



Erlang Training and Consulting Ltd

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 whys

Model-based testing

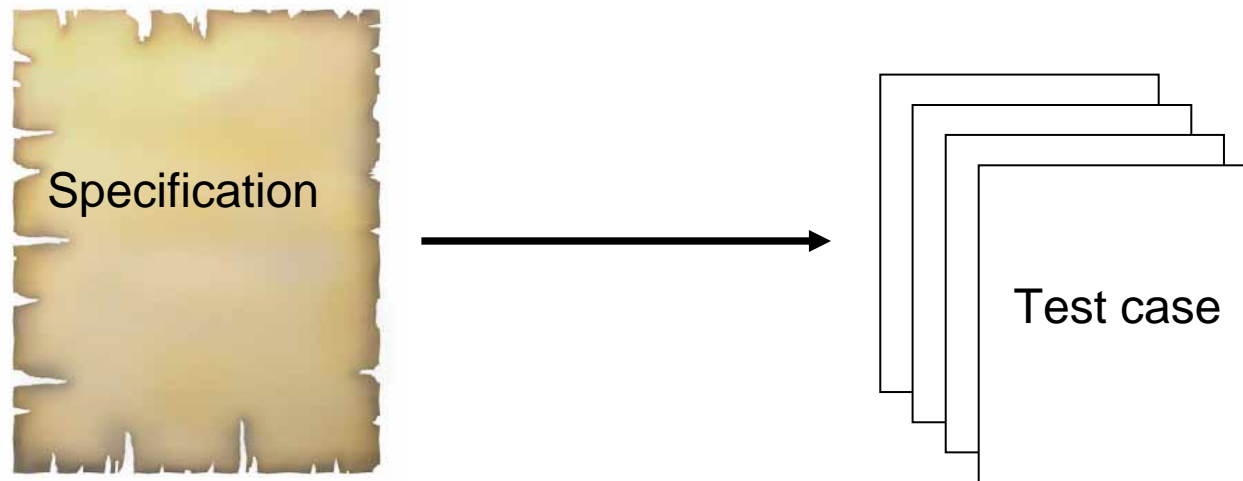
Research approach

Results

Challenges and
recommendations

Conclusions

“Traditional” test case design

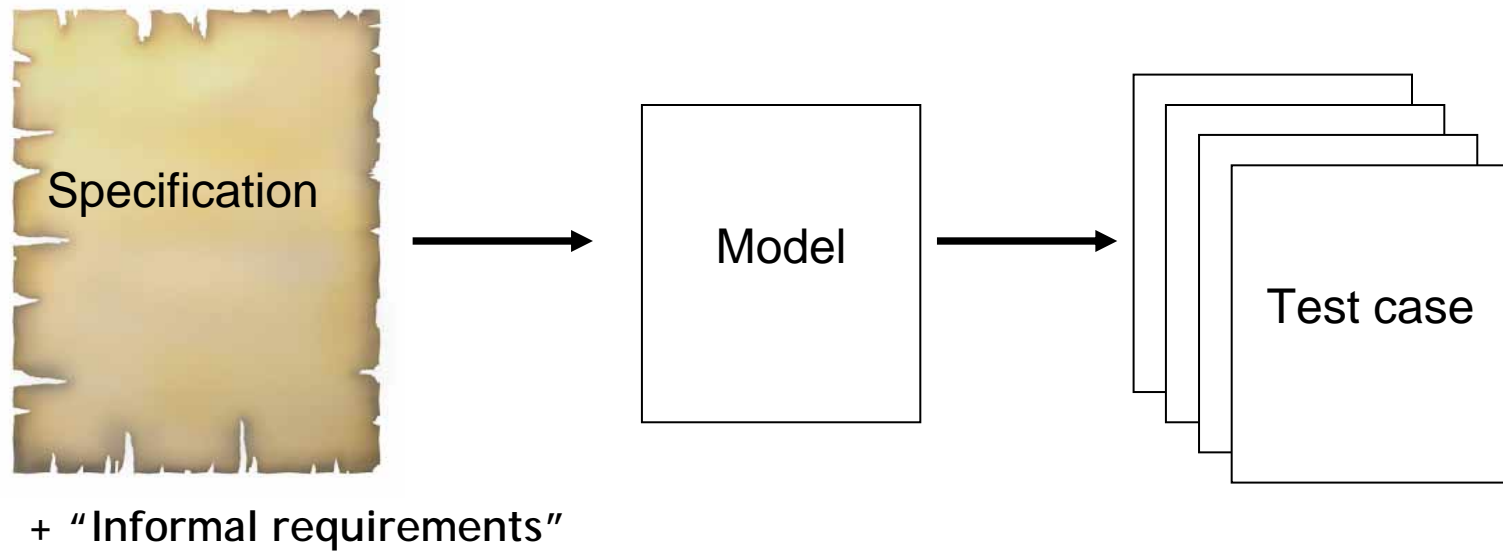


Trend: increasing complexity

Increasingly complex requirements on interaction between systems

Difficult to achieve high coverage with hand-crafted test cases

Model-based testing (MBT)



Steps of model-based testing

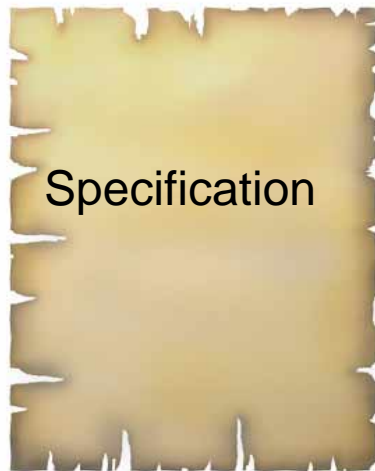
