

# Software Test

## Lesson 11 Test Preparation v1.3

Uwe Gühl, Jittat Fakcharoenphol



Fall 2007/ 2008



# Contents

- Test Preparation

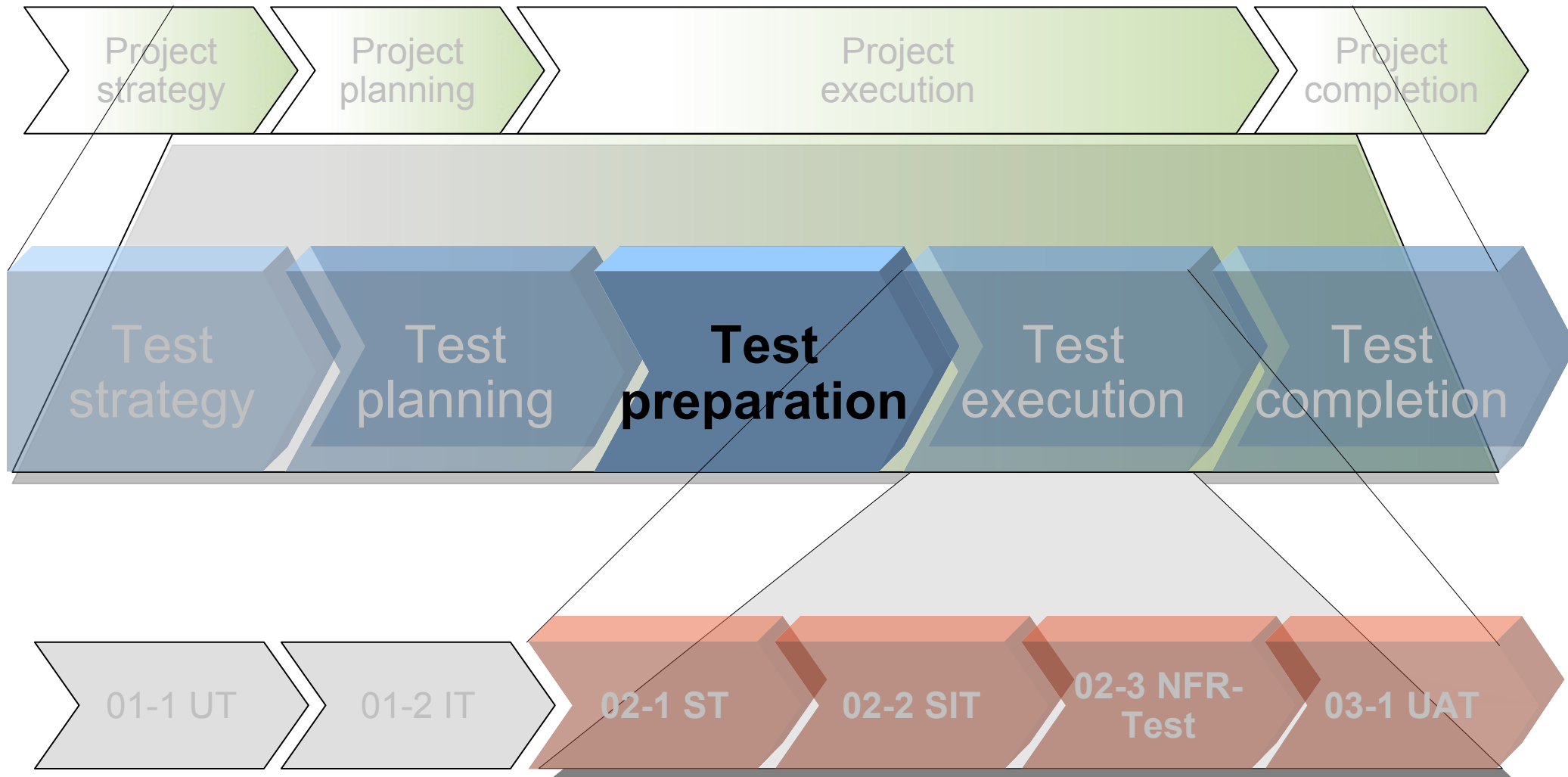
- Goal
- Involved people
- Overview
- Specification
- Test Cases
- Test Case Design
- Test Scenarios
- Test Data
- Quality
- Test environment

Scope



# A sample testing cycle

## Test Plan





# Test Preparation Goal

- Doing all preparations, so that the software test could be executed optimal
- Test environment is ready
- Resources, rooms, and tester are available
- Release plan done
- Test Cases, Test Scenarios, and Test Data are complete
- Test processes are defined
- Testers are trained



# Test Preparation Goal

- Constraint – Keep in mind
  - You can't prepare everything, there will be always a gap, to consider
    - Specification
      - Typical 80 % of all the code affects error handling, but not so much is described in the specification [p. 165 KBP02]
      - It's difficult to assume where problem areas will arise during testing – hence you need flexibility to build e. g. a task force and to change the test execution plan
    - Program
      - Typically it's simple not possible to test a program completely
      - Example: [KFN99] describes that a simple program of about 20 lines from Myers in 1979 had 100 trillion paths – meaning a fast tester needs a billion years to test



# Test Preparation People

- Test Manager
- Tester (Test Engineers, Test Designer)
- Test Data Manager
- Environment Manager
- Collaboration with
  - Specification creators
  - Software developer
  - Operation



# Test Preparation People

- Organization – Hint
  - Try to define “pairs” concerning the subjects of the specification / project, e.g.
    - Subject 1 – (Specification creator1 / Tester1)
    - Subject 2 – (Specification creator2 / Tester2)
    - ...
  - Extend the “pairs” to “triple” concerning the implementation
    - Subject 1 –  
(Specification creator1 / Tester1 / Software developer1)
    - ...



