How to improve your manual testing
... without getting bored!

Alex Schladebeck

(exploratory testing and session-based test management)
Short introduction

- BREDEX
  - Individual software development
    - Enterprise applications
  - Quality assurance
    - Based on Jubula
    - Tool-independent services

- This talk
  - Why exploratory testing?
  - How to use it
  - How to plan and organise it
Wait, what?!

*If you’re such test automation champions, why are we talking about manual testing?*
Testing as detective work

Testing is gathering information with the intention of informing a decision – Jerry Weinberg
Gathering information

- It is impossible to plan tests in advance to cover every possible condition
- If you try to plan/write all possible tests, you will only ever write them, never execute them
- You can only plan based on what you know
  - *If the map and the territory differ, believe the territory*

Elisabeth Hendrickson, Explore It!
Testing vs. Checking

Intended behaviour?  
Checking

Are there any other risks?  
Testing (Exploring)

Scripted test  
(manual or automated)

Unscripted test  
(manual only)

Are there any other risks?

Testing (Exploring)

Intended behaviour?

Checking
A couple of videos
1. Observational blindness

- Focus on one aspect makes us miss other things
  - *The narrower the view, the wider the ignorance*

- Errors avoid well-tested paths
2. Orders of ignorance

because as we know, there are known knowns—there are things we know we know...

We also know there are known unknowns; that is to say we know there are some things we do not know...

But there are also unknown unknowns—the ones we don't know we don't know.

Donald Rumsfeld, US Secretary of Defense.
Department of Defense News Briefing. Feb 12 2002
2. Orders of ignorance

- **0th order ignorance**
  - We provably know something
  - a scripted test that shows us the same questions being answered again and again

- **1st order ignorance**
  - We know what we don’t know
  - scripted and exploratory testing can help here: ask the questions that will give us the knowledge

- **2nd order ignorance**
  - We don’t even know what we don’t know
  - exploratory testing to find out what we don’t know
    - Repeat thinking, not steps
    - Find the holes in the net
Not scripted? Clicking around?
Exploratory testing

- Exploratory testing is **not ad hoc**
- Test design, execution and analysis all happen at the same time
  - No scripting – but normal test design practices can be used (personas, decision tables, ...)
  - Pre-defined test results not necessary
  - Thoughts and process not documented in advance
- Situational and emergent practice – context is important
  - The puzzle itself influences the strategies for solving it
  - Steps and test cases are designed based on context and on observation

... This sounds kind of difficult, but it’s actually our normal human practice of reasoning.
Exploratory testing
When should I use Exploratory Testing?

- To find out unknown unknowns
- To find out how the software works
  - E.g. when the specifications aren’t detailed
- When you need to find important information quickly
- To keep test script writing to a minimum
- To find new information and new questions
- To avoid observational blindness
How do I use Exploratory Testing?

- How do I know how the program should behave?
- What should I test?
- How do I know what is a problem?
Sources and oracles

Sources
- Places you can get information
  - Specification
  - Requirements
  - Documentation
  - Experience
  - Sales promises
  - Discussions
  - Customer
  - Users
  - Product menus, toolbars, context menus

Oracles
- How to know if there is a problem
  - Mental models
  - Own gut feeling
  - Consistency
  - *Is there a problem here?*

Heuristics
- Rules of thumb to guide testing and to identify problems
Heuristics – what to look at / for

- CRUD: Create, read, update, delete
- Goldilocks: Too big, too small, just right
- Users and scenarios: personas, extreme personalities, soap operas, tours
- Follow the data: Enter – Search – Report – Export – Import – Update – View
- HICCUPSF: History, Image, Comparable Products, Claims, User expectations, Product, Purpose, Statutes, Familiar Problems
- Areas of potential instability: interoperation, multiple files, network, ...
Advantages and disadvantages

Advantages
- Less preparation than scripting
- Complete specification not required
- Adaptable to new information
- Important errors found quickly
- More interesting

Disadvantages
- Not reviewable in advance
- Not reproducible
- Not good in low-availability environments
- Hard to organise and control
- Where to start?
  - Session-based test management
Session-based test management

A way of organising and reporting on Exploratory Testing

<table>
<thead>
<tr>
<th>Session-based test management recipe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charters</td>
</tr>
<tr>
<td>Sessions</td>
</tr>
<tr>
<td>Notes</td>
</tr>
<tr>
<td>Debriefing</td>
</tr>
</tbody>
</table>
Define charters

Prioritise and assign

Session – down the rabbithole!

Observe and note

Debrief
Charters

Explore [area] with [resources] to discover [information]

- Charters come from:
  - Stakeholders, team, risks, quality attributes, places where we require information, the nightmare headline game

- Defined in advance **but can be changed**

- (Re)prioritised before each new session
Session – down the rabbit hole

- **Uninterrupted** timebox
  - 30-120 minutes

- One charter
  - Down the rabbit hole
  - If you find another hole, write it down → new charter for next session

- Use heuristics to create and execute tests
Notes

- What did I do?
- What did I observe?
  - Videos, screenshots
- How do I feel about the quality of this charter?
- Ratio test:setup
- Ratio charter:opportunity
  - Opportunity = new areas that occur while testing
Debrief

PROOF
- Past: what happened
- Results: what was achieved?
- Obstacles: what blocked the testing?
- Outlook: What needs to be done?
- Feeling: how do I feel about this?
<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.00</td>
<td>Define, prio, choose charters</td>
</tr>
<tr>
<td>8.30</td>
<td>Session</td>
</tr>
<tr>
<td>10.00</td>
<td>Debrief</td>
</tr>
<tr>
<td>10.15</td>
<td>Break</td>
</tr>
<tr>
<td>10.30</td>
<td>Define, prio, choose charters</td>
</tr>
<tr>
<td>10.45</td>
<td>Session</td>
</tr>
<tr>
<td>12.00</td>
<td>Debrief</td>
</tr>
<tr>
<td>12.15</td>
<td>Lunch</td>
</tr>
<tr>
<td>1.15</td>
<td>Define, prio, choose charters</td>
</tr>
<tr>
<td>1.30</td>
<td>Session</td>
</tr>
<tr>
<td>3.00</td>
<td>Debrief</td>
</tr>
<tr>
<td>3.15</td>
<td>Bug entry</td>
</tr>
</tbody>
</table>
When to reproduce / enter bugs?
A good mix

Daily / in each sprint
- Automated tests
  - Regression tests
  - As part of acceptance testing for new features → tests that will become regression tests
- Exploratory testing
  - Manual testing of new features
  - Can use SBTM

Before a release
- Manual test checklists
  - Semi-scripted
- Exploratory testing days with session based test management
Thanks!

alexandra.schladebeck@bredex.de
@alex_schl

www.bredexsw.com
testing.bredex.de

www.eclipse.org/jubula
References

- Jerry Weinberg, Perfect Software and other Illusions about Testing
- Exploratory testing
  - Explore it! Elisabeth Hendrickson (Pragmatic Bookshelf)
  - [http://www.developsense.com/blog/2012/04/all-oracles-are-heuristic/](http://www.developsense.com/blog/2012/04/all-oracles-are-heuristic/)
  - [http://www.satisfice.com/articles/what_is_et.shtml](http://www.satisfice.com/articles/what_is_et.shtml)
- Cheat sheets and mnemonics
  - [http://www.qualityperspectives.ca/resources_mnemonics.html](http://www.qualityperspectives.ca/resources_mnemonics.html)
- Session-based test management