From Keywords to Model-Based Test Automation

Tuula Pääkkönen, Nokia & Mika Katara, Tampere Univ. of Technology, Finland
From Keywords to Model-Based Test Automation

Tuula Pääkkönen
Global Concept Owner, Test Automation
Nokia Technology Platforms
Finland

Mika Katara
Dr. Tech., Adjunct Professor (Docent)
Tampere University of Technology, Institute of Software Systems
Finland
Outlook

1. Introduction to the anticipated benefits of model-based software testing
2. Combining keywords and model-based testing in the Symbian S60 GUI testing domain
3. Introduction to the associated open-source tool set and experiences from industrial case studies
4. Conclusions
Model-Based Testing (MBT)

- MBT automates not only the execution of tests but also their generation.
- It is not a new idea to use formal models, for example finite state machines, to generate black box tests.
  - The topic has been widely studied in the academia over the last decades.
- Industrial deployment has been successful in certain domains (see, for instance [1,2]).
However, large scale deployment of MBT has turned out to be very difficult (see, for instance [3])

- Tools can be hard to use
- Managerial problems
- Re-organization of work etc.
- Adjusting the testing processes (agile etc.)
- Modeling is not easy, not everybody can do it
- Model modularity and management
  - (100 states vs. $100^5$ states)
Benefits of MBT

• Conventional test automation is not particularly good in finding new bugs
  • Mainly used for regression testing to make sure nothing already tested has been broken while adding new features / fixing bugs / refactoring
• MBT can generate new tests in every test run
  • Even a single loop in the model (state machine) makes it possible to generate an infinite number of different tests (based on the number of loop iterations)
• Much higher coverage can be achieved than with conventional techniques
  • Note: structural coverage based on model elements does not necessarily correlate with the code coverage
Benefits…

• Maintenance of conventional test scripts, especially in GUI testing, can be very time consuming
• Since in MBT test are generated automatically, it is sufficient to update the models and generate the tests again
• The larger the test suite, more anticipated savings compared to the maintenance of scripted tests
Generations of (System Level) Test Automation

Capture Replay, Spaghetti Scripts

Structured Test Scripts

Data-Driven Scripts

Keywords, Action words

See for instance, *Software Test Automation: Effective use of test execution tools*  
By Mark Fewster and Dorothy Graham, Addison Wesley, 1999.
Generations…

Manual testing

Scripted test automation

Model-based testing
From Keywords to Model-Based Automation

• A state machine is a collection of states and transitions
• Transitions describe how the state changes occur
• One of the states is marked as the initial state

• In his article [4], Hans Buwalda suggested using spreadsheets for combining state machines with action/keywords
• However, we prefer to present the models more visually and modularizing large models using parallel composition of smaller component models
On-line vs. Off-line Testing

Challenges in Industrial Adaptation

- Requires commitment to MBT, at least in testing / test automation organization
  - See Abrahamsson [5] for commitment development in process improvement activities
- Increased need for communication at lower levels, when test models are built to be as comprehensive as possible
- The most probable scenario is that the testing organization will have new roles and tasks incorporated in the traditional software testing activities
- Model creation requires effort
  - However, [6] reports that MBT-techniques can be learned within a few days
Challenges…

• The main challenge is how to take the step from pilots to deployment:
  • This has a vicious circle: someone should be the first to show-case the results, but on the other hand, the early innovators should get the benefits relatively fast, which may turn out to be difficult to achieve
  • In addition, scalability of the chosen MBT solution to real-life situations needs to be ensured [7]
• The role of design models, if such exist, is still unclear in MBT
  • Adopting MBT could trigger a change in the whole software development towards model driven approach
Domain-Specific Approach – S60 GUI Testing

- Our domain: GUI testing of Symbian S60 smart phones
- Our solution is based on a domain-specific test modeling language [8] instead of UML or any other generic language
  - This enables higher level of automation and usability
- “MBT-thinking” helps in partitioning the target of testing into manageable component models
- Testing complex interactions between different applications and systems
- Managing complexities within functional, regression testing goals in a *product family* context
- Performance aspects – monitoring longer sequences – being able to stress the system under test for longer periods of time
• *Action words* describe user’s actions at a high level of abstraction
  • Send an SMS, answer a call, add a new contact, browse the calendar etc.
• An action word is translated to a sequence of *keywords* (key strokes) for menu navigation, text inputting etc.
• *Action machines* containing action words are composed with *refinement machines* containing keywords
• The resulting composite model is input to the tools executing the model, i.e. generating sequences of keywords
• To avoid *state space explosion*, the test generation has been implemented using an on-the-fly algorithm
Additional Benefits

• Testing the interactions between the different smart phone applications is seen as additional benefit compared to existing methods

• Test data universe – being able to include more versatile test data in the test generation/execution phase

• Decisive test ending criteria based on test coverage
  • With the help of the test model the coverage can be visualized and utilized in the test generation/execution phase
Example Test Models

S60 Camera application, action word model

Illustration: Antti Kervinen/TUT
Example…

S60 Camera application, keyword model

Illustration: Antti Kervinen/TUT
TEMA tool set

Illustration: Mika Maunumaa/TUT
Experiences from Industrial Case Studies

• A lot of problems with building tool adaptation
  • Certain keyword implementations have turned out to be very problematic
  • Due to this, test execution has been delayed
• In the first case study, the SUT was a S60 phone already released to the market and the study concentrated on the built-in applications
  • Some defects were found during test modeling
    • The most severe of these caused one application to crash
Experiences...

- Due to the aforementioned technical difficulties, limited test execution became possible only in the second case study.
- This time the SUT was a third-party S60 application:
  - One defect was found in actual test execution and a few other issues while developing the test models.
  - None of these defects can be considered serious.
- In both studies, the SUT has been mature and very well tested before the case studies.
- Most of the defects found have been concurrency related, which supports our earlier findings [9].
Experiences…

• Creating the test models for the basic S60 applications has been easier than anticipated
  • Dozens of component models have been created for S60 built-in applications, such as Camera, Calendar, Contacts, Messaging etc.
• Even domain-specific models can be hard to understand
• In the third case study, a web interface is to be used for specifying test purposes (coverage requirements) that drive the test generation [10]
  • This allows hiding the complexity of the solution (test models and generation algorithms) from the testers
Conclusions

• Deployment of MBT might be hard… but the benefits are really promising
• Upgrading from keyword automation to MBT automation is possible
• Building a suitable tool adapter may turn out to be very difficult
• Test modeling reveals errors effectively
• A domain-specific approach needs custom tools, but promises higher level of automation and usability than generic approaches (UML etc.)
• TEMA tools are available under the MIT Open Source license, see http://practise.cs.tut.fi
References


References...


Thank you for your attention

Contact:

Tuula Pääkkönen
Nokia Technology Platforms
tuula.a.paakkonen@nokia.com

Mika Katara
Tampere Univ. of Tech.
mika.katara@tut.fi

Special thanks to TEMA team at TUT: Antti Kervinen, Mika Maunumaa, Antti Jääskeläinen, Merja Leppänen, Henri Heiskanen, Jukka Mikkonen