# Test Preparation Overview

- Which tests to prepare and what you need (extract)

- Functional System Test
  - Test Cases and Test Data
- Integration Test
  - Test Scenarios and Test Data
- Smoke Test

**Focus**

- Basic Test Cases, Test automation tools
- Regression Test (to verify bug fixes, older bug fixes and stability concerning side effects)
  - (Basic) Test Cases, Test automation tools





# Test Preparation Overview

- Which tests to prepare and what you need
  - Installation Test
    - Release Management, Environment, Operation, Installation Manual
  - Load Testing
    - NFR Test Cases, NFR Testing Tools
  - Performance Testing
    - NFR Test Cases, NFR Testing Tools, specific environment
  - Security Testing
    - NFR Test Cases, special security know-how



# Test Preparation Specification

- The specification is basic for testing  
Work on it and use it!
  - If informations are missing they have to be provided during Test Preparation  
→ Additional value for the project
  - Typical is the detection of ambiguities, inconsistencies or open issues in the specification  
→ Change management has to be established in the project
  - Use Cases as requirements are the basis for Test Cases – for every Use Case at least one Test Case should be there



# Test Preparation Specification

- If there is no specification? [KBP02]
  - Plenty of other sources can help you
  - Examples
    - User manual draft and previous version's manual
    - Interviews with project manager, customer, developers, operation, subject matter experts, service / technical support
    - Marketing presentations and all other documents concerning the product
    - Test suites of compatible third-party products
    - Related published regulations and standards
    - Technical: Source code, database table definitions



# Test Preparation Specification

- More – implicit specification [KBP02]
  - useful source of requirements information not acknowledged as authoritative by the clients
  - Examples
    - Competing or related products
    - Legacy systems or older versions of the same product
    - Related sources like books, magazine articles
    - Comments by customers
    - GUI style guides
    - Your experience



# Test Preparation Specification

- Proceeding
  - Specification creator should present the contents in a training session for the Testers
  - Communication between specification creator and tester should be established
  - Consider the structure of the specification for structuring the Test Cases  
Example: If the specification has 7 subjects you should think about a “subject” keyword in your Test Cases referring to these 7 subjects
  - Concerning coverage: Each part of the specification should be covered with Test Cases



# Test Preparation

## Test Cases

- Test Cases – How to?  
[KBP02] describe a Five-fold Testing system,  
any testing can be described in terms

**1** – Tester: Who does the testing?

**2** – Coverage: What gets tested?

**3** – Potential problems: Why are you testing?  
What are potential risks you are testing for?

**4** – Activities: How do you test?

**5** – Evaluation: How to tell if a test passed or failed?  
How do you know if you've found a bug?



# Test Preparation

## Test Cases

- Test Cases – How to? **1** Tester
  - User testing: People who typically would use the system
  - Alpha testing: Testing with a test team
  - Beta testing: Testers out of the organization with a product close to completion
  - Subject matter expert testing: Experts on issues with valuable knowledge
  - Paired testing: 2 testers work together to find bugs
  - Eating one's own dog food: Company internals use prerelease versions of the product



# Test Preparation

## Test Cases

- Test Cases – How to? **2** Coverage
  - Specification based testing: Testing focused on every factual claim (yes/no) in the specification
  - Requirements-based testing: Focused on requirements in the specification
  - Equivalence class analysis: Dividing a set of values for a variable into different equivalence classes, testing only one or two members of it
  - Boundary testing: Testing smallest or largest members of an equivalence class





# Test Preparation

## Test Cases

- Test Cases – How to? **2** Coverage
  - Best representative testing: Most critical value of an equivalence class – if this works, others will as well
  - Input field test catalog: Collecting possible types for input fields – could result in a test matrix
  - State based testing: Changing a state of a program and testing again (e. g. different roles)
  - Function testing: Test every function (Unit test)
  - Function integration testing (Unit test)
  - Logic testing: Check every logical relationship



# Test Preparation


## Test Cases

- Test Cases – How to? **3** Potential problems
  - Risk based testing, two approaches
    - focus on probability to find expensive failures
    - focus on purpose of finding errors
  - Idea: Focusing on constraints, e. g. input constraints or computation constraints (like multiplying big numbers)
  - Hint: Comparable nonrisk-based testing because the worst risks are the risks we don't know



# Test Preparation


## Test Cases

- Test Cases – How to?  Activities
  - Scenario testing
    - Realistic: Based on what customers do
    - Complex: Should cover several features
    - Quick determinable if the test passed or not
  - Regression testing
    - Bug fix regression: Just to be sure
    - Old bug regression: Do they revive?
    - Side effect regression or stability regression
  - Smoke testing: Is a new build worth testing?
  - Installation testing



# Test Preparation

## Test Cases

- Test Cases – How to?  Activities
  - Guerilla testing: Fast and vicious attack on the program
  - Load testing: Many demands for resources at the same time. The system will probably fail but it helps to find vulnerabilities in the software.
  - Long sequence testing: To find wild pointers and memory leaks
  - Performance testing: How quickly runs the program? Where is optimization needed?



