Early fault detection with model-based testing

ACM SIGPLAN Erlang Workshop 2008
Jonas Boberg
Victoria, British Columbia, Canada, September 27, 2008
Agenda

- Model-based testing
- Research approach
- Results
- Challenges and recommendations
- Conclusions
Model-based testing

The whats and why

Model-based testing
Research approach
Results
Challenges and recommendations
Conclusions
“Traditional” test case design
Trend: increasing complexity

Increasingly complex requirements on interaction between systems

Difficult to achieve high coverage with hand-crafted test cases
Model-based testing (MBT)

Specification + “Informal requirements” → Model → Test case

Copyright 2008
Steps of model-based testing

- Build an abstract model of system
- Validate the model
- Generate and execute test cases
- Assigning pass/fail verdict
- Analyzing the execution result.
Steps of model-based testing - automation

Can be automated?

- Build an abstract model of system
- Validate the model
- Generate and execute test cases
- Assigning pass/fail verdict
- Analyzing the execution result.
-module(imap_eqc).
-behaviour(eqc_statem).

command(S) ->
    ...

precondition(State, {call, _, select, [CPid,_]}) ->
    min_state(client(CPid,State), ?AUTH)

postcondition(State, {call, _, select, [CPid,_], Result} ->
    is_status(Result, ?OK_RESP);

eqc:sample( ... )
eqc:quickcheck( ... )
Why is MBT feasible?

Only small part of system-code typically concerned with functionality

Functionality: 10-30%

- Glue
- Concurrency
- Fault-tolerance
- Distribution
- ...
Why is MBT feasible?

You don’t model the full system
Abstractions are applied when modeling
Model-based testing advantages

- Fault prevention
- Reduced cost of updating test cases
Model-based testing in industry

AGEDIS Case Studies (France Telecom, Intrasoft and IBM)

Dalal et al.
Identifies organizational obstacles to practice introduction
Model-basted testing must be well integrated in the test process
Pretschner et al

“Model-based test cases does not detect more faults than hand-crafted test cases”
“Tests were executed after the system was completely implemented”

Artho et al.

component level testing of NASAs K9 planetary rover

“the testing was conducted after the implementation of the system was finished”
Research approach

How we applied model-based system testing to a large scale Erlang system
Project: Messaging Gateway

Study executed as a process improvement initiative

Goal: reduce fault-slip-through to acceptance testing
Costs of faults-slip-through

- Higher cost of tracking and fixing faults
- Reduced confidence in system
- Build and deploy additional release candidates
Boehm curve

Central question

How can model-based testing be applied at the system-level, to enable early fault-detection and increased confidence in the system?
Messaging gateway system

- E-mail gateway (EMGW)
  - IMAP Front-end
  - POP3 Backend
  - IMAP Backend

- Instant messaging gateway (IMGW)
  - Wireless Village front-end
  - Protocol Backend
  - Protocol Backend
  - Protocol Backend

- Shared components
Test approach

- Client
- Client
- Client

E-mail gateway

- POP3
- IMAP
Test approach

QuickCheck
Abstract State
Machine model

IMAP Client

E-mail gateway

POP3

IMAP
Hypothesis: Model-based will result in lower fault-slip through from system testing to customer acceptance testing
Study timeline

Three studied releases:

- Release X
- Release X+0.5
- Release X+1
Model-based testing in iterative development

For each iteration:

- Select requirements
- Model requirements
- Validate and execute
Fault-slip-through measurement

Unit test  Integration test  System test

Should have been found here

Fault found here

Customer acceptance test
Results
## Results - Release X

### EMGW

<table>
<thead>
<tr>
<th>Found/Belonging</th>
<th>System Test</th>
<th>Acceptance test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Test</td>
<td>1 (0)</td>
<td>1</td>
</tr>
<tr>
<td>Integration Test</td>
<td>1 (1)</td>
<td>0</td>
</tr>
<tr>
<td>External Integration Test</td>
<td>0 (0)</td>
<td>0</td>
</tr>
<tr>
<td>System Test</td>
<td>9 (6)</td>
<td>3</td>
</tr>
<tr>
<td>Acceptance Test</td>
<td>0 (0)</td>
<td>2</td>
</tr>
<tr>
<td><strong>Total found/test level</strong></td>
<td><strong>11 (7)</strong></td>
<td><strong>4</strong></td>
</tr>
</tbody>
</table>

### IMGW

<table>
<thead>
<tr>
<th>Found/Belonging</th>
<th>System Test</th>
<th>Acceptance test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Test</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Integration Test</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>External Integration Test</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>System Test</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Acceptance Test</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td><strong>Total found/test level</strong></td>
<td><strong>14</strong></td>
<td><strong>14</strong></td>
</tr>
</tbody>
</table>
### Results - Release X+1

#### EMGW

<table>
<thead>
<tr>
<th>Found/Belonging</th>
<th>System Test</th>
<th>Acceptance test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Test</td>
<td>24 (21)</td>
<td>3</td>
</tr>
<tr>
<td>Integration Test</td>
<td>1 (1)</td>
<td>0</td>
</tr>
<tr>
<td>External Integration Test</td>
<td>0 (0)</td>
<td>0</td>
</tr>
<tr>
<td>System Test</td>
<td>39 (26)</td>
<td>4</td>
</tr>
<tr>
<td><strong>Acceptance Test</strong></td>
<td><strong>0 (0)</strong></td>
<td><strong>10</strong></td>
</tr>
<tr>
<td><strong>Total found/test level</strong></td>
<td><strong>64 (48)</strong></td>
<td><strong>7</strong></td>
</tr>
</tbody>
</table>

#### IMGW

<table>
<thead>
<tr>
<th>Found/Belonging</th>
<th>System Test</th>
<th>Acceptance test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Test</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Integration Test</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>External Integration Test</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>System Test</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td><strong>Acceptance Test</strong></td>
<td><strong>0</strong></td>
<td><strong>2</strong></td>
</tr>
<tr>
<td><strong>Total found/test level</strong></td>
<td><strong>14</strong></td>
<td><strong>12</strong></td>
</tr>
</tbody>
</table>
Phase Input Quality (PIQ) =

Should have been found on earlier level

Total found on level

<table>
<thead>
<tr>
<th>Release/FST</th>
<th>X</th>
<th>X+0.5</th>
<th>X+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMGW FST to ST</td>
<td>18%</td>
<td>24%</td>
<td>39%</td>
</tr>
<tr>
<td>EMGW FST to AT</td>
<td>67%</td>
<td>–</td>
<td>41%</td>
</tr>
<tr>
<td>IMGW FST to ST</td>
<td>43%</td>
<td>–</td>
<td>14%</td>
</tr>
<tr>
<td>IMGW FST to AT</td>
<td>78%</td>
<td>–</td>
<td>86%</td>
</tr>
</tbody>
</table>
Challenges and recommendations

(As usual,) there is no silver bullet
Challenges

Finding suitable abstractions is difficult

We cannot execute partial tests
Recommendations

Iterative model development crucial
Test techniques are complementary
Challenges and recommendations

A substantial initial investment is required to integrate the model-based testing into the test process.

Visibility of results crucial to success

- both internal and external
Conclusions
Conclusions

Significantly more faults found by model-based testing

Results supports our hypothesis - model-based testing decreased the fault-slip-through
Questions?
Agenda

- Model-based testing
- Research approach
- Results
- Challenges and recommendations
- Conclusions