



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



Model-Based Testing in Practice

Alexander Pretschner

ForTIA Industry Day 2005 @ Formal Methods, Newcastle, July 20th, 2005

Agenda

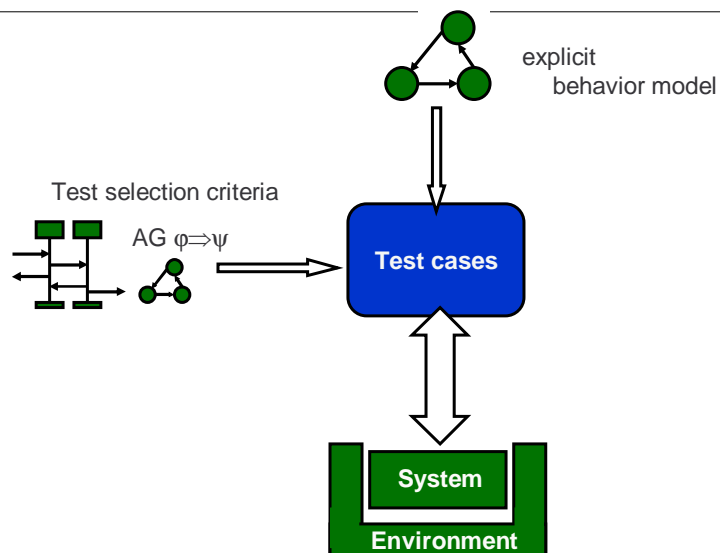
- Model-Based Testing
- Assumptions and Evidence
- Case Study
- Conclusions

Agenda

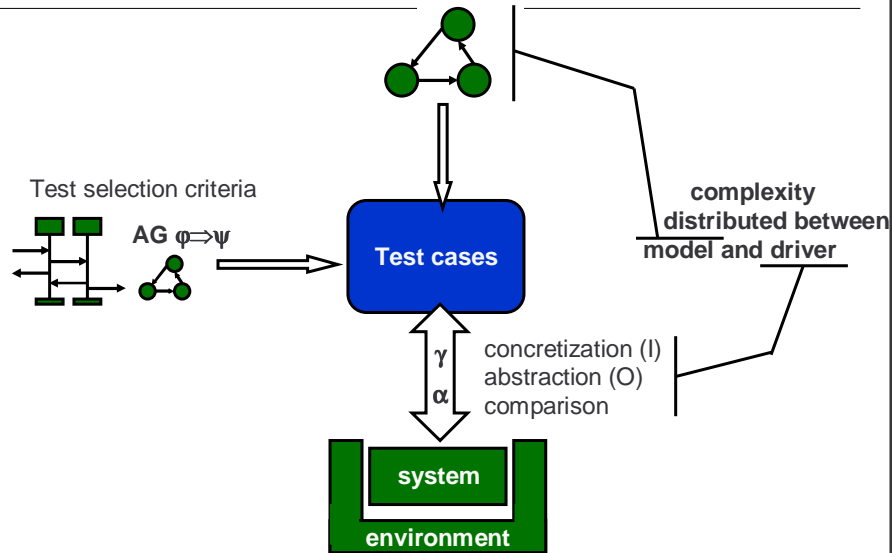
- Model-Based Testing
- Assumptions and Evidence
- Case Study
- Conclusions

Models increase the number of detected errors.
Automatic test case generation has to prove it's beneficiary.
Instead, study building and maintenance of models
as well as domain-specific definitions of "good" test cases.

Model-Based Testing



Levels of Abstraction



Model-Based Testing in Practice, I-Day @ Formal Methods, July 20th, 2005, Alexander Pretschner

5

Assumptions

- Effectiveness and cost effectiveness
 - Models help with getting requirements/specs straight
 - Test suite vs. model: creation and maintenance
- Existence of adequate level of abstraction
 - Abstraction and precision
 - Easy model validation and maintenance
 - Distribution of complexity
- Reuse
 - Simple changes in the model
 - Adaptor and environment models/TC specifications

Model-Based Testing in Practice, I-Day @ Formal Methods, July 20th, 2005, Alexander Pretschner

6

Evidence: (Cost) Effectiveness

- “Model-Based Testing does find errors”
- Different/more errors in SUT?
 - Dalal et al. '99, Farchi et al. '02, Pretschner et al. '05
 - No precise description of reference
 - Errors in model or specs
- Cost Effectiveness
 - Farchi et al. '02, Bernard et al. '04
 - “building tests took less time”
- Ongoing dispute on testing vs. reviews
- In sum: hard to admit, but very little evidence!

Case Study

The use of models leads to a sixfold increase in detected requirements errors and to a 30% increase in detected programming errors.