# Test Preparation

## Test Cases

- Test Cases – How to? **5** Evaluation
  - Self-verifying data: The data files carry information to determine if the output data are fine
  - Comparison with
    - saved results: Could be used in regression testing – just comparing the results of the prior run
    - specification
  - Oracle based testing: An evaluation tool, typically another program, tells if a program has passed or failed a test



# Test Preparation

## Test Cases

- Test Cases – How to? **5** Evaluation
  - Heuristic consistency
    - History: Present function behaves like in the past
    - Comparable products: E. g. OpenOffice behaves like Microsoft Office
    - Claims: Behavior is consistent to what people say
    - User's expectation: What we think the user wants
    - Within product: Consistent with the overall behaviour of the program (e. g. [F1] calls in every window help)



# Test Preparation

## Test Cases

- Test Cases result out of the specification
  - Use Cases
  - More requirements and their descriptions
  - GUI templates
  - „intuitive“ – from the experience and know how of the tester



# Test Preparation Test Cases

- Number of Test Cases
  - Better many simple instead of less complex - Why?
    - A Test Case should produce only few potential errors, otherwise it has to be retested too often, if too many defects are found with it.
  - Proposal: Number of Test Cases realign on number and complexity of corresponding Use Cases (measurable on duration of creation), e. g.
    - Complex Use Case: 12 Test Cases
    - Average Use Case: 8 Test Cases
    - Simple Use Case: 4 Test Cases





# Test Preparation

## Test Cases

- Number of Test Cases
  - Positive Test Cases → Successful execution
  - Negative Test Cases → Error case
  - Every condition (if / else or case statement) results in corresponding Test Cases
- Hint: It's possible, that out of multiple Use Cases one Test Case arises (intelligent Test Design)



# Test Preparation Test Cases

- Prioritization
  - Typical classification: 1 (high), 2 (medium), 3 (low)
  - Criteria
    - Following Use Cases: High prioritized Use Cases result in higher prioritized Test Cases
    - At least one Test Case of every Use Case should have highest priority (coverage)
    - Complexity of basic Use Case: The more complex, the more important are the Test Cases
    - Positive Test Cases are more important than negative Test Cases



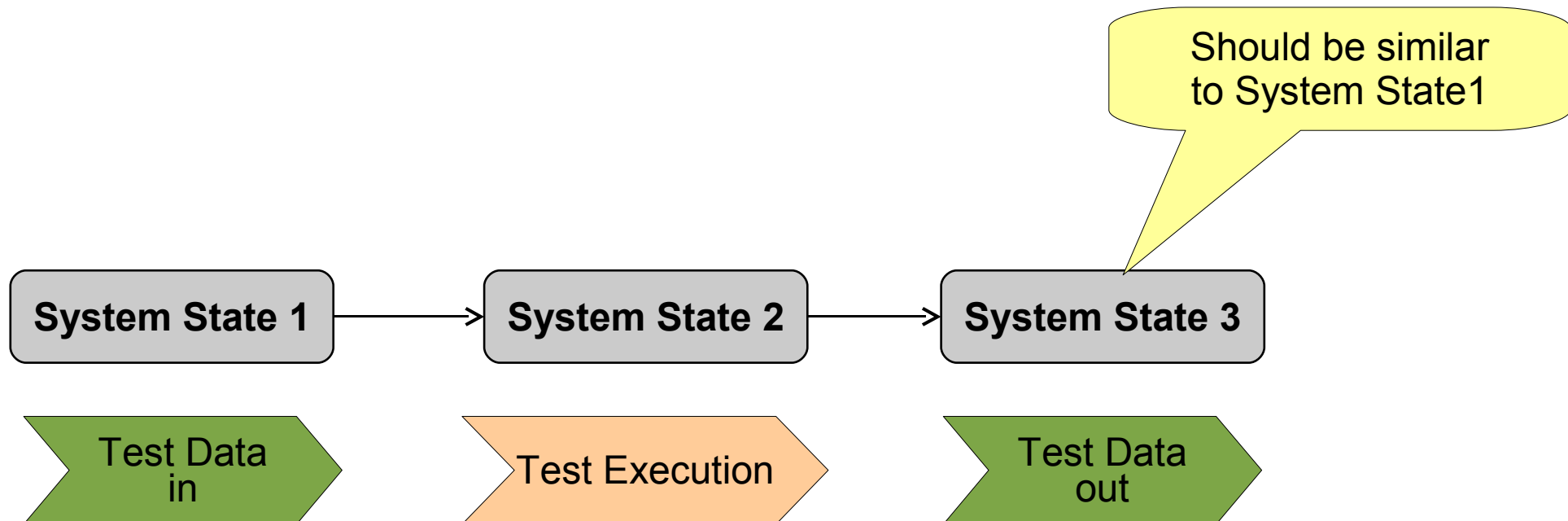
# Test Preparation Test Cases

- Contents
  - Roles
  - Input criteria:
    - Which Test Data (for attributes) are needed?
    - Which dependencies to other Test Cases exist?
  - Test Steps
  - Expected result
  - Output criteria
    - Which data were generated (for attributes)



