

UML-Based Statistical Test Case Generation

Matthias Riebisch,
Ilka Philippow, Marco Götze
Ilmenau Technical University; Germany

<http://www.theoinf.tu-ilmenau.de/~pld>

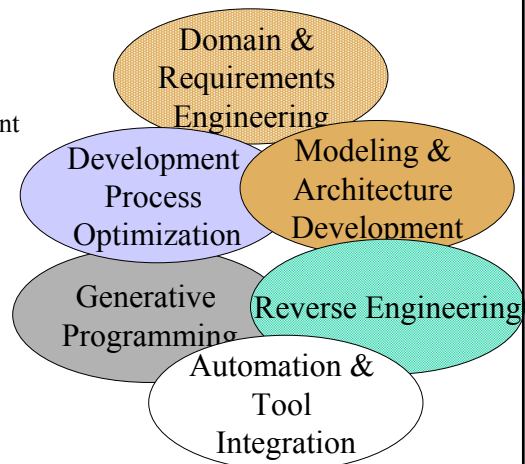


NetObjectDays'02, Erfurt, October 10, 2002

ALEXANDRIA Research Project



- Reusability at Higher Level
- Integration of Domain Engineering with Architecture Development and Generative Programming
- Generation of Customized Solutions
- Incorporation of Existing Systems and Components
- **Generation of Tests**
- Verification of Results in Industrial Projects



Outline





- Introduction: Model-Based Statistical Testing
- The Approach
- Scenario for Practical Use
- UsageTester – a Proof-of-Concept Tool
- Next Steps



Introduction: Testing Strategies



Type:

- Black Box *-functional-* testing vs. 
- White Box *-structural-* testing 

Goal:

- Tests for error detection *-coverage-* vs.
- Statistical testing *-reliability-*



Statistical Usage Testing



- Check for reliability at system level

Approach:

- Different probability of execution of modules
- Testing effort dispensed due to probability
- Execution probability derived from usage probability

➤ **Usage Model**



Motivation & Aim



- Incremental development needs short cycles
- Early tests for process maturity
 - Feedback for developer and for project management
 - Early failure detection reduces re-work
- Time and costs reduction
- Test case quality improvement
- Encouragement for model quality



Outline



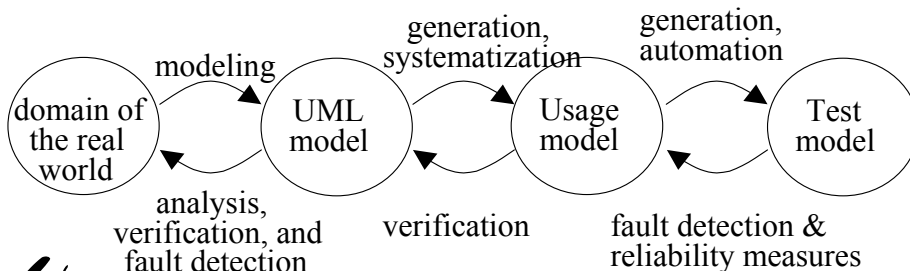
- ✓ Introduction: Model-Based Statistical Testing
- The Approach
- Scenario for Practical Use
- UsageTester – a Proof-of-Concept Tool
- Next Steps



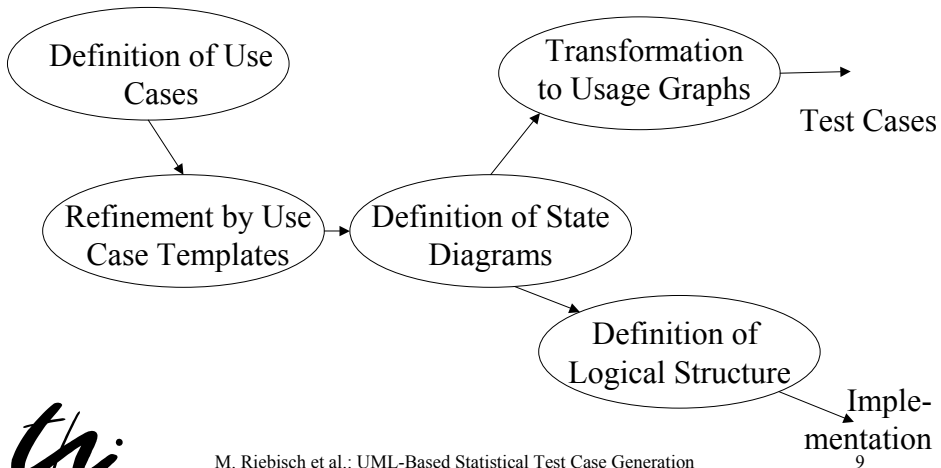
Model-Based Statistical Testing



- Black-box, system-level testing
- Statistical testing for reliability
- Test case derivation from specification