- ❑ Build an abstract model of system
- ❑ Validate the model
- ❑ Generate and execute test cases
- ❑ Assigning pass/fail verdict
- ❑ Analyzing the execution result.

Steps of model-based testing - automation

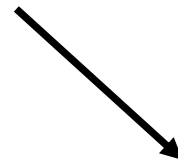
Can be automated?

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

Model-based testing - with QuickCheck



```
-module(imap_eqc).  
-behaviour(eq_c_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);
```

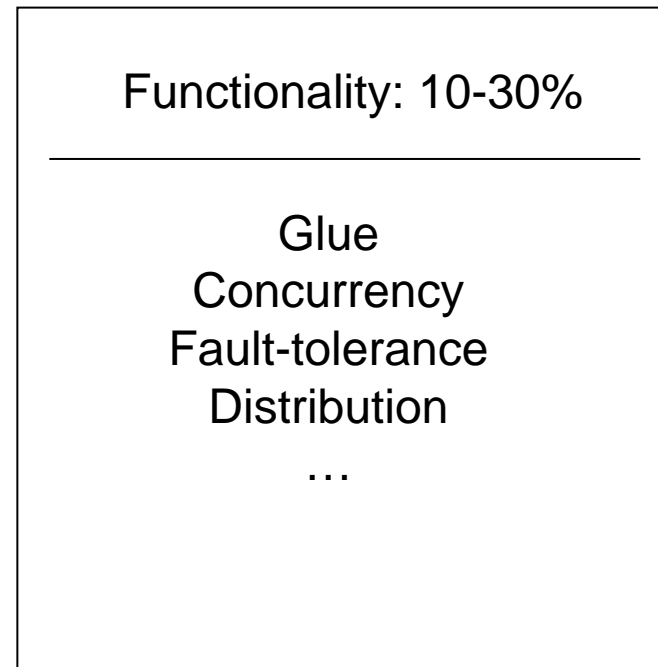


```
eqc:sample( ... )  
eqc:quickcheck( ... )
```



Why is MBT feasible?