# Test Preparation Test Cases

- Autarkic Test Cases:
  - Optimal are independent Test Cases
  - After execution of a Test Case the system should be in the same status as before





# Test Preparation Test Cases

- Autarkic Test Cases:
  - As autarkic Test Cases are difficult to reach in praxis, following alternatives are possible
    - Defining different status of test data feed into the database of the system, e. g.
      - TestDataFeed0 – empty data base
      - TestDataFeed1 – data base with a specified set of objects
    - On defined time stamps the data base gets “cleaned” and all data generated so far get completely deleted
    - A feed establishes the system with the specified data set



# Test Preparation Test Cases

- Test Management Tool
  - Where / How to store all the Test Cases?
  - Typically you use a Test Management Tool
  - Alternative: Spreadsheets
  - Tasks concerning the tool:
    - Installation, licenses, enabling access, administration
    - Configuration of tool (reporting, process specific)
    - Training, examples
    - Guidelines, manual



# Test Preparation Test Cases

- Proceeding
  - Plan:
    - How many Test Cases would we like to generate?
    - Validation on milestones (weekly)
  - Status of Test Cases
    - In work
    - Ready for review
    - Review done
    - Ready for test execution



# Test Preparation

## Test Case Design

- Example – Use Case
  - A shop calculates shipping and handling for all orders with an order value less than ฿ 5,000
  - Shipping and handling is about ฿ 70
  - For orders higher or equal to ฿ 5,000 no shipping and handling is calculated

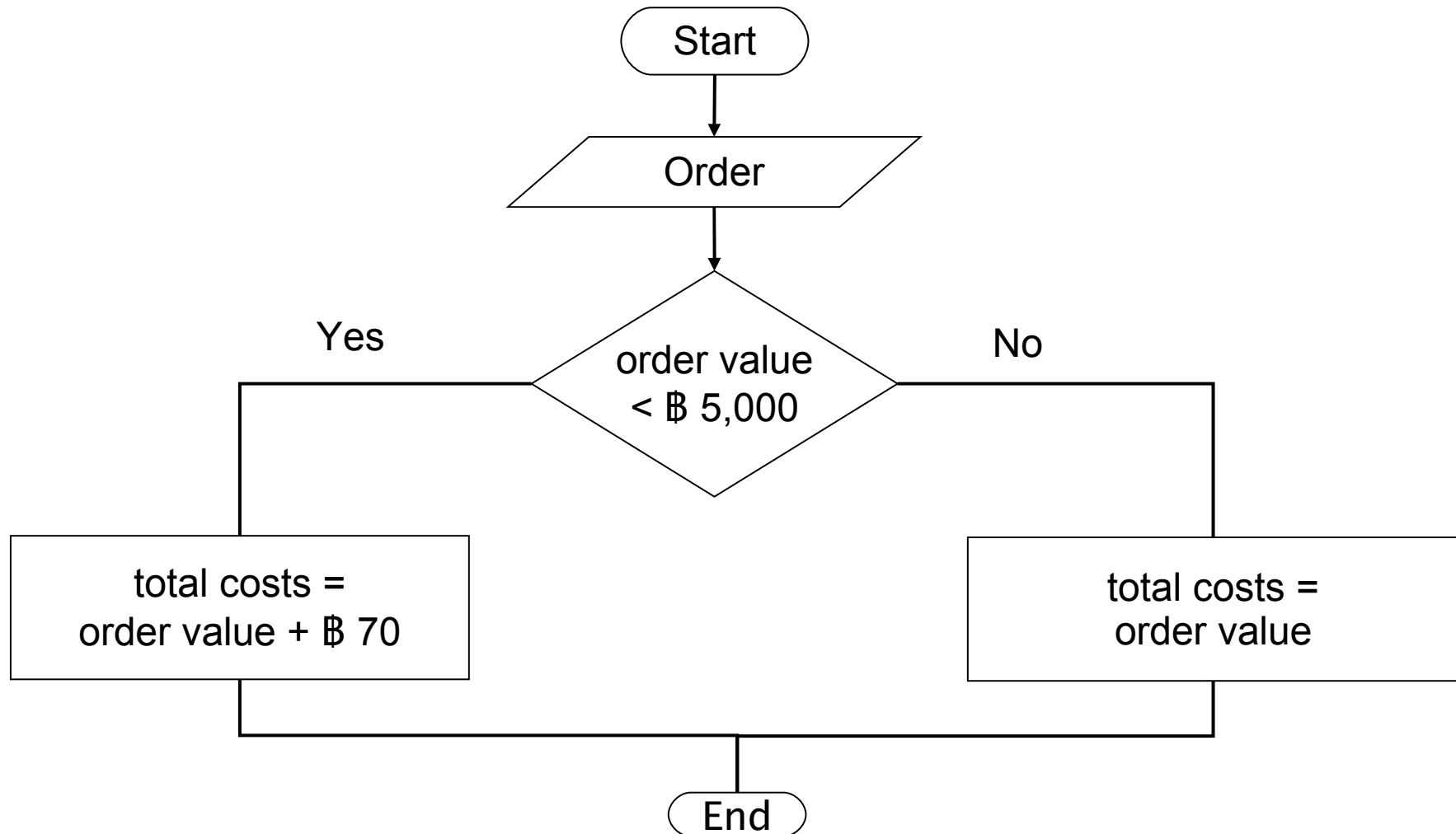




# Test Preparation

## Test Case Design

- Example – Activity Diagram in a Use Case





# Test Preparation

## Test Case Design

- Which Test Cases to create?
- Possible input values (Test Data)?