Model Refinement Steps Detailed



M. Riebisch et al.: UML-Based Statistical Test Case Generation

9

Use Case Templates



- Refining use case diagrams
- Semi-formal description
- State & transition information covered
- Not (yet) defined by OMG standard

Lend a Book



Name	Lend a book
Goal	Lend a book to a user of this library for a limited time.
Actor	Librarian
Preconditions	Library system is running. The user is valid to the library.
Postconditions	The book was lent to the user. A due date for return is defined.
Invariants	None
Main Success scenario	1) The Librarian opens the user selection dialog with selected user. 2) He opens the lending dialog for selecting a book, a return date is shown. 3) He
Variations	None
Exceptions	2a) User has to pay a fee at a time. 2a1) Message: no lending allowed 2a2) Successful - return to lending dialog. 2a3) Not successful - return to user selection
Included use cases	Display books lent by a user. Collect payment from a user.

see example ...



M. Riebisch et al.: UML-Based Statistical Test Case Generation

10

Name	Lend a book
Goal	Lend a book to a user of this library for a limited time.
Actors	Librarian
Preconditions	Library system is running. The user is valid to the library
Postconditions	The book was lent to the user. A due date for return is defined.
Invariants	None
Main Success scenario	1) The librarian opens the user selection dialog with selected user. 2) He opens the lending dialog for selecting a book, a return date is shown. 3) He confirms the operation. 4) The book is marked as lent, the <i>lent-book-list</i> of the user is updated.
Variations	None
Extensions	2a) User has to pay a fee or a fine. 2a1) Message: no lending allowed 2a2) Successful - return to lending dialog. 2a3) Not successful - return to new user selection 2b) The book of interest is already lent or is for reference only. 2b1) Message - return to book selection 3a) Cancelled by librarian 3a1) Return to book selection.
Included use cases	Display books lent by a user; Collect payment from a user.

State Diagrams



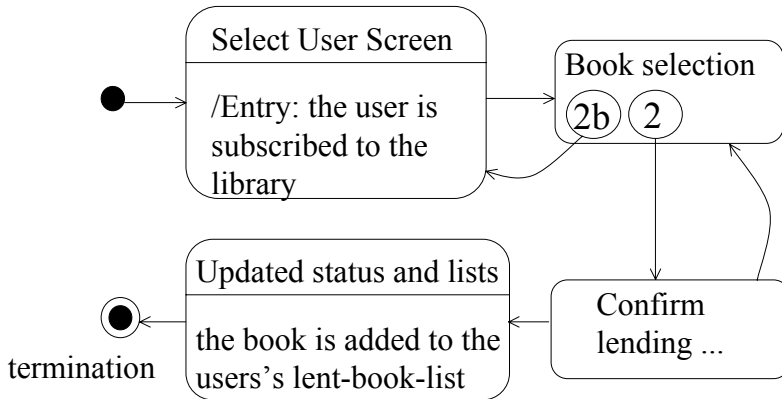
ALEXANDRIA

- Defined by UML
- Intended for *object* behavior
- Hierarchy for mastering complexity

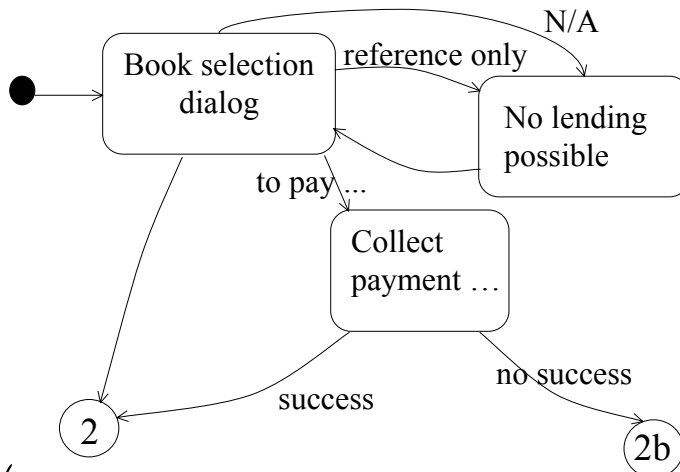
- Showing system states and transitions according to refined Use Cases
- Transformable to Usage Graphs

see example ...

