	no	no	yes	yes	no	no
	Fever	yes	no	yes	no	yes	no	yes	no
Actions	No visit	x	x	x	x				
	Visit with nurse					x		x	x
	Visit max. 30 minutes							x	
	Regular visit						x		



Black-box Techniques

Decision Table Testing (7/7)

- Example [Dus]: Reducing to 5 Test Cases

Visiting rules		1	5	6	7	8
Conditions	Infectious sickness	yes	no	no	no	no
	Visit during visiting time	-	yes	yes	no	no
	Fever	-	yes	no	yes	no
Actions	No visit	x				
	Visit with nurse		x		x	x
	Visit max. 30 minutes				x	
	Regular visit			x		



Black-box Techniques

State Transition Testing (1/6)

- A system may exhibit a different response depending on current conditions or previous history (its state).
 - ⇒ can be shown with a state transition diagram.
- It allows to view the software in terms of
 - its states,
 - transitions between states,
 - the inputs or events that trigger state changes (transitions),
 - the actions which may result from those transitions.



Black-box Techniques

State Transition Testing (2/6)

- The states of the system or object under test are separate, identifiable and finite in number.
- A state table shows the relationship between the states and inputs, and can highlight possible transitions that are invalid.



Black-box Techniques

State Transition Testing (3/6)

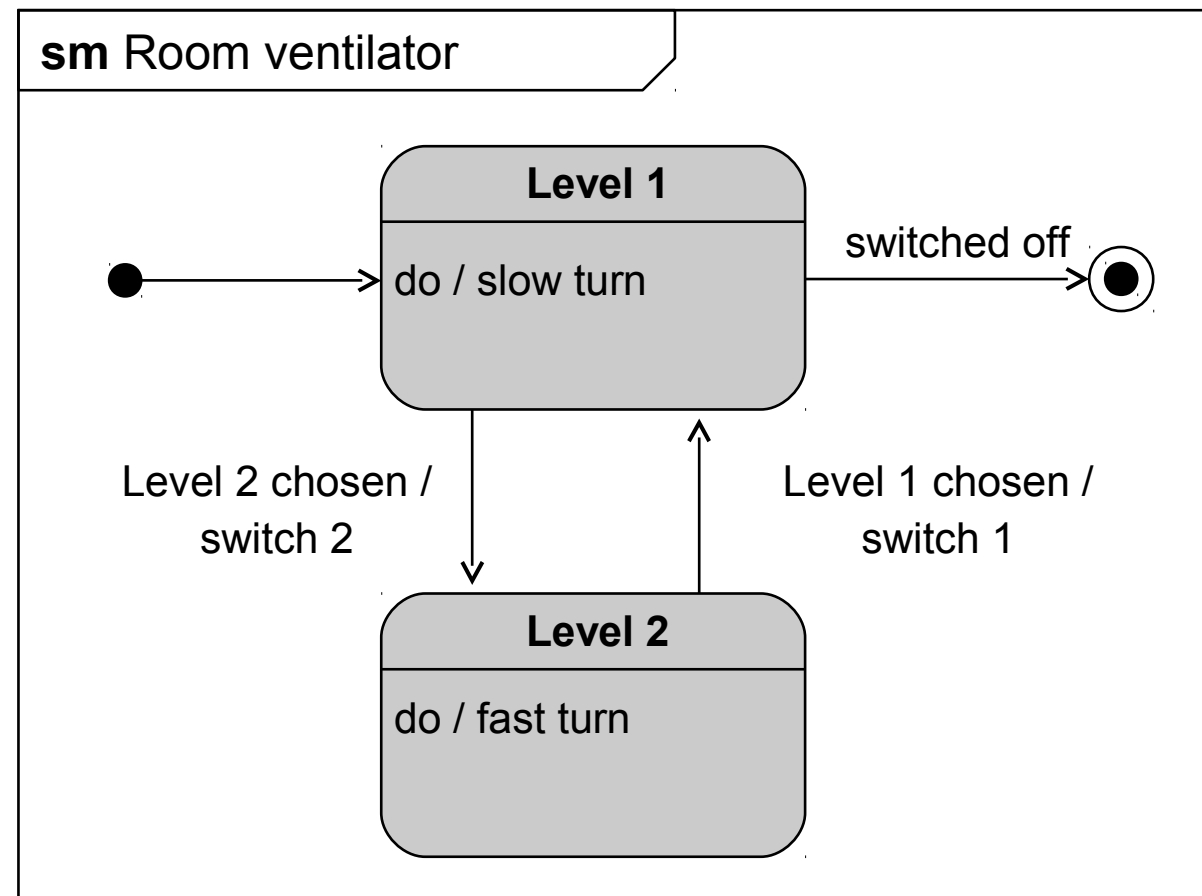
- Tests can be designed to
 - cover a typical sequence of states,
 - cover every state,
 - exercise every transition,
 - exercise specific sequences of transitions
 - test invalid transitions.
- Usage of State transition testing
 - embedded software,
 - technical automation,
 - modelling a business object with specific states (in business scenarios)
 - testing screen-dialogue flows (Web applications)