฿ 3,000

฿ 0

฿ 10,000

฿ 99,999,999

฿ 5,000

– ฿ 100

฿ 4,999

Ads f!!\$%aa

฿ 5,001

\$ 100



# Test Preparation

## Test Case Design

- Proceeding for Test Case creation
- Which methods did we use?

฿ 3,000	Equivalence class analysis
฿ 10,000	
฿ 5,000	Boundary testing
฿ 4,999	
฿ 5,001	

฿ 0	Boundary testing
฿ 99,999,999	
– ฿ 100	Type error tolerance
Ads f!!\$%aa	
\$ 100	

- Do we have Test Cases now?



# Test Preparation

## Test Case Design

- Which Test Cases to create?
- Possible input values (Test Data) – more ideas
  - nothing
  - empty space
  - Nonprinting characters
  - Language reserved characters
  - ASCII 255 (End of file)
  - Modifiers ([Strg] [Ctrl])
  - special signs
  - Function keys ([F2])
  - Enter digits, wait, enter more
  - Enter digits, during operation more



# Test Preparation

## Test Case Design

- Combination Testing [KBP02]
  - Basic Challenge:  
Impossible Testing of all combinations – Example
    - 3 variables with 100 possible value means  
 $(100 \times 100 \times 100) = 1,000,000$  Test Cases



# Test Preparation

## Test Case Design

- Combination Testing

$3 \times 3 \times 4 \times 2 \times 5 \times 5$   
= 1,800 possibilities  
**How to test?**

- Example – What to do? [Rod07]

- 3 policy types (TP (Third Party), TPFT (Third Party Fire & Theft, FC (fully comprehensive insurance))
    - 3 storage modes (garaged, drive, road)
    - 4 No-claims discount (NCD) (0, 1, 2 and 3+ years)
    - 2 license types (full and provisional)
    - 5 age categories (17-21, 22-30, 31-40, 41-50 and 51+)
    - 5 engine sizes (<1000 cc, <1600 cc, <2000 cc, <2999 cc and 3000+ cc)

- Unit:

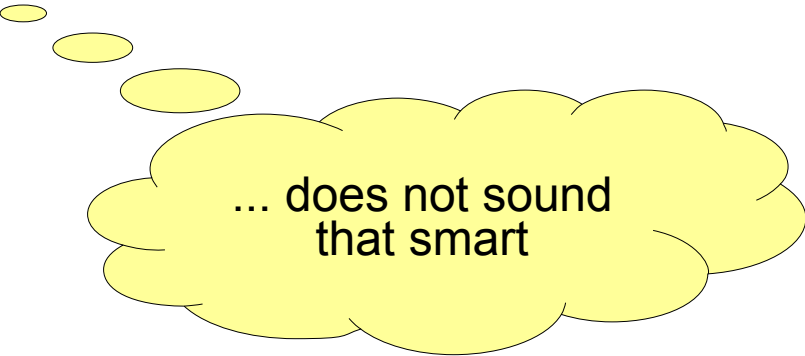
- FC, garaged, 3+ NCD, provisional, 51+, <1600 cc



# Test Preparation

## Test Case Design

- Combination Testing [Rod07]
  - What to do?
    - Test all combinations
    - Don't test at all! (too many)
    - Choose one or two and hope for the best
    - Choose the tests you like or that are easy
    - Choose the first few
    - Randomly select a subset



... does not sound  
that smart



# Test Preparation

## Test Case Design

- Combination Testing [KBP02]
  - What to do? – General idea: Domain partitioning
    - Reduce the number of values to be tested by partitioning in subdomains
    - Example: Assume 3 variables with 100 possible value could be reduced to 5 subdomains
    - Instead of  $(100 \times 100 \times 100) = 1,000,000$  Test Cases “only”  $(5 \times 5 \times 5) = 125$  Test Cases are necessary
    - Good choice of subdomains is necessary





# Test Preparation

## Test Case Design

- Combination Testing [KBP02] [Rod07]
  - What to do? – Idea: Testing subsets
    - 1) Representative testing  
Testing of defined “representatives”
    - 2) Achieving all singles  
Testing of each value of variables at least once
    - 3) Achieving all pairs  
Testing of each value of variable at least in combination with all values of each other variable.  
Two possibilities to achieve
      - a) Orthogonal Arrays
      - b) All-pairs algorithms



# Test Preparation

## Test Case Design

- Combination Testing [KBP02]
  - Working example
    - Variable V1 with possible values
      - A (Windows XP)
      - B (Windows Vista)
      - C (Linux)
      - D (FreeBSD)
      - E (Macintosh OS X)
    - Variable V2 with possible values
      - I (Mozilla Firefox)
      - J (Safari)
      - K (Internet Explorer 6)
      - L (Internet Explorer 7)
      - M (Opera)

Testing of all combinations would result in  
 $(5 \times 5 \times 5) = 125$  tests

### Variable V3 with possible values

- V (Disk option 1)
- W (Disk option 2)
- X (Disk option 3)
- Y (Disk option 4)
- Z (Disk option 5)



# Test Preparation

## Test Case Design

- Combination Testing

- 1) Representative testing

- Repr. 1 to 5: (A, K, V) to (A, K, Z)
    - Repr. 6 to 10: (A, L, V) to (A, L, Z)
    - Repr. 11 to 15: (A, I, V) to (A, I, Z)
    - Repr. 16 to 20: (B, L, V) to (B, L, Z)
    - Repr. 21 to 25: (B, I, V) to (B, I, Z)
    - Repr. 26 to 30: (C, I, V) to (C, I, Z)
    - ...