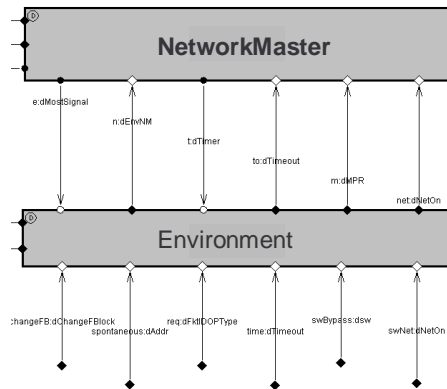
Automation does not increase test effectiveness.

Model and implementation coverages correlate moderately.

- Study setup
- Test cases
- Error detection

System and model

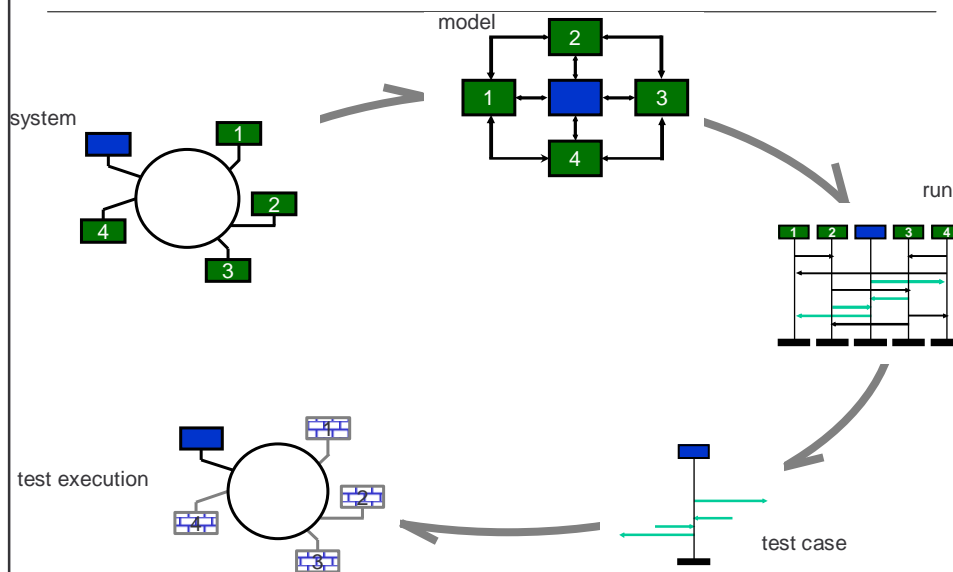
- MOST network controller
 - Infotainment in cars
 - Assignment of logical addresses, mapping of logical functions
- Basis: informal MSCs
- SUT: SW-in-the-loop simul.
- Model: AutoFocus
 - hierarchic components, EFSMs, functional programs



Model-Based Testing in Practice, I-Day @ Formal Methods, July 20th, 2005, Alexander Pretschner

9

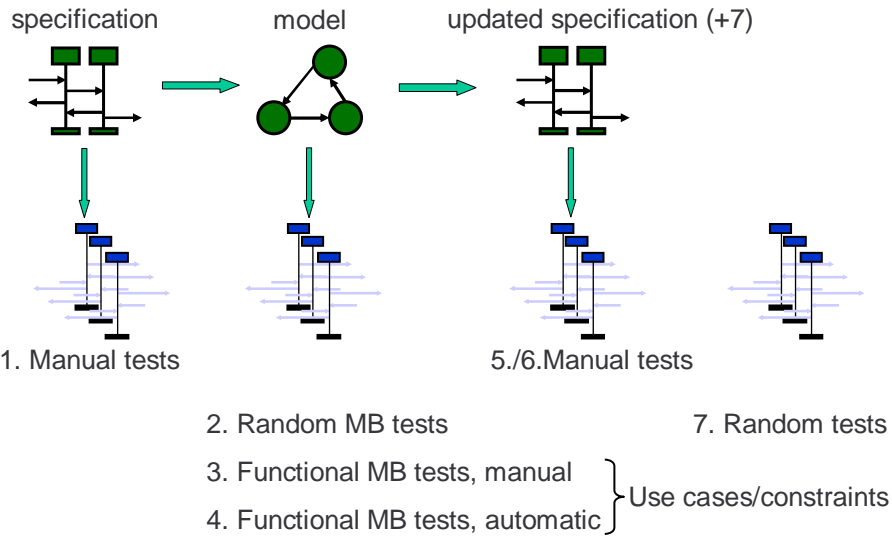
Model-based test generation and execution



