MTAT.03.159: Software Testing

Lecture 03:  
Black-Box Testing (advanced) Part 2

Spring 2018

Dietmar Pfahl
email: dietmar.pfahl@ut.ee
Black-Box Testing Techniques

- Equivalence class partitioning (ECP)
- Boundary value analysis (BVA)
- Cause-effect graphing
- Combinatorial testing
- State transition testing (State-based testing)
- Exploratory testing
- Usability testing
- A/B testing (UX)
Black-Box Testing Techniques

- Equivalence class partitioning (ECP)
- Boundary value analysis (BVA)
- Cause-effect graphing
- Combinatorial testing
- **State transition testing** (State-based testing)
- Exploratory testing
- Usability testing
- A/B testing (UX)
State-Transition Testing - Example

Use Case Diagram

- Check Account
- Withdraw Money
- ...

ATM
State-Transition Testing - Example

Use Case Description: Check Account

Role: Customer
Goal: Customer wants to check the amount of money in his/her accounts

Scenario (actions):
1. ATM asks for customer card
2. Customer enters card
3. ATM asks for PIN code
4. Customer enters PIN code
5. …
State-Transition Diagram

State-Transition Graph

Card inserted
Ask for PIN

Invalid PIN
Beep

Wait for card
Cancel
Return card

Wait for PIN
Valid PIN
Ask amount

Now create a set of test cases that triggers each state-transition at least once
State Table

<table>
<thead>
<tr>
<th>Input (Event)</th>
<th>State</th>
<th>Wait for Card (S1)</th>
<th>Wait for PIN (S2)</th>
<th>Next (S3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card inserted</td>
<td>Ask for PIN -&gt; S2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Invalid PIN</td>
<td>-</td>
<td>Beep -&gt; S2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Valid PIN</td>
<td>-</td>
<td>Ask amount -&gt; S3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Cancel</td>
<td>-</td>
<td>Return card -&gt; S1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
# State Table

<table>
<thead>
<tr>
<th>Input (Event)</th>
<th>State</th>
<th>Wait for Card (S1)</th>
<th>Wait for PIN (S2)</th>
<th>Next (S3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card inserted</td>
<td>Ask for PIN -&gt; S2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Invalid PIN</td>
<td>-</td>
<td>Beep -&gt; S2</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Valid PIN</td>
<td>-</td>
<td>Ask amount -&gt; S3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Cancel</td>
<td>-</td>
<td>Return card -&gt; S1</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
| ...           | ... | ... | ... | ...

Test (=sequence of test cases):
S1 -> 'Card inserted' / 'Ask for PIN' -> S2 -> 'Cancel' / 'Return card' -> S1

4 Test Cases:
S1: 'Card inserted' / 'Ask for PIN' -> S2
S2: 'Invalid PIN' / 'Beep' -> S2
S2: 'Valid PIN' / 'Ask amount' -> S3
S2: 'Cancel' / 'Return card' -> S1
State-Transition Testing: Example 2

Extract of a Specification Doc:

Parameters
- PORT_A: calling phone
- PORT_B: called phone

PORT_A identifies the connection from which a call is to be set up. The actual state of the call setup is globally available. Depending on this a new state arises after the evaluation of the transferred event. The delivered state is DISCONNECTED, if the call setup was terminated, it is DIALING, if the call setup is in progress but not completed yet. It is CONNECTED, if the call setup was successfully completed. In this case PORT_B delivers the connection of the selected subscriber, otherwise the data content of PORT_B is undefined. A call setup requires the sequence UNHOOK (DIGIT_N)* and the digit sequence must represent a valid number. HANG UP always leads to the complete termination of the call. If TIMEOUT occurs, HANG UP brings the software back into the initial state (DISCONNECTED)
State-Transition Testing: Example 2

State Chart
State-Transition Testing: Example 2

The minimal test strategy is to cover each state at least once.