“Standard old IE”

“Standard new IE”

“Standard Mozilla”

“Modern new IE”

“Modern Mozilla”

“Linux Mozilla”



# Test Preparation

## Test Case Design

- Combination Testing

- 1) Representative testing

- The “representatives” should stand for typical configurations
    - Less than all possible combinations
    - Most probable combinations could be tested first
    - No consideration of less probable or improbable combinations (like Explorer with Linux)



# Test Preparation

## Test Case Design

- Combination Testing [KBP02]

### 2) Achieving all singles

- Goal: Basic combination of all values
- Every value is covered at least once
- Only 5 tests necessary
- If important tests are missing (compare to “representatives”), add them

Test Case	Variable 1		Variable 2		Variable 3	
1	A	Windows XP	I	Mozilla Firefox	V	Disk option 1
2	B	Windows Vista	J	Safari	W	Disk option 2
3	C	Linux	K	Internet Explorer 6	X	Disk option 3
4	D	FreeBSD	L	Internet Explorer 7	Y	Disk option 4
5	E	Macintosh OS X	M	Opera	Z	Disk option 5



# Test Preparation

## Test Case Design

- Combination Testing

### 3) Achieving all pairs – Statistics [Rod07]

- Running 1% to 20% of all possible tests you will find 70% to 85% of the total bugs
- Cohen reported that Test Cases created by the “allpairs” algorithm provided better code coverage than random tests

- 300 random tests:
  - 67% statement coverage\*
  - 58% decision coverage\*\*
- 200 “all-pairs” tests:
  - 92% statement coverage
  - 85% decision coverage

\* Has each line of the source code been executed?

\*\* Has each evaluation point (such as a true/false decision) been executed?



# Test Preparation

## Test Case Design

- Combination Testing

### 3) Achieving all pairs – Hints [KBP02] [Rod07]

- Don't focus on all pairs only
  - It's too risky only to use all-pairs cases
  - Specific combinations should be added, if they are critical
- No priority is given to the test cases
- Some combinations produced may be infeasible
- Always start in the first column with the variable with the highest number of values, the second column contents the variable with the second highest number and so on



# Test Preparation

## Test Case Design

- Combination Testing

### 3) Achieving all pairs

- Tools [Rod07]
  - “allpairs” by James Bach [Bac07]
  - “pict” by Jacek Czerwonker [Cze08a]
  - Classification-Tree Editor CTE/XL at [PW07]
  - Taguchi Orthogonal Array Selector [Tag08]
- Read on
  - Study for tools, papers and more concerning pairwise technique at [Cze08]





# Test Preparation

## Test Case Design

- Combination Testing

### 3a) Achieving all pairs – Orthogonal Arrays [Rod07]

- have unique properties that allow them to be applied to a systematic way of testing.
- are a series of two-dimensional arrays based upon the following notation:

$L_x(n^y)$  with

- $x$  = number of rows
- $y$  = number of columns
- $n$  = maximum number choices within each category/variable



# Test Preparation

## Test Case Design

- Combination Testing

### 3a) Achieving all pairs – Orthogonal Arrays [Rod07]

- Introducing example  
Trying numbers 1 and 2 results in  
(1, 1), (1, 2), (2, 1), and (2, 2)
- The table shows the  $L_4(2^3)$  orthogonal array

	1	2	3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

- If we pick any two columns, we can see that every pair combination has been covered



# Test Preparation

## Test Case Design

- Combination Testing [KBP02]

3a) Achieving all pairs – Orthogonal Arrays

- 25 Test Cases as result

Test Case	Variable 1	Variable 2	Variable 3
1	A Windows XP	I Mozilla Firefox	V Disk option 1
2	A Windows XP	J Safari	W Disk option 2
3	A Windows XP	K Internet Explorer 6	X Disk option 3
4	A Windows XP	L Internet Explorer 7	Y Disk option 4
5	A Windows XP	M Opera	Z Disk option 5
6	B Windows Vista	I Mozilla Firefox	W Disk option 2
7	B Windows Vista	J Safari	X Disk option 3
8	B Windows Vista	K Internet Explorer 6	Y Disk option 4
9	B Windows Vista	L Internet Explorer 7	Z Disk option 5
10	B Windows Vista	M Opera	V Disk option 1
11	C Linux	I Mozilla Firefox	X Disk option 3
12	C Linux	J Safari	Y Disk option 4
13	C Linux	K Internet Explorer 6	Z Disk option 5
14	C Linux	L Internet Explorer 7	V Disk option 1
15	C Linux	M Opera	W Disk option 2
16	D FreeBSD	I Mozilla Firefox	Y Disk option 4
17	D FreeBSD	J Safari	Z Disk option 5
18	D FreeBSD	K Internet Explorer 6	V Disk option 1
19	D FreeBSD	L Internet Explorer 7	W Disk option 2
20	D FreeBSD	M Opera	X Disk option 3
21	E Macintosh OS X	I Mozilla Firefox	Z Disk option 5
22	E Macintosh OS X	J Safari	V Disk option 1
23	E Macintosh OS X	K Internet Explorer 6	W Disk option 2
24	E Macintosh OS X	L Internet Explorer 7	X Disk option 3
25	E Macintosh OS X	M Opera	Y Disk option 4