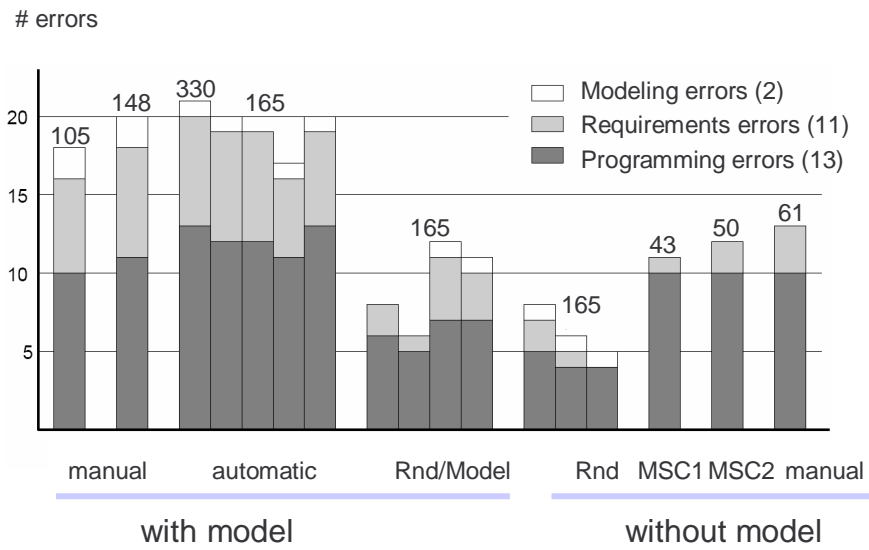
Model-Based Testing in Practice, I-Day @ Formal Methods, July 20th, 2005, Alexander Pretschner

10

Test suites



Errors



Discussion

- Use of models effective: 30+11 spec problems found
- Programming errors found regardless of models
- Automation does not increase effectiveness
 - Possible reasons: TC specs, generation technology, system class, model, level of maturity
 - Possible generalization: event-discrete reactive systems with few ordered data types; ~ same complexity
 - Different errors detected:
Combination with manual tests desirable
 - Efficiency (cost, severity) not regarded here
 - Here: rather small sets of rather short tests

Summary

- “Complete” behavior model encodes intended behavior
 - Intrinsic value: check requirements!
 - Abstraction vs. Precision
- Test model likely too abstract for code generation
- OEM and suppliers
- Fully automatic TC generation for syntactic criteria only
- Models helpful, not automation, coverage unclear

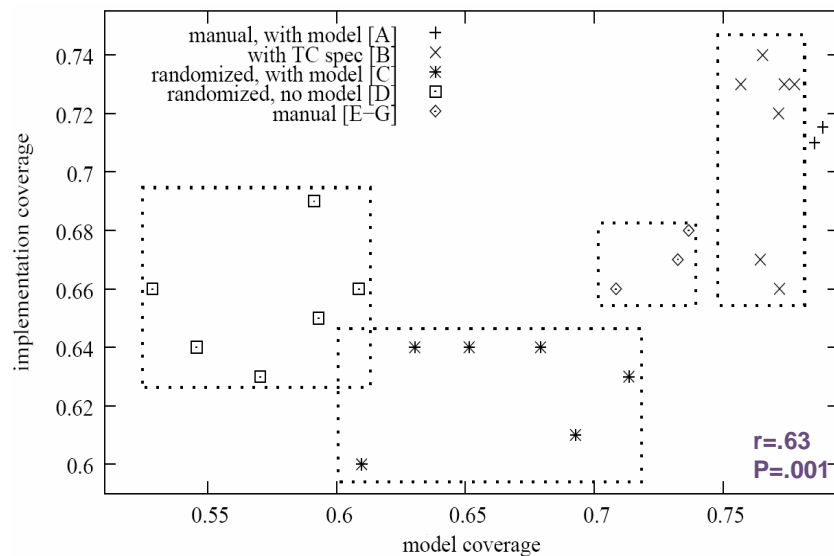
Outlook

- Generalization?
 - More empirical studies necessary, in particular in the context of business information systems
 - Cost, severity
- Dedicated modeling language for TC specs
 - Environment models; models of SUT
 - Based on requirements and based on historic faults
- Focus on abstractions, modeling heuristics & languages, tool support for (incremental) modeling; not generation!