A better solution is to cover each transition at least once, which leads, e.g., to the following tests …
State-Transition Testing: Example 2

State Chart

DISCONNECTED, unhook -> DIALING, hang up -> DISCONNECTED
State-Transition Testing: Example 2

State Chart

State Transition:
- DISCONNECTED, unhook -> DIALING
- DIALING, timeout -> TIME OUT OCCURRED
- TIME OUT OCCURRED, hang up -> DISCONNECTED
State-Transition Testing: Example 2

State Chart

DISCONNECTED, unhook -> DIALING, Digit 0..9 -> DIALING, Digit 0..9 -> DIALING, dialed number valid -> CONNECTED, hang up -> DISCONNECTED
State-Transition Testing: Example 2

State Chart

- DISCONNECTED, unhook ->
- DIALING, Digit 0..9 ->
- DIALING, Digit 0..9 ->
- DIALING, dialed number invalid -> INVALID NUMBER,
- timeout -> TIMEOUT OCCURRED, hang up -> DISCONNECTED
State-Transition Testing: Example 2

The minimal test strategy is to cover each state at least once.

A better solution is to cover each transition at least once, which leads, e.g., to the following tests:

- DISCONNECTED, unhook -> DIALING, hang up -> DISCONNECTED

- DISCONNECTED, unhook -> DIALING, timeout -> TIMEOUT OCCURRED, hang up -> DISCONNECTED

- DISCONNECTED, unhook -> DIALING, Digit 0..9 -> DIALING, Digit 0..9 -> DIALING, dialed number valid -> CONNECTED, hang up -> DISCONNECTED

- DISCONNECTED, unhook -> DIALING, Digit 0..9 -> DIALING, Digit 0..9 -> DIALING, dialed number invalid -> INVALID NUMBER, timeout -> TIMEOUT OCCURRED, hang up -> DISCONNECTED

... and so on ...
State-Transition Testing: Example 2

The minimal test strategy is to cover each state at least once.

A better solution is to cover each transition at least once, which leads, e.g., to the following tests:

- DISCONNECTED, unhook -> DIALING, hang up -> DISCONNECTED
- DISCONNECTED, unhook -> DIALING, timeout -> TIMEOUT OCCURRED, hang up -> DISCONNECTED
- DISCONNECTED, unhook -> DIALING, Digit 0..9 -> DIALING, Digit 0..9 -> DIALING, dialed number valid -> CONNECTED, hang up -> DISCONNECTED
- DISCONNECTED, unhook -> DIALING, Digit 0..9 -> DIALING, Digit 0..9 -> DIALING, dialed number invalid -> INVALID NUMBER, timeout -> TIMEOUT OCCURRED, hang up -> DISCONNECTED

Furthermore, it is useful to test all events, if transitions can be initiated by more than one event. The result is a hierarchy of test techniques:

all states ≤ all transitions ≤ all events

Important: Do not forget to test the failure treatment!
State-Transition Testing: Example 2

<table>
<thead>
<tr>
<th>Event</th>
<th>DISCONNECTED</th>
<th>DIALING</th>
<th>CONNECTED</th>
<th>INVALID NUMBER</th>
<th>TIMEOUT OCCURRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>unhook</td>
<td>DIALING</td>
<td>FAILURE</td>
<td>FAILURE</td>
<td>FAILURE</td>
<td>FAILURE</td>
</tr>
<tr>
<td>hang up</td>
<td>FAILURE</td>
<td>DISCONNECTED</td>
<td>DISCONNECTED</td>
<td>DISCONNECTED</td>
<td>DISCONNECTED</td>
</tr>
<tr>
<td>digit_0</td>
<td>DISCONNECTED</td>
<td>DIALING</td>
<td>CONNECTED</td>
<td>INVALID NUMBER</td>
<td>TIMEOUT OCCURRED</td>
</tr>
<tr>
<td>digit_9</td>
<td>DISCONNECTED</td>
<td>DIALING</td>
<td>CONNECTED</td>
<td>INVALID NUMBER</td>
<td>TIMEOUT OCCURRED</td>
</tr>
<tr>
<td>timeout</td>
<td>FAILURE</td>
<td>TIMEOUT OCCURRED</td>
<td>FAILURE</td>
<td>TIMEOUT OCCURRED</td>
<td>TIMEOUT OCCURRED</td>
</tr>
<tr>
<td>dialed number invalid</td>
<td>FAILURE</td>
<td>CONNECTED</td>
<td>FAILURE</td>
<td>FAILURE</td>
<td>FAILURE</td>
</tr>
<tr>
<td>dialed number valid</td>
<td>FAILURE</td>
<td>CONNECTED</td>
<td>establish connection</td>
<td>FAILURE</td>
<td>FAILURE</td>
</tr>
</tbody>
</table>
State-Transition Testing: Example 2