Only small part of system-code typically concerned with functionality



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-based testing must be **well** integrated in the test process

Model-based testing in industry

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

Model-based testing

Research approach

Results

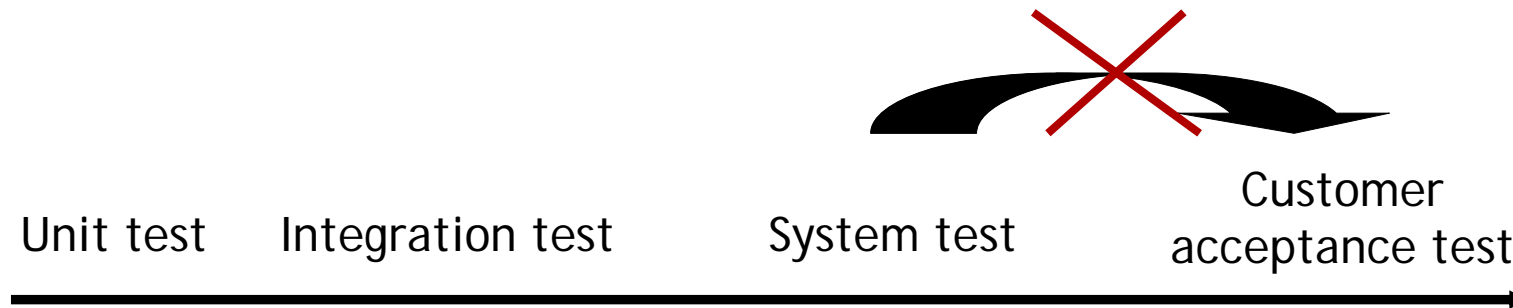
Challenges and
recommendations

Conclusions

Project: Messaging Gateway

Study executed as a process improvement initiative

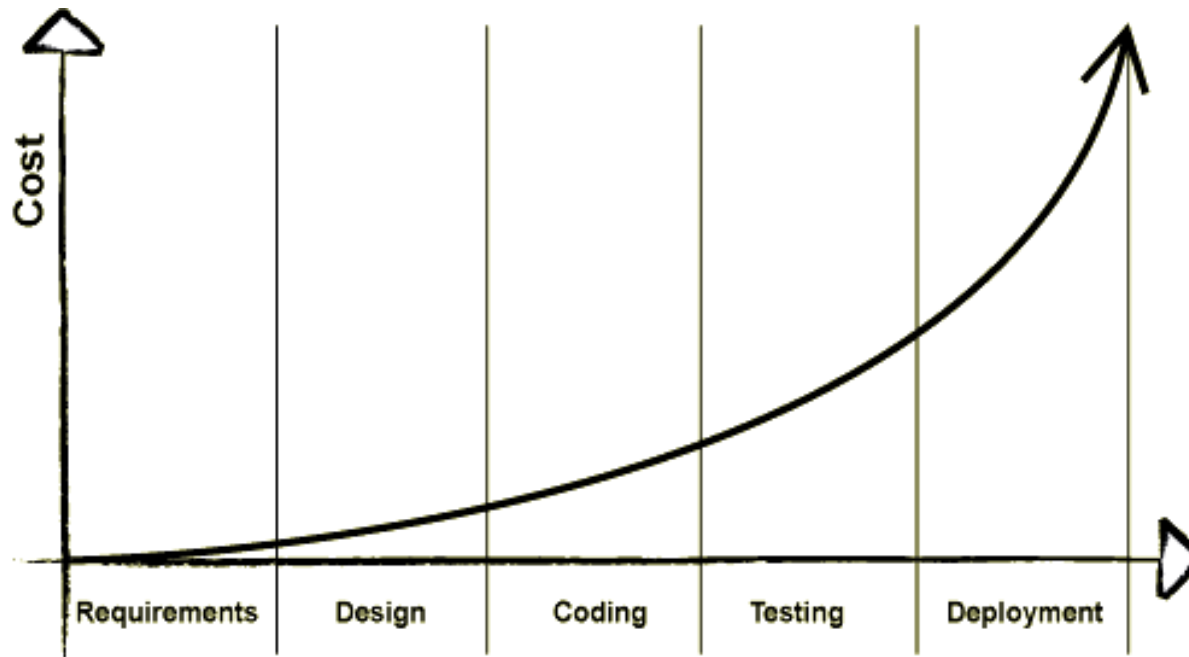
Goal: reduce fault-slip-through to acceptance testing