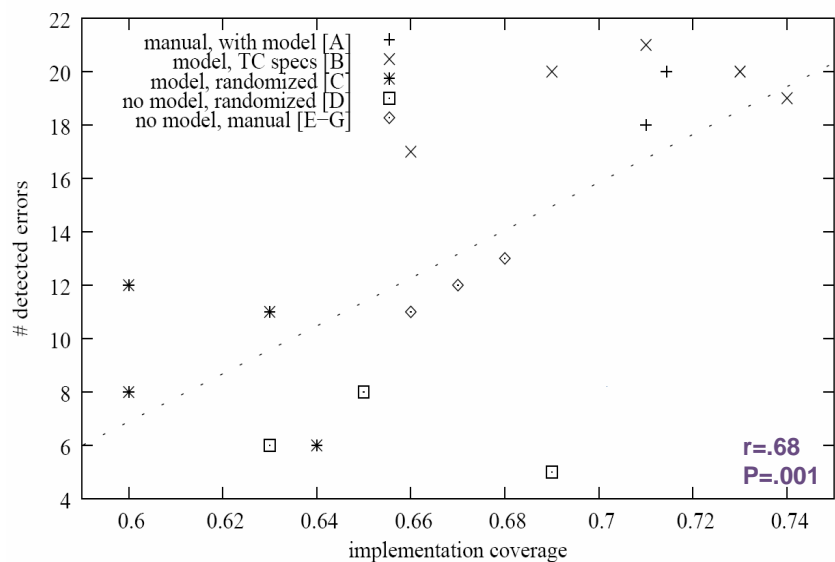
Coverage

- Objective
 - Coverage vs. errors
 - Consider using coverage a-priori
 - Model coverage vs. implementation coverage
 - Most studies on *implementation* coverage
- Condition/decision coverage
 - Each atomic condition and the decision take both possible outcomes
 - $A \wedge B$: (TT, FF) \rightarrow 100%, (TF, FT) \rightarrow 83%, (TT, FT) \rightarrow 83%
- Implementation: C code; model: generated Java code

Coverages



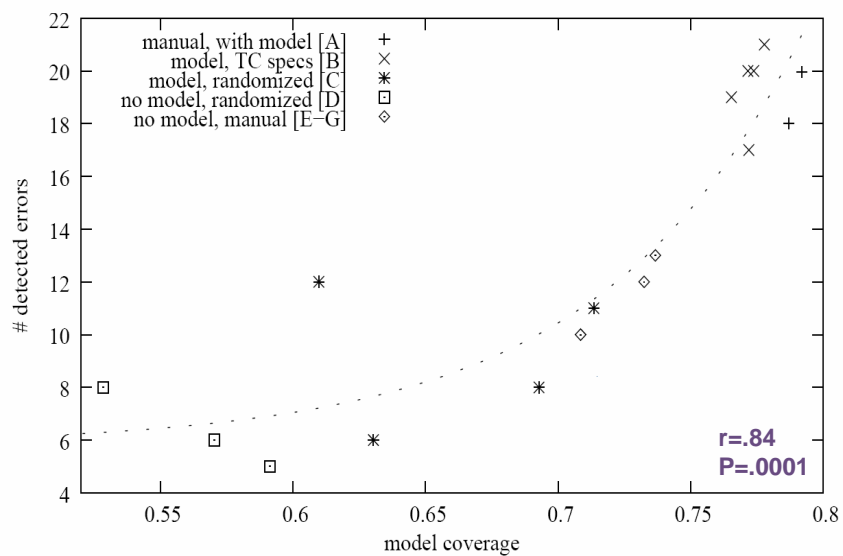
Implementation coverage and errors



Model-Based Testing in Practice, I-Day @ Formal Methods, July 20th, 2005, Alexander Pretschner

19

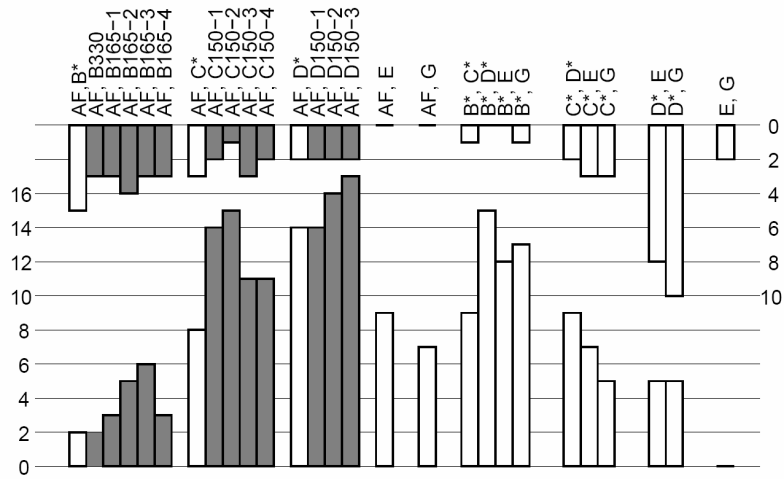
Model coverage and errors



Model-Based Testing in Practice, I-Day @ Formal Methods, July 20th, 2005, Alexander Pretschner

20

Different Errors



Model-Based Testing in Practice, I-Day @ Formal Methods, July 20th, 2005, Alexander Pretschner

21

“Complexity”

- 17 components
100 channels, 138 ports,
12 EFSMs, 16 control states, 16 local variables,
104 transitions,
34 data types, 80 constructors, 141 functions
- 12,300 lines of executable C code

Model-Based Testing in Practice, I-Day @ Formal Methods, July 20th, 2005, Alexander Pretschner

22