Black-box Techniques

State Transition Testing (4/6)

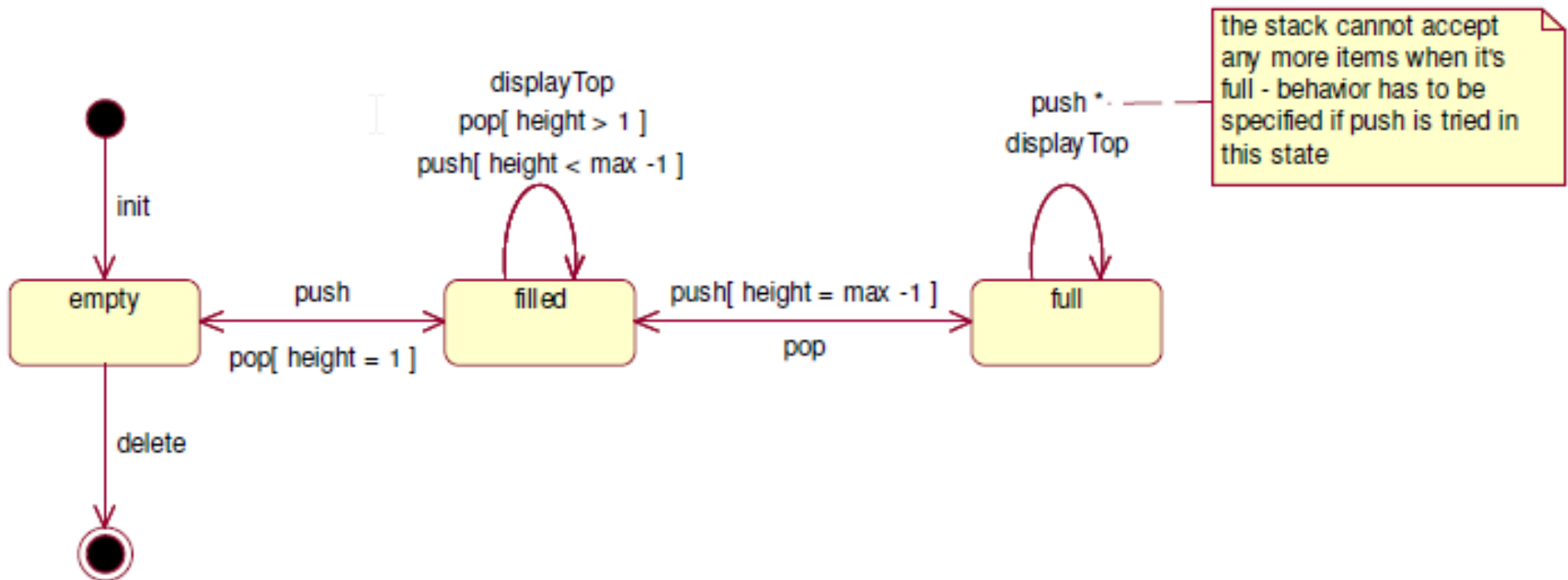
- Example 1: Fan



Black-box Techniques

State Transition Testing (5/6)