Costs of faults-slip-through

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

Boehm curve

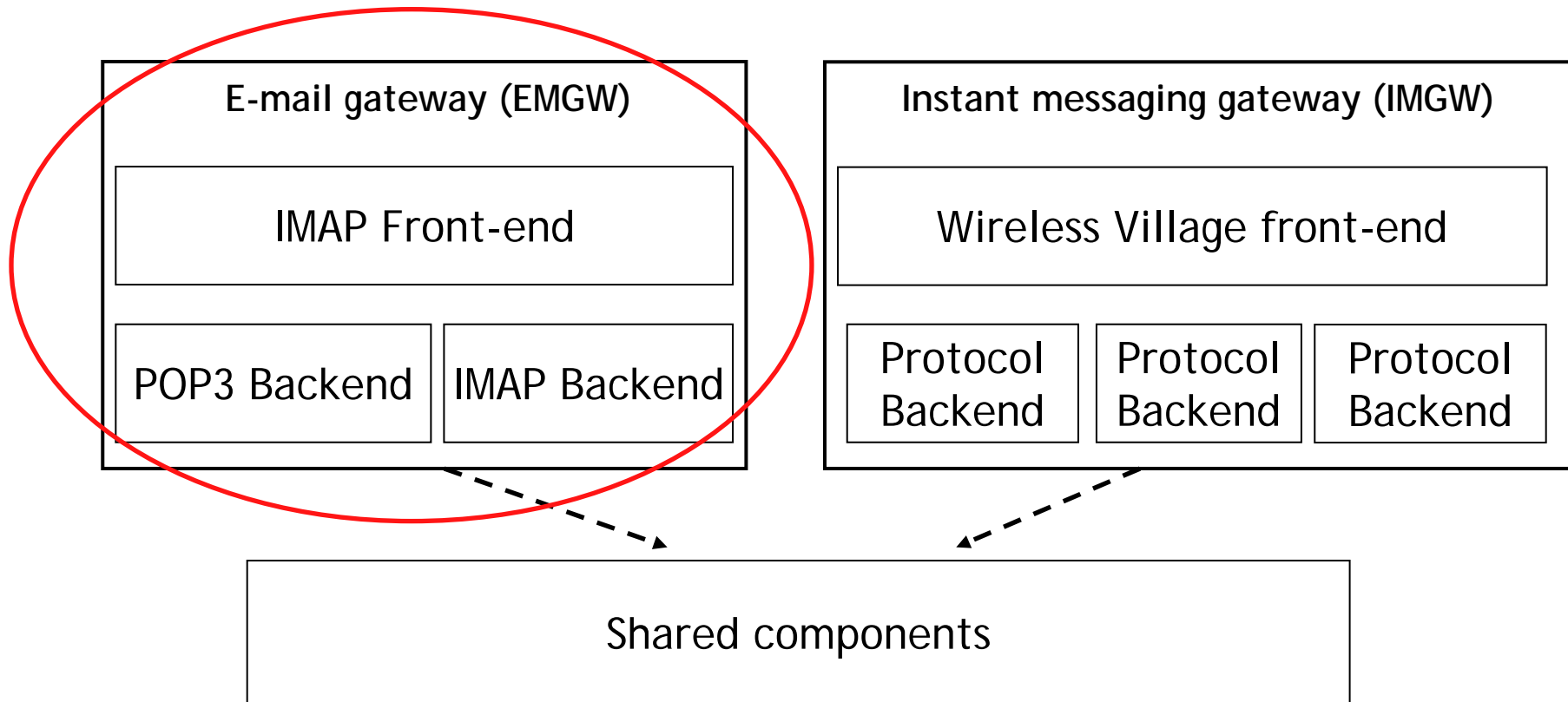


* Boehm, Software Engineering Economics, Prentice Hall PTR, 1981