State Chart
with FAILURE state
Black-Box Testing Techniques

- Equivalence class partitioning (ECP)
- Boundary value analysis (BVA)
- Cause-effect graphing
- Combinatorial testing
- State transition testing (State-based testing)
- **Exploratory testing**
- Usability testing
- A/B testing (UX)
Exploratory Testing

• = Error guessing (?), happy testing, ...
• Not the same as ‘random testing’
• Always worth doing (on top of “regular” testing)
• Can trigger failures that systematic techniques miss
• Consider
  • ”What is the craziest thing we can do?”
  • Intuition / Experience / Brainstorming
  • Past failures / Lists in literature
• Tools
  • http://www.softwaretestinghelp.com/tools/top-17-exploratory-testing-tools/
Exploratory Testing

• Inventors:
  • Cem Kaner, James Bach (1990s)

• Definition:
  • “Exploratory testing is simultaneous learning, test design, and test execution.”

• Elements / Variants
  • Charter: defines mission (and sometimes tactics to use)
    • Example: “Check UI against Windows interface standards”
  • Session-based test management:
    • Defects + Notes + Interviews of the testers
Exploratory Testing - 5 Steps

One type of exploratory testing -- also called session based test management (SBTM Cycle) – has the following 5 stages:

**STEP 1: Create a Bug Taxonomy (classification)**

- Categorize common types of faults found in the past projects
- Analyze the root causes of the problems or faults
- Find the risks and develop ideas to test the application
Exploratory Testing - 5 Steps

STEP 2: Develop Test Charter

- Test Charter should suggest
  - what to test
  - how it can be tested
  - what needs to be looked at
- Test ideas are the starting point of exploration testing
- Test charter helps determine how the end user could use the system
Exploratory Testing - 5 Steps

STEP 3: Set a Time Box

• A single tester or a pair of testers is working not less than 90 minutes
• There should not be any interruption in those 90 minutes sessions
• Time box can be extended or reduced by 45 minutes
• This session encourages testers to react on the response from the system and prepare for the correct outcome
Exploratory Testing - 5 Steps

STEP 4: Review Results
- Evaluation of the defects
- Learning from the testing
- Analysis of coverage areas

STEP 5: Have a Debriefing
- Compilation of the output results
- Compare the results with the charter
- Check whether any additional testing is needed
Exploratory Testing

During exploratory execution, following needs to be done:

• Mission of testing should be very clear

• Keep notes on what needs to be tested, why it needs to be tested and the assessment of the product quality

• Tracking of questions and issues raised during exploratory testing

• Better to pair up the testers for effective testing

• The more we test, more likely to execute right test cases for the required scenarios
Exploratory Testing

It is very important to take document and monitor the following:

• Test Coverage - Whether we have taken notes on the coverage of test cases and improve the quality of the software

• Risks - Which risks needs to be covered and which are all important ones?

• Test Execution Log - Recordings on the test execution

• Issues / Queries - Take notes on the question and issues on the system
Exploratory Testing

**PRO**

- Useful when requirement documents are not available or only partially available
- Involves Investigation process which helps find more bugs than normal testing
- Helps to expand the imagination of testers by executing more and more test cases which finally improves productivity as well
- This testing drill down to smallest part of application and covers all the requirements
- This testing covers all the types of testing and it covers various scenarios and cases
- Encourages creativity and intuition
- Generation of new ideas during test execution

**CON**

- Purely depends on the tester skills
- Limited by domain knowledge of the tester
- Not suitable for Long execution time (e.g., scientific programs)
Black-Box Testing Techniques

- Equivalence class partitioning (ECP)
- Boundary value analysis (BVA)
- Cause-effect graphing
- Combinatorial testing
- State transition testing (State-based testing)
- Exploratory testing
- **Usability testing**
- A/B testing (UX)
Quality Attributes – ISO 9126

(ISO 25000)