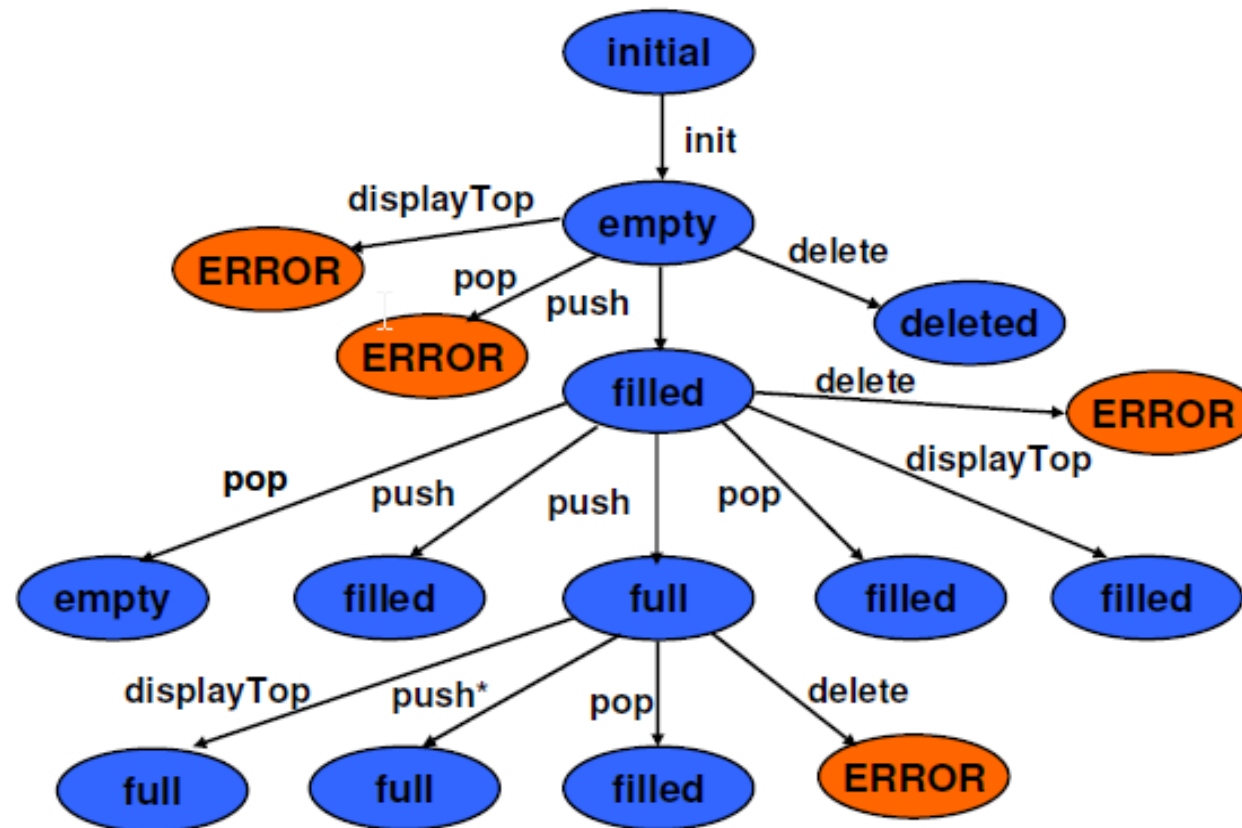
- Example 2: Vendor machine [Dus07]
- Status: „empty“, „filled“ (less then full), „full“



Black-box Techniques

State Transition Testing (6/6)

- Example 2: Transition Tree [Dus07]





Black-box Techniques

Use Case Testing (1/18)

- Tests can be derived from use cases.
- A use case describes interactions between actors (users or systems), which produce a result of value to a system user or the customer.
- Use cases may be described at the
 - abstract level (business use case, technology-free, business process level),
 - the system level (system use case on the system functionality level).



Black-box Techniques

Use Case Testing (2/18)

- Use Case has
 - a main scenario,
 - alternative scenarios,
 - failure scenarios.
 - preconditions which need to be met for the use case to work successfully.
 - postconditions which are the observable results and final state of the system after the use case has been completed.