# Test Preparation

## Test Case Design

- Combination Testing [KBP02]

### 3b) Achieving all pairs – All-pairs algorithms

- Goal: Combination of all of the pairs of values of every variable
- Result are much more values than with “all singles” approach, but less than all combinations

“choose a specially selected, fairly small subset that find a large number of defects – more than you would normally find with a normal subset”  
(Lloyd Roden, inspired by Lee Copeland)



# Test Preparation

## Test Case Design

- Combination Testing [KBP02]

### 3b) Achieving all pairs – All-pairs algorithms

- 25 Test Cases as result

Test Case	Variable 1	Variable 2	Variable 3
1	A Windows XP	I Mozilla Firefox	V Disk option 1
2	A Windows XP	J Safari	W Disk option 2
3	A Windows XP	K Internet Explorer 6	X Disk option 3
4	A Windows XP	L Internet Explorer 7	Y Disk option 4
5	A Windows XP	M Opera	Z Disk option 5
6	B Windows Vista	I Mozilla Firefox	W Disk option 2
7	B Windows Vista	J Safari	Z Disk option 5
8	B Windows Vista	K Internet Explorer 6	Y Disk option 4
9	B Windows Vista	L Internet Explorer 7	V Disk option 1
10	B Windows Vista	M Opera	X Disk option 3
11	C Linux	I Mozilla Firefox	X Disk option 3
12	C Linux	J Safari	Y Disk option 4
13	C Linux	K Internet Explorer 6	Z Disk option 5
14	C Linux	L Internet Explorer 7	W Disk option 2
15	C Linux	M Opera	V Disk option 1
16	D FreeBSD	I Mozilla Firefox	Y Disk option 4
17	D FreeBSD	J Safari	X Disk option 3
18	D FreeBSD	K Internet Explorer 6	V Disk option 1
19	D FreeBSD	L Internet Explorer 7	Z Disk option 5
20	D FreeBSD	M Opera	W Disk option 2
21	E Macintosh OS X	I Mozilla Firefox	Z Disk option 5
22	E Macintosh OS X	J Safari	V Disk option 1
23	E Macintosh OS X	K Internet Explorer 6	W Disk option 2
24	E Macintosh OS X	L Internet Explorer 7	X Disk option 3
25	E Macintosh OS X	M Opera	Y Disk option 4



# Test Preparation

## Test Scenarios

- Test Scenarios are used to test the functionality of the processes
- Test Scenarios are generated out of several Test Cases – Number and depth of Test Steps could differ



# Test Preparation

## Test Scenarios

- Proceeding
  - „Top Down“ approach (parallel to Test Case creation)
    - Study of Business Scenarios (Business processes)
    - Out of a Business Scenario arise a specific number of Test Scenarios (if-else, case junctions)
    - Definition of number and rough contents of the Test Scenarios
    - Detailing up to the Test Cases



# Test Preparation Test Scenarios

- Proceeding
  - „Bottom Up“ approach (after Test Case creation)
    - Study of Test Cases
    - Combination of Test Cases (Test Case Chains), until complete scenarios result
    - Verification / Completion with Business Scenarios





# Test Preparation Test Data

- Depending on the topic this could be a big challenge in the test project
- Basic strategy; decision:
  - Based on the
    - Business Object Data Model (BuOM) out of the specification or
    - Physical Data Model (PhDM) out of the implementation
  - Artificial generated or out of business data, e. g. out of legacy systems
  - For the Test execution Test Data are required that are as realistic as possible