- Functionality: The capability to provide functions that meet stated or implied needs.
- Portability: The capability to be transferred from one environment to another.
- Reliability: The capability to maintain a specified level of performance.
- Maintainability: The capability to be modified.
- Usability: The capability to be understood, learned, used and attractive to user.
- Efficiency: The capability to provide appropriate performance, relative to amount of resources used, under stated conditions.
Software Product Quality Model – ISO 25010 Standard
Testing Usability Requirements

<table>
<thead>
<tr>
<th>Problem counts</th>
<th>Task time</th>
<th>Keystroke counts</th>
<th>Design-level reqs</th>
<th>Product-level reqs</th>
<th>Guideline adherence</th>
<th>Development process reqs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: At most 1 of 5 novices shall encounter critical problems during tasks Q and R. At most 5 medium problems on list.</td>
<td>R2: Novice users shall perform tasks Q and R in 15 minutes. Experienced users tasks Q, R, S in 2 minutes.</td>
<td>R3: Recording breakfast shall be possible for guests. No mouse.</td>
<td>R6: System shall use screen pictures in app. xx, buttons work as app. yy.</td>
<td>R7: For all code fields, user shall be able to select value from drop-down list.</td>
<td>R8: System shall follow style guide zz. Menus shall have at most three levels.</td>
<td>R9: Three prototype versions shall be made and usability tested during design.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opinion poll</td>
<td>Score for understanding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R4: 80% of users shall find system easy to use. Recommend system to others.</td>
<td>R5: Show 5 users 10 common errors marked &quot;large.&quot; Ask for the cause. 80% of the time, the cause is due to the system’s design.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Testing Usability Requirements

How to test:
- Define several (typical) usage scenarios involving tasks Q and R
- Select test users and classify as 'novice' and 'experienced'
- Let 5 (or better 10, 15) novices perform the scenarios
- Observe what problems they encounter
- Classify and count observed problems
Usability Test Types + Environment

Rubin’s Types of Usability Tests (Rubin, 1994, p. 31-46)

**Exploratory test** – early product development

**Assessment test** – most typical, either early or midway in the product development

**Validation test** – confirmation of product’s usability

**Comparison test** – compare two or more designs; can be used with other three types of tests
Usability Testing – What? How?

- **Test Focus**
  - **Understandability**
    - Easy to understand?
  - **Ease of learning**
    - Easy to learn?
  - **Operability**
    - Matches purpose & environment of operation?
    - Ergonomics: color, font, sound,...
  - **Communicativeness**
    - In accordance with psychological characteristics of user?

- **Test Environments**
  - Free form tasks
  - Procedure scripts
  - Paper screens
  - Mock-ups
  - Field trial
Example: Evaluating UI Designs

- Inspection Methods
  - Heuristic Evaluation
    - Cognitive Walkthrough
  - Guidelines Review
- Usability Testing
  - Laboratory Experiment
  - Field Study
Example: Evaluating UI Designs

Inspection Methods

- Heuristic Evaluation
  - evaluates design on how well it supports user in learning task
  - usually performed by expert in cognitive psychology
  - expert ‘walks though’ design to identify potential problems using psychological principles
  - Scenarios may be used to guide analysis

Usability Testing

Field Study

Laboratory Experiment

Cognitive Walkthrough

- evaluates design on how well it supports user in learning task
- usually performed by expert in cognitive psychology
- expert ‘walks though’ design to identify potential problems using psychological principles
- Scenarios may be used to guide analysis
Example: Evaluating UI Designs

Inspection Methods
- Cognitive Walkthrough
  - usability criteria (heuristics) are identified
  - design examined by experts to see if these are violated

Usability Testing
- Laboratory Experiment
- Field Study

Guidelines
- Heuristic Evaluation
Heuristic Evaluation by Inspection

List of 10 Heuristics according to (Nielsen, 2005):

<table>
<thead>
<tr>
<th></th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Visibility of system status</td>
</tr>
<tr>
<td>2</td>
<td>Match between the system and the real world</td>
</tr>
<tr>
<td>3</td>
<td>User control and freedom</td>
</tr>
<tr>
<td>4</td>
<td>Consistency and standards</td>
</tr>
<tr>
<td>5</td>
<td>Error prevention</td>
</tr>
<tr>
<td>6</td>
<td>Recognition rather than recall</td>
</tr>
<tr>
<td>7</td>
<td>Flexibility and efficiency of use</td>
</tr>
<tr>
<td>8</td>
<td>Aesthetic and minimalist design</td>
</tr>
<tr>
<td>9</td>
<td>Help users recognize, diagnose, and recover from errors</td>
</tr>
<tr>
<td>10</td>
<td>Help and documentation</td>
</tr>
</tbody>
</table>

List violations of heuristics:

Rank by severity: 0...4
0: positive (or neutral) aspect of system
...
4: major, catastrophic aspect of system

(See example report on course wiki!)