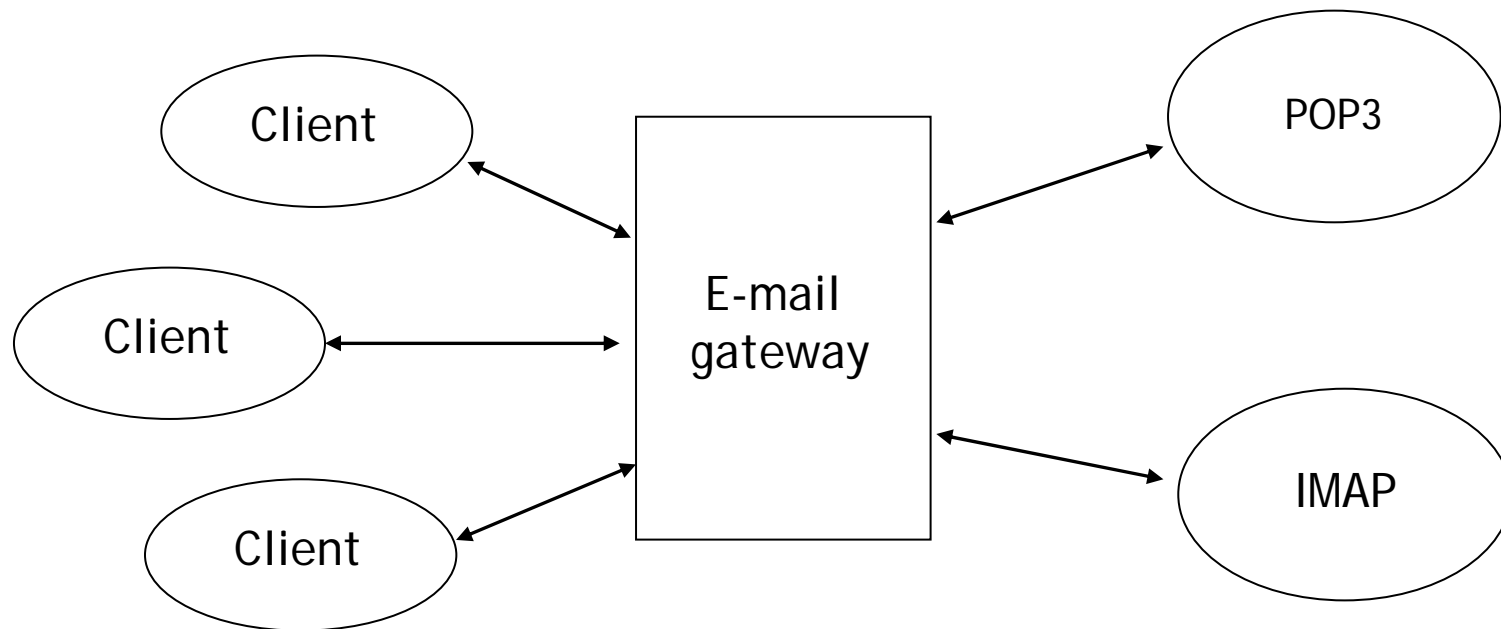
Central question

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

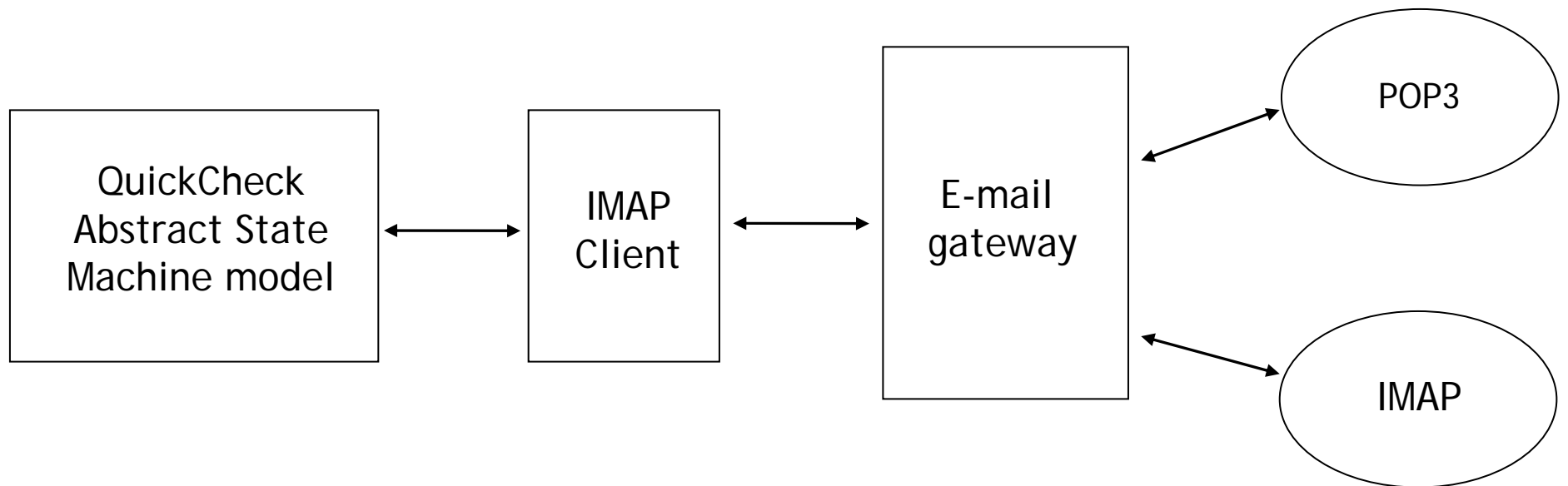
Messaging gateway system



Test approach



Test approach