Black-box Techniques

Use Case Testing (3/18)

Id / Name	214 / Rent a car
Short description	A customer comes to the car rental agency and chooses a car which he rents for a fixed period
Actors	Customer, agent
Trigger	Customer asks agent
Pre condition	The rental system is ready to get customer data and to realize a lease contract
Result	Leasing is done, and the customer has signed the contract
Post condition	The rental system is ready to get customer data and to realize a lease contract
Activities	<ol style="list-style-type: none">1. Enter customer data. If customer is yet not registered \Rightarrow UC 12 <i>Register customer</i>.2. Enter desired car category3. Enter desired leasing period4. If a car is available in the desired period:<ol style="list-style-type: none">1. Reserve a car2. Enter credit card information3. Print contract and signOtherwise: Adapt item 2. or 3., if possible

Example of a
use case description



Black-box Techniques

Use Case Testing (4/18)

- Use cases
 - describe the “process flows” through a system based on its actual likely use,
 - are very useful for designing acceptance tests with customer/user participation.
- Test cases derived from use cases detect
 - defects in the process flows during real-world use of the system.
 - integration defects caused by the interaction and interference of different components



Black-box Techniques

Use Case Testing (5/18)

- Test Case
 - Sequence of steps consisting of actions to be performed on the system under test [Bla04]
 - is the “basic unit” in Testing
 - serves to validate the functionality and to confirm the realization of a requirement
 - functional
 - non functional (quality criteria)
 - originates typically out of an Use Case
 - → Usually NFR-Test Cases are taken from regular Test Cases, if so simplification



Black-box Techniques

Use Case Testing (6/18)

- Test Case
 - describes the role who should execute it
 - contents Test Steps with
 - Activities of the tester
 - Input values
 - Expected output values
 - describes preconditions and postconditions



Black-box Techniques

Use Case Testing (7/18)

- Test Case – Example

- Test Case name „IU22_Create-Object“
- Test Case ID 7
- Priority 1
- Test classification Standard
- Preparation Hours 1
- Execution Hours 1
- Description Creation of an Object. The user must select an object
He has to decide which specific kind ...
- Risk Without Creation of objects Software can't be used
- Version 01
- is Test Case Chain []



Black-box Techniques

Use Case Testing (8/18)

- Test Case – Example (cont.) Condition

– Goal	Creation of a new object
– Prerequisites	Following objects must be available in database to execute this test case: * object A * object B
– Remarks	Function „select module“ is described in Test Case „IU21_Display-Object“



Black-box Techniques

Use Case Testing (9/18)

- Test Case – Example (cont.) Test Steps

– StepNo.	Description	Comments	Expected Result
– 10	Select an object in the tree structure		Selected Item will be highlighted
– 20	Choose „create“		Dialog box opens
– 30	Choose radio button		
– 40	Enter Obj ID		



Black-box Techniques

Use Case Testing (10/18)

- Following prioritization
Use Cases are typically prioritized
 - High priority: Must – no discussion
 - Medium priority: Should – necessary
 - Low priority: Could – Nice to have
- should be considered in derived test cases