# Test Preparation

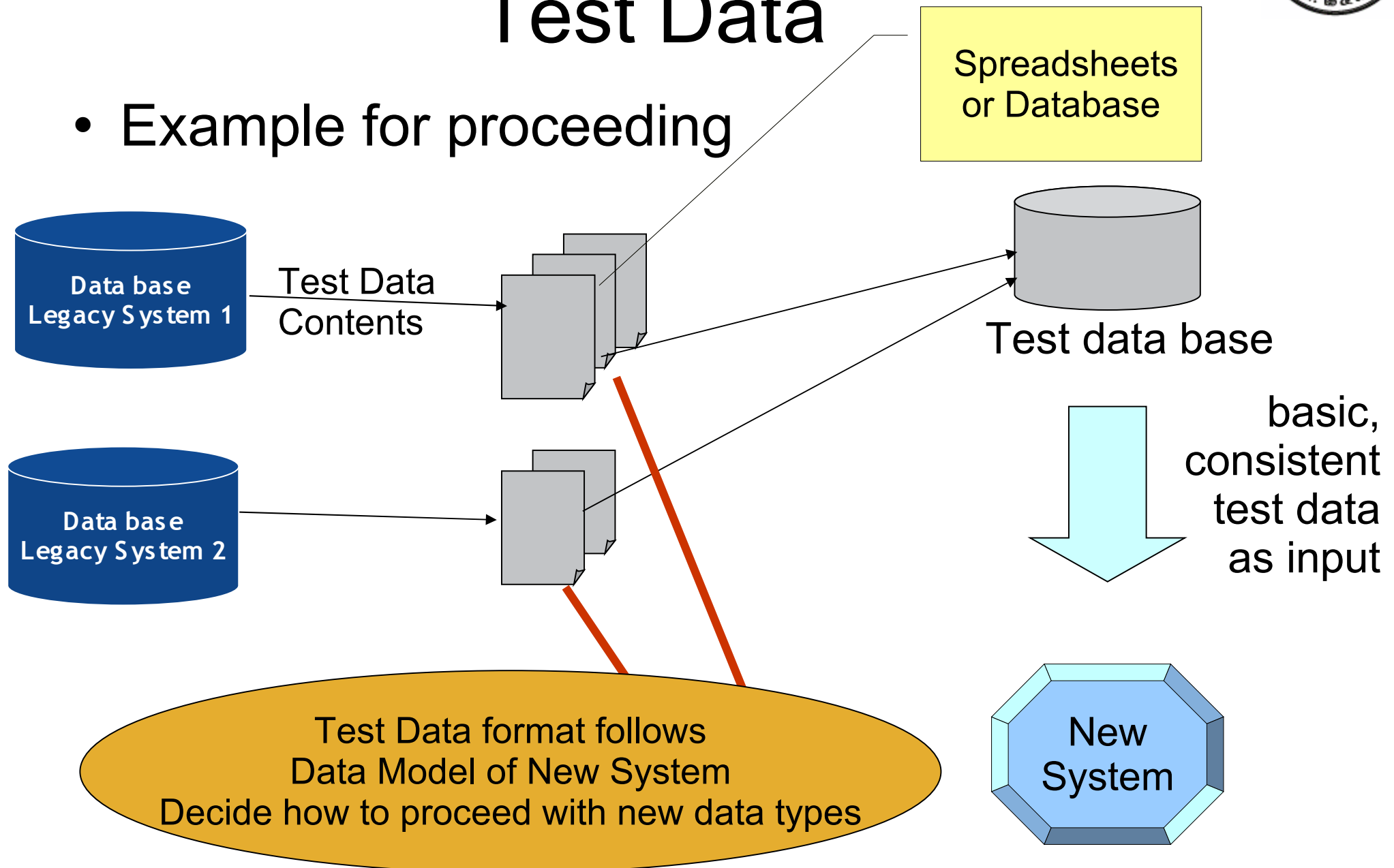
## Test Data

- Assignment Test Data to Test Cases / Test Scenarios
- Feed:
  - Delivers the software vendor software with a basic (Test) data set?
  - Is the input of Test Data only possible via the GUI or is a feed possible – with scripts and API (interface)?
    - important for tests simulating the software use with already existing data sets
    - important for mass data, e. g. for Performance tests



# Test Preparation Test Data

- Example for proceeding





# Test Preparation Quality

- Generated Test Cases, Test Scenarios, and Test Data could contain faults or could be wrong!
- Possibilities to raise the quality
  - Use of check lists
  - Common agreements (e. g. naming convention, granularity, description of attributes)
  - Documentation of a quality handbook with description of proceeding



# Test Preparation Quality

- Possibilities to raise the quality
  - Communication of the tester with each other – presenting each other current status of work, working in pairs
  - Review / Rework / Lessons learned
    - Review with all project stakeholders (business area!)
    - Planning and executing overwork of Test Cases, Test Scenarios, and Test Data



# Test Preparation Environment

- Definition of requirements
- Coordination with operator / responsible person of test laboratory
- Ensuring operation
- Connecting of necessary additional systems (Test versions, simulators)
- If needed parallel environments (functional, NFR)



# Sources

- [Bac07] James Bach: Test Tools, Website <http://www.satisfice.com/tools.shtml>, 2007
- [Cze08] Jacek Czerwonka: Pairwise Testing, Website [www.pairwise.org](http://www.pairwise.org), 2008
- [Cze08a] Jacek Czerwonka: Pairwise Testing – Available Tools, Website <http://www.pairwise.org/tools.asp>, 2008
- [KBP02] Cem Kaner, James Bach, Bret Pettichord: Lessons Learned in Software Testing, Wiley Computer Publishing, 2002
- [KFN99] Cem Kaner, Jack Falk, Hung Quoc Nguyen: Testing Computer Software, Wiley Computer Publishing, 1999
- [Pet08] Bret Pettichord: Five Ways to Think about Black Box Testing, Website [http://www.stickyminds.com/pop\\_print.asp?ObjectId=2968&ObjectType=COL](http://www.stickyminds.com/pop_print.asp?ObjectId=2968&ObjectType=COL), 2008
- [PW07] Roman Pitschinetz, Joachim Wegener: Systematic Testing, Website <http://www.systematic-testing.com>, 2007
- [Rod07] Lloyd Roden: Pairwise Testing, Thailand SPIN, November 2007
- [Tag08] Genichi Taguchi: Taguchi Orthogonal Array Selector, <http://www.freequality.org/Default.aspx?page=60>, 2008