Example: Evaluating UI Designs

Written guidelines recommended for larger projects:
- Screen layout
- Appearance of objects
- Terminology
- Wording of prompts and error messages
- Menu’s
- Direct manipulation actions and feedback
- On-line help and other documentation
Example: Evaluating UI Designs

**Inspection Methods**
- Heuristic Evaluation

**Usability Testing**
- Laboratory Experiment
- Field Study

Usability testing in a controlled environment:
- There is a test set of users
- They perform pre-specified tasks
- Data is collected (quantitative and qualitative)
- Take mean and/or median value of measured attributes
- Compare to goal or another system
Example: Evaluating UI Designs

1. Direct observation in actual use
   - discover new uses
   - take notes, don’t help, chat later

2. Logging actual use
   - objective, not intrusive
   - great for identifying errors
   - which features are/are not used
   - privacy concerns

Cognitive Walkthrough

Guidelines Review

Usability Testing

Field Study
Example: Evaluating UI Designs

3. Questionnaires and interviews with real users
   - ask users to recall critical incidents
   - questionnaires must be short and easy to return

4. Focus groups
   - 6-9 users
   - skilled moderator with pre-planned script
   - computer conferencing??

5. On-line direct feedback mechanisms
   - initiated by users
   - may signal change in user needs
   - trust but verify

6. Bulletin boards and user groups
Black-Box Testing Techniques

- Equivalence class partitioning (ECP)
- Boundary value analysis (BVA)
- Cause-effect graphing
- Combinatorial testing
- State transition testing (State-based testing)
- Exploratory testing
- Usability testing
- A/B testing (UX)
A/B Testing

Two GUI Versions A & B
A/B Testing (cont’d)

Tool support

Visual Website Optimizer divides traffic between the two versions

Randomly selected

Two GUI Versions A & B
A/B Testing (cont’d)

Tools:
https://blog.crazyegg.com/2014/06/25/best-testing-software/

More Sales?  

Two GUI Versions A & B
A/B Testing – Real-World Example

Former US president
Obama’s 2008
Election campaign

Source:
A/B Testing – Real-World Example

Button variations:
A/B Testing – Real-World Example

Media variations:

- Family Image
- Change Image
- Barack’s Video
- ...
(6 alternatives in total)
A/B Testing – Real-World Example

In total
4 x 6 = 24 combinations
(including the original button and medium)

Sign-up rates for each section (button, medium)
A/B Testing – Real-World Example

In total

4 x 6 = 24 combinations

(including the original button and medium)

Combination 11:
“Learn More” & “Family Image”

Sign-up rates for each <button, medium>-combination
A/B Testing – Real-World Example

Former US president Obama’s 2008 Election campaign

Combination 11: “Learn More” & “Family Image”

A/B Testing – Multivariate Testing

- Only 2 items in previous example
A/B Testing (cont’d)

What to vary …

• Call-To-Actions – Placement, wording, size
• Copywriting – Value propositions, product descriptions
• Forms – Their length, field types, text on the forms.
• Layout – Homepage, content pages, landing pages
• Product pricing – Try testing for revenue by testing your prices
• Images/Videos – Their placement, content and size
• Amount of content on the page (short vs. long)

Link:

http://conversionxl.com/how-to-build-a-strong-ab-testing-plan-that-gets-results/
A/B Testing -- Tools

A/B Split Test Significance Calculator by VWO

• A widely used tool for calculating the significance of your A/B testing results.

A/B Split and Multivariate Test Duration Calculator by VWO

• The calculator allows you to calculate maximum duration for which your test should run.

Crazyegg, Inspectlet, Clicktale and Mouseflow

• Heatmap software for tracking your visitor’s behavior on your site. You can get good data for hypotheses generation.