Software Testing: A Profession of Paradoxes?

Zeger van Hese,
CTG, Belgium
Software Testing: a profession of paradoxes?

Zeger Van Hese
zeger.van_hese@ctg.com
« Life is a riddle...

...unfortunately the answer is not written on the back of anything »

(Old Chinese proverb)
Overview

- Goal
- Definitions
- Catch-22 of testing
- Paradoxes in software testing
  - Testing realities
  - Test automation
  - Test reporting
  - Test team
  - Unit testing
  - Usability testing
- Conclusion
Goal

- Awareness
- Food for thought
- Solution?
Some definitions

Catch-22

- Satirical fiction novel
  Joseph Heller, 1961

- Trap preventing soldiers from leaving the army

- Universal meaning:
  - No-win situation
  - Situation in which you cannot obtain A without B, but B requires A

- Job seeker’s catch-22
Catch-22 of software testing

- Main cause: complexity of software
- Testing is potentially endless

\[
\text{We end testing when we’re finished…} \\
\text{… but we’re never finished!}
\]

- When to stop?
  - Pessimistic
    - Time - budget – test cases exhausted
  - Optimistic
    - Reliability meets requirements
    - Benefit from continuing < testing cost
Paradoxes in software testing: 1. Testing realities

Who watches the watchmen?
- Humans err - testers are human
- Inattentional blindness
- Classic testing mistakes (M. Heusser)

Testing can’t show that bugs don’t exist
- No guarantee there are more bugs to find

Testing influences its own outcome
- Observer effect
- Complexity barrier (B. Beizer)

“I was most surprised the day I realized the paradox of - how am I going to write tests for the tests that I’m writing?”
(Sara Ford)
Paradoxes in software testing: 1. Testing realities cont

Failure breeds success

- Scientific proof (*university of Exeter*)
- We learn more from failures than from mistakes:
  - Lateral thinking
  - Improvement & planning
  - Honest

The biggest bugs are the hardest to find

*Intermittent bug = a mysterious and undesirable behavior of a system, observed at least once, that we cannot yet manifest on demand* (James Bach)

The more bugs you find...
Paradoxes in software testing:
2. Pesticide paradox

Every method you use to prevent or find bugs leaves a residue of subtler bugs against which those methods are ineffectual
(Boris Beizer)

- Software builds resistance to tests
- Tests become ineffective
- Major drawback of automated testing
  - But: important in agile projects (regression)
- Solution?
  - Continually design new testcases
  - Model based testing
Paradoxes in software testing: 3. Simpson’s paradox

or a possible test reporting paradox

- Two or more sets of data lead to one conclusion when evaluated individually, but lead to an opposite conclusion when the sets are combined

- Beware of aggregated data!

**Figure 1** Comparing System A and System B

<table>
<thead>
<tr>
<th>Manual Tests</th>
<th>System A</th>
<th>System B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tests passed</td>
<td>50</td>
<td>15</td>
</tr>
<tr>
<td>Total number of tests</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>% tests passed</td>
<td>25%</td>
<td>15%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Automated Regression Tests</th>
<th>System A</th>
<th>System B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tests passed</td>
<td>85</td>
<td>300</td>
</tr>
<tr>
<td>Total number of tests</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>% tests passed</td>
<td>85%</td>
<td>75%</td>
</tr>
</tbody>
</table>

**Figure 2** Combined Data for Manual and Automated Regression Tests

<table>
<thead>
<tr>
<th>Combined Data</th>
<th>System A</th>
<th>System B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tests passed</td>
<td>135</td>
<td>315</td>
</tr>
<tr>
<td>Total number of tests</td>
<td>300</td>
<td>500</td>
</tr>
<tr>
<td>Score</td>
<td>45%</td>
<td>63%</td>
</tr>
</tbody>
</table>

*Source: James McCaffrey – software testing paradoxes*
Paradoxes in software testing: 4. Test Team paradox

- Testing = a procedure for critical evaluation
  \((American\ heritage\ dictionary)\)
- Constantly critical attitude
- Successful teams need to be negative
- Key: critical focus – positive outlook
  - Highlight good points
  - Set clear goals
  - Look for positives in software
  - Try to find bugs early
  - Temper your enthusiasm
  - Try to bring bad news in a positive way
Bug Fix Bingo

Rules of the Game
1. Bingo squares are marked off when a developer makes the matching statement during bug fix sessions.
2. Testers must call "Bingo" immediately upon completing a line of 5 squares either horizontally, vertically or diagonally.
3. Statements that arise as a result of a bug that later becomes "deferred", "as designed", or "not to be fixed" should be reclassified as not marked.
4. Bugs that are not reported in an incident report can not be used.
5. Statements should also be recorded against the bug in the defect tracking system for later confirmation.
6. Any tester that marks off all 25 statements should be awarded 2 weeks stress leave immediately.
7. Any developer found using all 25 statements should be seconded into the test group for a period of no less than 6 months for re-education.

1. "It works on my machine."
2. "Where were you when the program blew up?"
3. "Why do you want to do it that way?"
4. "You can’t use that version on your system."
5. "Even though it doesn’t work, how does it feel?"
6. "Did you check for a virus on your system?"
7. "Somebody must have changed my code."
8. "It works, but it hasn’t been tested."
9. "THIS can’t be the source of THAT."
10. "I can’t test everything!"
11. "It’s just some unlucky coincidence."
12. "You must have the wrong version."
13. "I haven’t touched that module in weeks!"
14. "There is something funky in your data."
15. "What did you type in wrong to get it to crash?"
16. "It must be a hardware problem."
17. "How is that possible?"
18. "It worked yesterday."
19. "It’s never done that before."
20. "That’s weird..."
21. "That’s scheduled to be fixed in the next release."
22. "Yes, we knew that would happen."
23. "Maybe we just don’t support that platform."
24. "It’s a feature, We just haven’t updated the specs."
25. "Surely nobody is going to use the program like that."

Read what people are saying about Bug Fix Bingo !!!
"Thanks Bug Fix Bingo, You have made defect review meetings fun again !!!"
"I used to fall asleep when meeting with developers, now with BFB, I anticipate every word"

Paradoxes in software testing: 5. Test automation paradoxes

- **N° 1: Bugs in automated tests**
  - Test automation = software development
  - Who watches the watchmen – revisited
  - Tests for tests for tests for tests for tests for…
  - False alarms – Silent horrors

- **N° 2: Automated regression tests**
  - Least reliable when we need them the most

Source: Brett Pettichord (2003)
Paradoxes in software testing: 6. Developer testing paradox

- Unit testing’s benefits:
  - Best option to improve quality
  - Opportunity to catch bugs early
- But still… not a general practice

“Too time-consuming”

“It only proves that the code does what the code does”

“I don’t need unit tests”

“Integration tests will catch the bugs anyway”

Common misconceptions

Sources: Alberto Savoia (2005) & IPL – “Why bother to unit test”
Paradoxes in software testing: 7. Usability paradox

- Usability testing: important & expensive
- When do you plan?
  - Early?
    - Heading in right direction
    - Changes made when they’re least expensive
  - Later?
    - Software available to user – all valid feedback
    - No time/money to correct
- True story
- Prototyping
Conclusion

“How wonderful that we have met with a paradox. Now we have some hope of making progress”

Niels Bohr

aware think re-think expand
Thank you for your time!

Questions/Comments
[zeger.van_hese@ctg.com]