Black-box Techniques

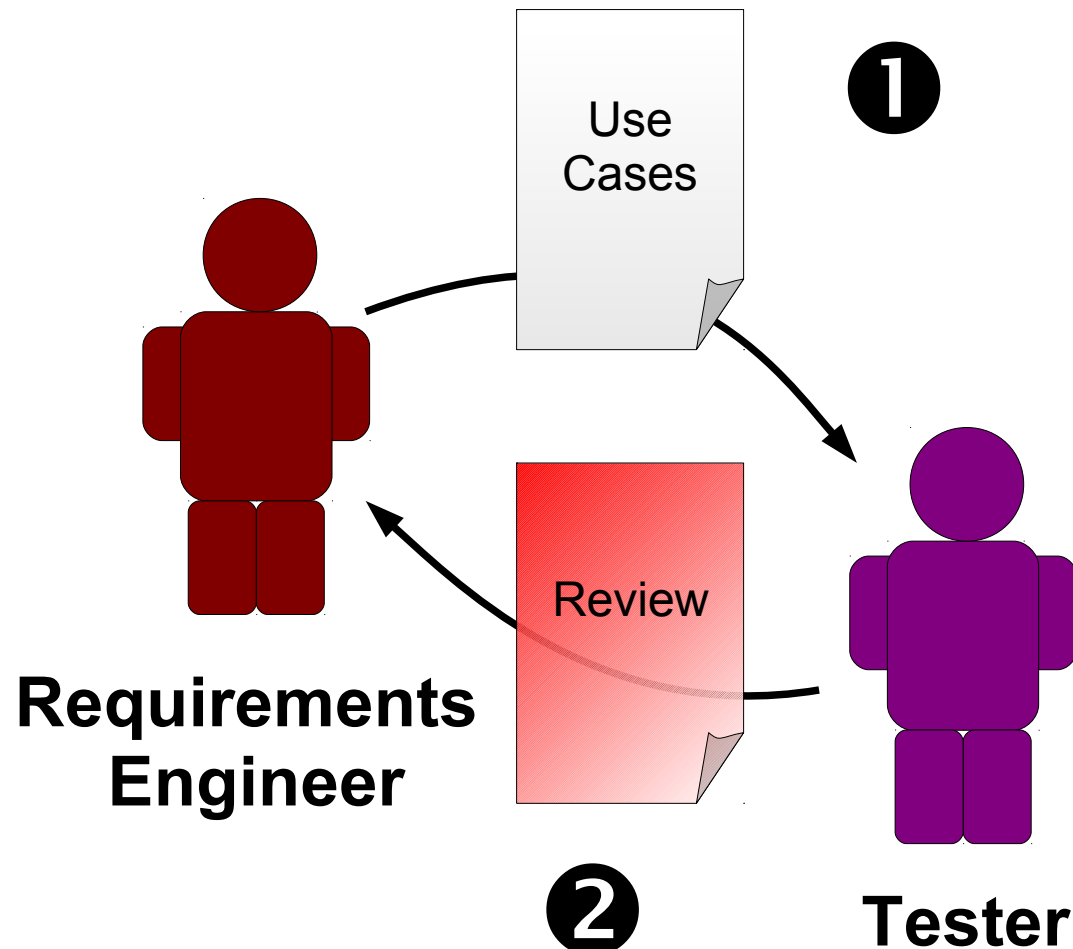
Use Case Testing (11/18)

- Effort calculation
The more complex and the more effort in creating Use Cases, the higher the test effort, e.g.
 - Very complex Use Cases (> 1 week effort)
→ 12 Test Cases
 - Medium complex Use Cases (> 1 day, ≤ 1 week effort)
→ 8 Test Cases
 - Low complex Use Cases (≤ 1 day effort)
→ 4 Test Cases
 - good basic to calculate effort in designing test cases

Black-box Techniques

Use Case Testing (12/18)