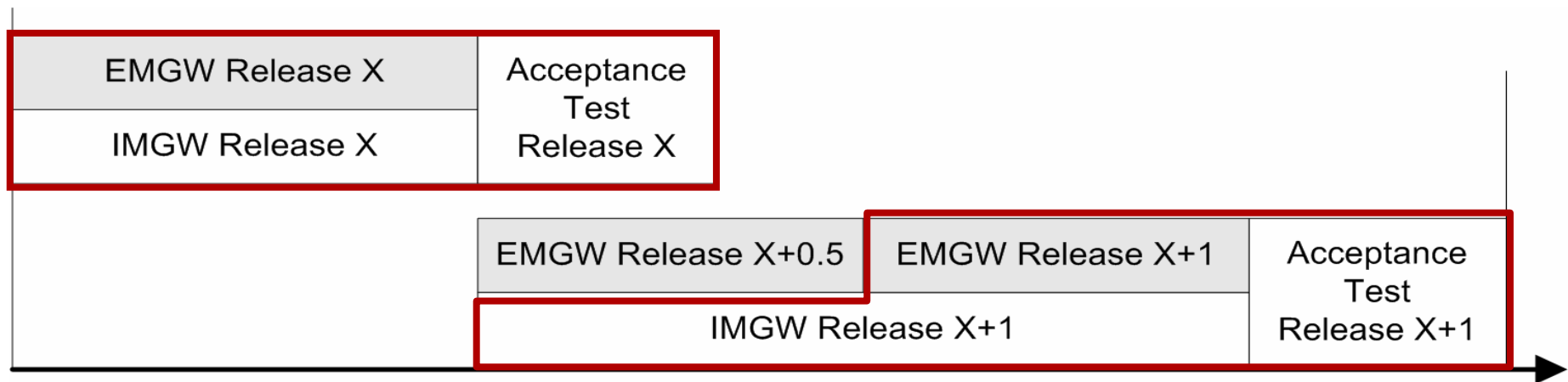


Hypothesis: Model-based will result in lower fault-slip through from system testing to customer acceptance testing

Study timeline

Three studied releases:

- Release X
- Release X+0.5
- Release X+1

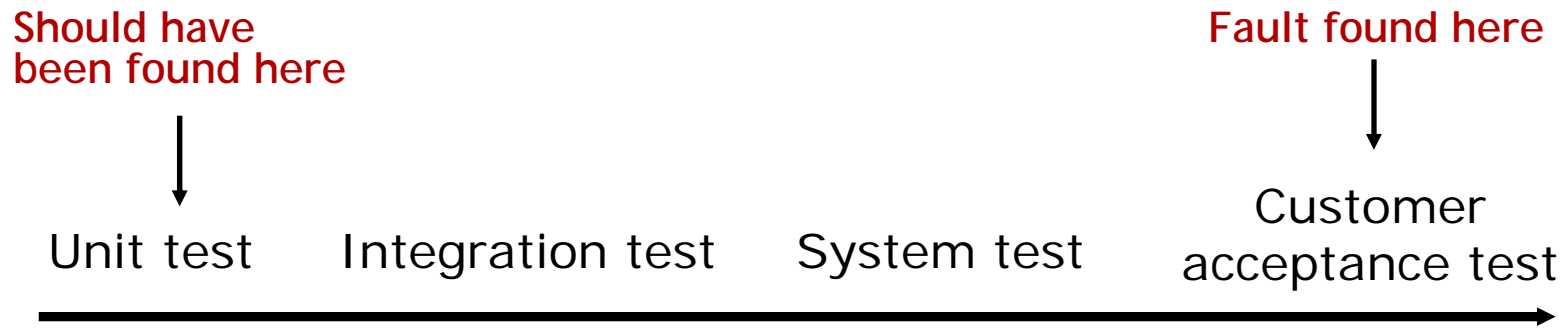


Model-based testing in iterative development

For each iteration:

- Select requirements
- Model requirements
- Validate and execute

Fault-slip-through measurement



Results

Model-based testing

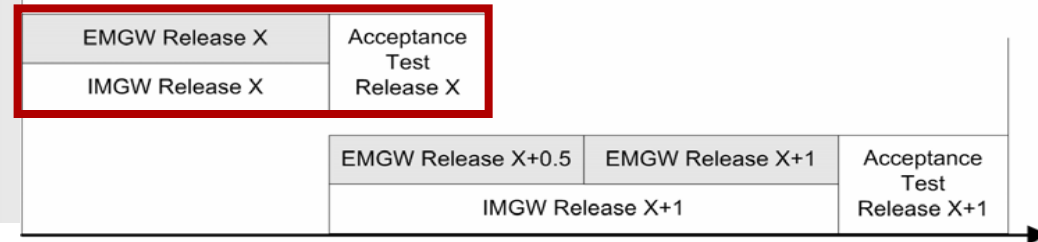
Research approach

Results

Challenges and
recommendations

Conclusions

Results - Release X



EMGW

Found/Belonging	System Test	Acceptance test
Unit Test	1 (0)	1
Integration Test	1 (1)	0
External Integration Test	0 (0)	0
System Test	9 (6)	3
Acceptance Test	0 (0)	2
Total found/test level	11 (7)	6

IMGW

Found/Belonging	System Test	Acceptance test
Unit Test	6	4
Integration Test	0	0
External Integration Test	0	0
System Test	8	10
Acceptance Test	0	4
Total found/test level	14	18

Results - Release X+1



EMGW

Found/Belonging	System Test	Acceptance test
Unit Test	24 (21)	3
Integration Test	1 (1)	0
External Integration Test	0 (0)	0
System Test	39 (26)	4
Acceptance Test	0 (0)	10
Total found/test level	64 (48)	7

IMGW

Found/Belonging	System Test	Acceptance test
Unit Test	1	4
Integration Test	1	0
External Integration Test	0	1
System Test	12	7
Acceptance Test	0	2
Total found/test level	14	12

Phase Input Quality

Phase Input Quality (PIQ) =

Should have been found on earlier level

Total found on level

Release/FST	X	X+0.5	X+1
EMGW FST to ST	18%	24%	39 %
EMGW FST to AT	67%	–	41%
IMGW FST to ST	43%	–	14%
IMGW FST to AT	78%	–	86%

Challenges and recommendations

(As usual,) there is no silver bullet

Model-based testing

Research approach

Results

Challenges and
recommendations

Conclusions

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

Model-based testing

Research approach

Results

Challenges and
recommendations

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