State Diagram Example



State *Book Selection* Refined



Usage Graphs



- Special form of Automata graph
- Transitions labeled
 - with user actions for triggered transitions
 - with epsilon ϵ for transitions performed automatically
- References to state diagram and use cases via state label prefix



Usage Models

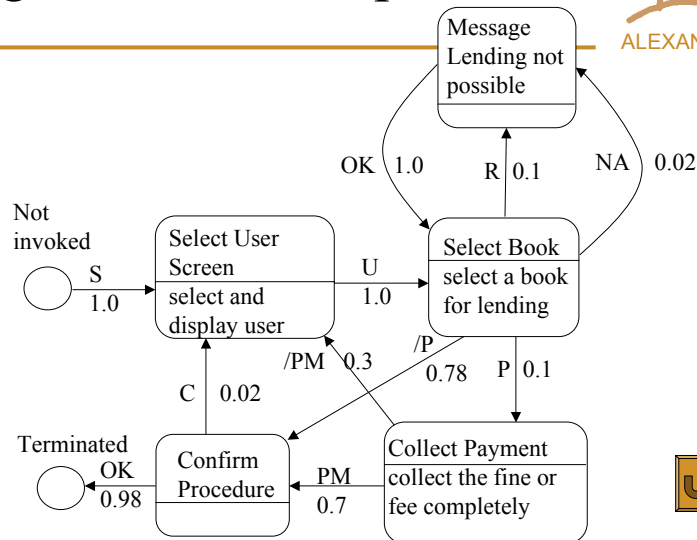


- State diagrams extended by usage information
- Probabilities of user actions derived by requirements or customer data
- Probabilities annotated to outbound transitions of each state
- ϵ transitions: probability of 1.0

see example ...



Usage Model Example



tri

M. Riebisch et al.: UML-Based Statistical Test Case Generation

17

Test Cases



- Usually: test cases programmed as scripts or built by teach-in
- Now: Input Values + Expected Results derived from the models
- Input for Test Automation tools

tri

M. Riebisch et al.: UML-Based Statistical Test Case Generation

18

Outline



- ✓ Introduction: Model-Based Statistical Testing
- ✓ The Approach
 - Scenario for Practical Use
 - UsageTester – a Proof-of-Concept Tool
 - Next Steps



Scenario for Practical Use



- Development of test cases independently from programming
- Test cases based on requirements to system behavior
- Test cases prior to design
- Test tool available for developer – early feedback with low effort



UsageTester – a Proof-of-Concept Tool



Goal: verification of the approach in practical application

- Supported: model refinement & conversion
- Manual activities: model completion
- Automated: test case generation
- XML interface for model input and test case output
- Portable implementation with Java, Swing, DOM4J

tr

UsageTester – Snapshot



The screenshot shows a window titled "WithdrawMoney (use case)". The main area displays a use case diagram with the following elements:

- Instance: (NONE)
- Use case WithdrawMoney: Withdrawal of money from the customer's account.
- Main scenario: Customer withdraws an amount of money entered by him/her.
- #0: Call of use case CheckPIN: The user authenticates himself/herself via his/her PIN.
- #0: scenario for terminal at #0.0.0.0.0: Customer failed to enter valid PIN.
- #0 Terminal: Customer fails to authenticate himself/herself.
- #1: Customer is prompted to and enters a valid amount of money to withdraw. <- EnterValidAmount
- #0: Variation: Customer decides to abort. <- Cancel
- #0 Terminal: Abort on customer's command.
- #1: Extension: Customer enters invalid amount of money. <- EnterInvalidAmount
- #0: Transition to #1: Display error message and return to prompt.
- #2 Terminal: Withdrawal and ejection of entered amount of money.

Below the diagram, there are text fields for:

- Brief:** Withdrawal of money from the customer's account.
- Goal:** This use case describes how a customer withdraws money from his/her account.
- Pre-condition:** There is a debit card inserted in the ATM machine, and the user is known to the system to have entered a valid PIN.
- Invariants:** (empty field)

At the bottom, there are buttons for Save, Apply, Rename..., and Discard.

tr

Conclusion



- Test cases derived from requirements specification and usage behavior
- Tool support; no automation intended
- Model exploitation encourages modeling effort
- Providing test cases early supports programmers



Next Steps



- Next 2 years: project funded by DFG
- Cooperation with UCF Orlando, Florida
- Application of Sequence Diagrams for deriving State Diagrams
- Hierarchical models for scalability
- How to acquire usage information
- How to derive expected test result date
- How to add user profiles and product variants
- Experiences from practical usage





Questions and Comments

More information about ALEXANDRIA:

<http://www.theoinf.tu-ilmenau.de/~pld/>

thi