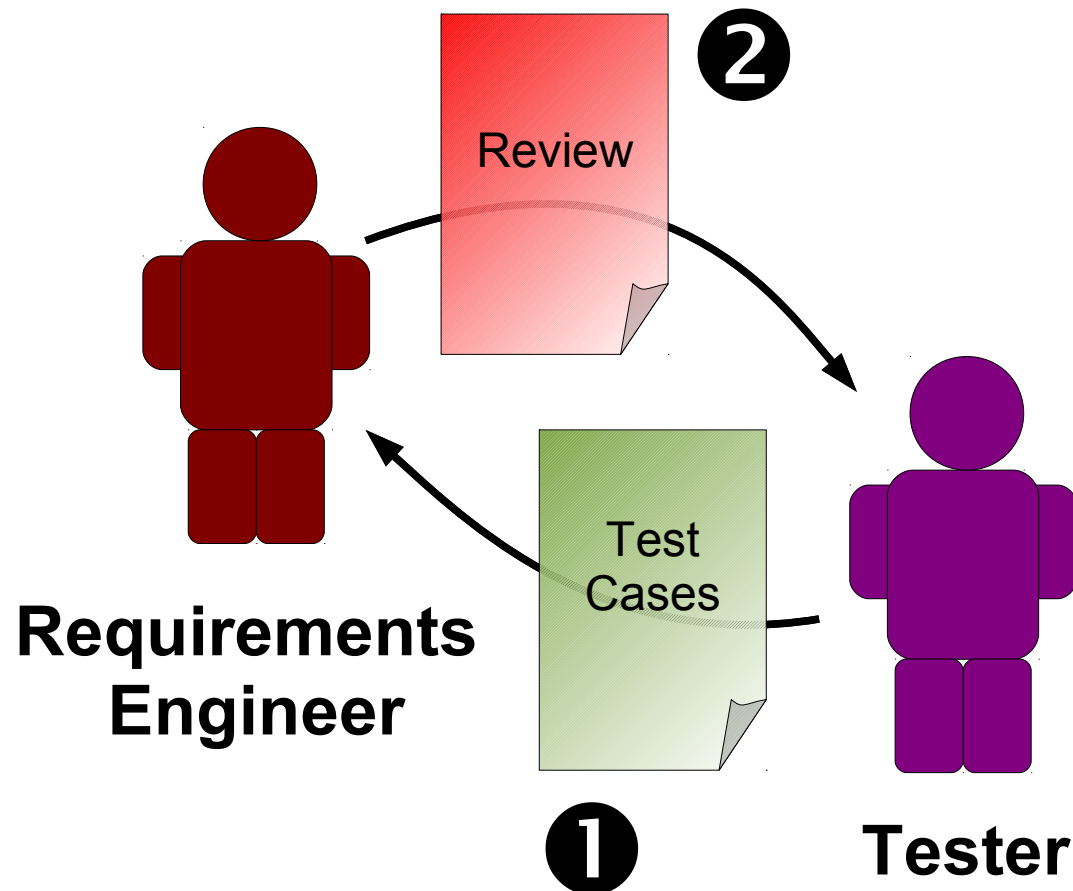
- Reviews of Use Cases as basic



Black-box Techniques

Use Case Testing (13/18)

- Reviews of Test Cases to achieve better quality





Black-box Techniques

Use Case Testing (14/18)

- Designing test cases from use cases may be combined with other specification-based test techniques.



Black-box Techniques

Use Case Testing (15/18)

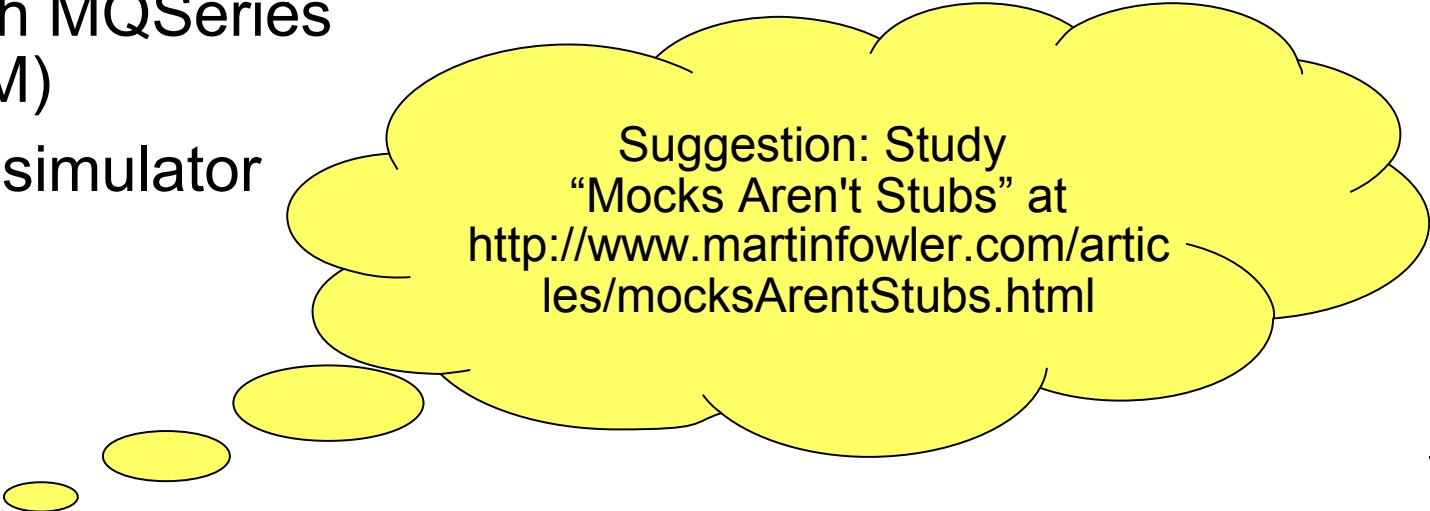
- Test Scenario
 - Synonym: Test Case Chain, Test Suite [Bla04]
 - Collection of logically related test cases [Bla04]
 - Test Scenarios are used to test processes – were process requirements implemented completely and correct?
 - A Test Scenario is a combination of – possibly modified (as a rule simplified) – Test Cases
 - A Test Scenario arises typically from a Business Scenario (Business Use Case)



