CONTENT

– Motivation
– Research questions
– Case study
– Research methods
– Results
– Conclusions
– Discussion
MOTIVATION

Testing?

- Find Bugs!
- Fast Deployment!
- Quality!
- Short Release Cycles!
MOTIVATION

# Test Cases?
RESEARCH QUESTIONS

RQ1: “Does the discovery of bugs push the writing of tests?”

RQ2: “Do existing tests prevent the occurrence of bugs?”

RQ3: “Is the system’s architecture designed in a way, that addition of new tests is easy?”
Project ePostOffice

- Digital platform for receiving and sending physical and electronic mail
- Mixed technologies
- Ongoing for 6 years

- Confluence (Wiki, Requirements)
- Jira (Issue-Tracker, SCRUM)
- Git (Version control)
RESEARCH METHODS 1/3

Issues, Commits, File changes

Jira Software

Bitbucket

Defective classes not covered by Tests
Possibly recurring Bugs
Defective classes covered by Tests
Bug statistics

SQLite
RESEARCH METHODS 2/3

Defective classes not covered by Tests
- Why are they not covered by tests? => Assess testability of code
- How could we test them? => Suggest improvements

Possibly recurring Bugs
- Which ones are really recurring bugs? => Manual inspection
- Why did they occur again? => Study history
Defective classes covered by Tests
- How good are those tests? => Apply mutation testing to them

Bug statistics
- Any anomalies in development history? => Ask developers
- Proneness to bugs of different components? => Ask developers
Method
- Manual inspection of 200 issues/bugfixes
- Classification of Testability

Results
- Testability is bad in most cases
- Reasons:
  - Violation of Single Responsibility Principle
  - Too many dependencies
RESULTS 2/5
RECURRING BUGS

Method
- Tool to compare diffs on method level
- List of commits which changed same method
- Manual inspection of suggestions to find real recurring bugs

Results
- C# files: 20 possibly recurring bugs, no really recurring bugs
- Java files: 135 possibly recurring bugs, 7 similar bugs
- No tests added after first and also not after second bug
Method

- Apply mutation testing to bugs which are possibly covered by tests
- Mutate focal methods under test

Results

- Newer components achieve higher scores
- If only main control flow is covered by tests, many mutations are not detected

<table>
<thead>
<tr>
<th>Component</th>
<th># Classes</th>
<th># Mut</th>
<th># Kill</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelOrd</td>
<td>2</td>
<td>11</td>
<td>9</td>
<td>81.8%</td>
</tr>
<tr>
<td>GKPortal</td>
<td>2</td>
<td>24</td>
<td>21</td>
<td>87.5%</td>
</tr>
<tr>
<td>PkPortal</td>
<td>3</td>
<td>66</td>
<td>44</td>
<td>66.7%</td>
</tr>
<tr>
<td>RecPref</td>
<td>2</td>
<td>31</td>
<td>26</td>
<td>83.9%</td>
</tr>
<tr>
<td>Transfer</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>100.0%</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>134</td>
<td>102</td>
<td>76.1%</td>
</tr>
</tbody>
</table>
RESULTS 4/5
BUG REPORT HISTORY

Method

- Retrieve bugs from database
- Plot and inspect timeline
- Approach developers

Results

- Ascents right before and right after releases
- Clear “bend” in Nov 2016
RESULTS 5/5
PRONENESS TO BUGS

Method
– Use query to get number of bugs per component
– Approach developers for “worst 15” components

Results
– 58.6% of buggy components are only affected by one bug
– Buggy components are usually large and complex
CONCLUSIONS

RQ1: “Does the discovery of bugs push the writing of tests?”

RQ2: “Do existing tests prevent the occurrence of bugs?”

RQ3: “Is the system’s architecture designed in a way, that addition of new tests is easy?”
«It is all about testability!»
Wenn Sie diesen Text lesen können, müssen Sie die Folie im Post-Menü mit der Funktion «Folie eingefügen» erneut eingefügen. Sonst kann die Fläche nicht eingefärbt werden!