Black-box Techniques

Use Case Testing (16/18)

- Test Scenario
 - Test Scenarios typically test the data flow in the system
 - Tests usually don't end with testing the system itself only
 - Interface test (e.g. with MQSeries from IBM)
 - System simulator



Suggestion: Study
“Mocks Aren't Stubs” at
<http://www.martinfowler.com/articles/mocksArentStubs.html>



Black-box Techniques

Use Case Testing (17/18)

- Test Scenario Example “User logs in a vocabulary training system and does 1st lecture”
 - Test case 17 „First login“
 - Test case 33 „Choose Language“
 - Source language „English“, Target language „Thai“
 - Level „Starter“
 - Lesson „Vacancy“
 - Learning strategy 1
 - Test case 46 „First lesson“
 - Test case 103 „Follow-up lesson“
 - Test case 132 „Score“
 - Choose Bar Chart



Black-box Techniques

Use Case Testing (18/18)

- Test data
 - All data needed for testing
 - Discussion
 - Based on Business Object Data Model or Physical Data Model
 - Artificial data or based on real business data, e. g. out of legacy systems
 - Which test data are included with delivery?
 - Feed of Test data
 - Remove of Test data („nacked system“)



Comparison (1/2)

- White Box Testing
 - Based on internal structure, code, database
 - Typically done by developer, designer
 - Test Levels: Component, Component Integration
- Black Box Testing
 - Based on Requirements, Functionality
 - Typically done by (professional) testers, users
 - Test Levels: System, System Integration, Acceptance



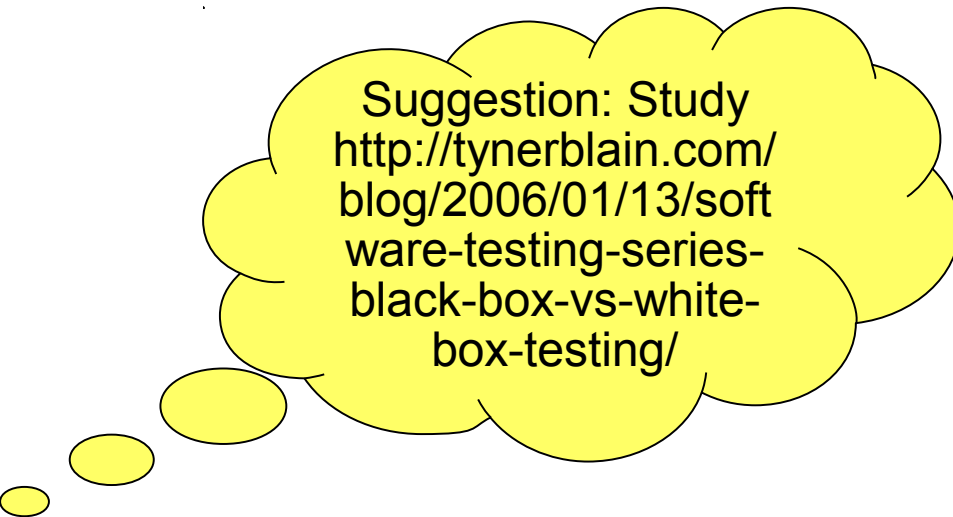
Comparison (2/2)

- White Box Testing

- Less organizational effort
- Automation easy
- Higher code quality

- Black Box Testing

- Good Testing of the complete software
- Review of specification
- Independent from implementation
- Test focus only on specification
Less quality of specification
→ less quality of test results



Suggestion: Study
<http://tynerblain.com/blog/2006/01/13/software-testing-series-black-box-vs-white-box-testing/>



Sources

- International Software Testing Qualifications Board: Certified Tester Foundation Level Syllabus, Released Version 2011, <http://istqb.org/display/ISTQB/Foundation+Level+Documents>
- [Dus07] , Dr. K. Dussa-Zieger: Testen von Software-Systemen FAU Erlangen-Nürnberg, SS 2007