Automatic Test Generation for Mobile GUI Applications

Henri Heiskanen Tommi Takala
Department of Software Systems
Tampere University of Technology, Finland
first.lastname@tut.fi

Contents

About Model-Based Testing

- Model-based testing is a testing approach where not only the execution, but also the generation of tests is automated
- Automatic test generation is based on a *test model* describing system behavior at a detailed enough level to enable the automatic test generation.
- Two ways to go about it:
  - Online MBT – Tests are generated and executed simultaneously
  - Offline MBT – Test cases are generated prior to execution

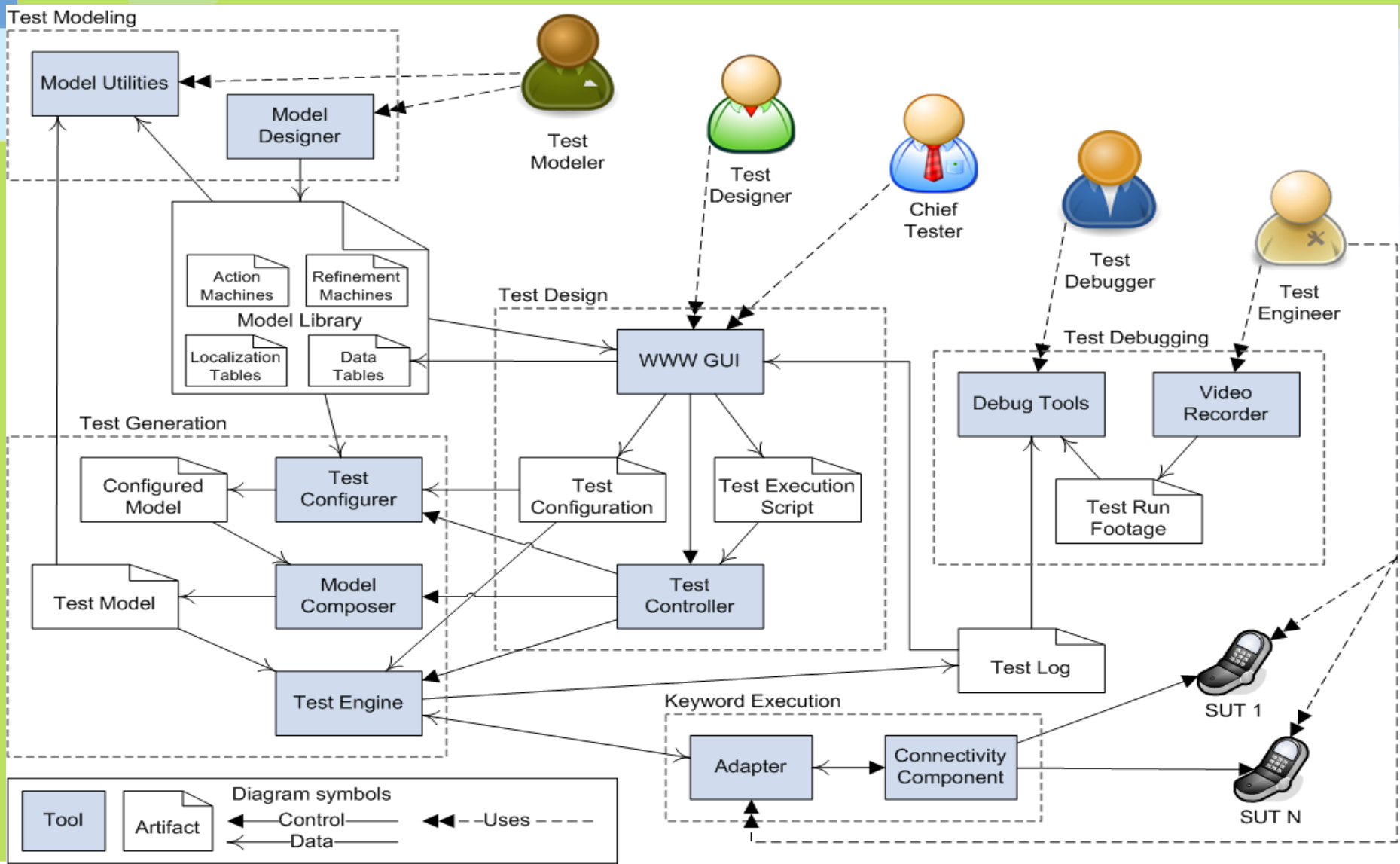
Obstacles and Opportunities for MBT

- Practitioners are willing to try out new tools that might help them
  - Wide variety of open-source testing tools already used
- Practitioners are not willing to invest heavily on modeling or specification in general
- When quality is not a prime consideration, conventional testing methods seem to work reasonably well
- There are areas that are very hard to test using conventional methods (static and linear test cases)
  - Many applications running concurrently and sharing resources may suggest concurrency problems
  - End users who experience application hang-up/crashing problems etc. may post their bad experiences to the Internet…

TEMA Toolset - Hiding Innate MBT Complexity

- Since testers don't want to directly deal with models or test generation algorithms, we have abstracted them out in our web GUI
- TEMA web GUI is testers' interface with the test server, used for designing and managing test configurations, running and tracking actual tests, and managing test model packages
- This all boils down to allowing testers to just choose what they want to test and what physical device they want to run their tests on
- Organizational impact:
  - Need for test design has diminished, only test configurations (that may involve use cases) have to be created
  - Modeling is imperative
    - High-level models can be reused, but domain-specific refinements must be created case by case for each domain

Toolset Architecture



Keywords and Action Words

- Action words describe the user's actions at a high level of abstraction
  - Send an SMS, answer a call, add a new contact etc.
  - Used in high-level models (action machines)
- An action word is translated to a sequence of keywords (keystrokes) for menu navigation, text inputting etc.
  - Some action words can have multiple keyword sequences implementing them
  - Keywords are used in low-level models (refinement machines)

Example Test Models

S60 Camera application, action machine

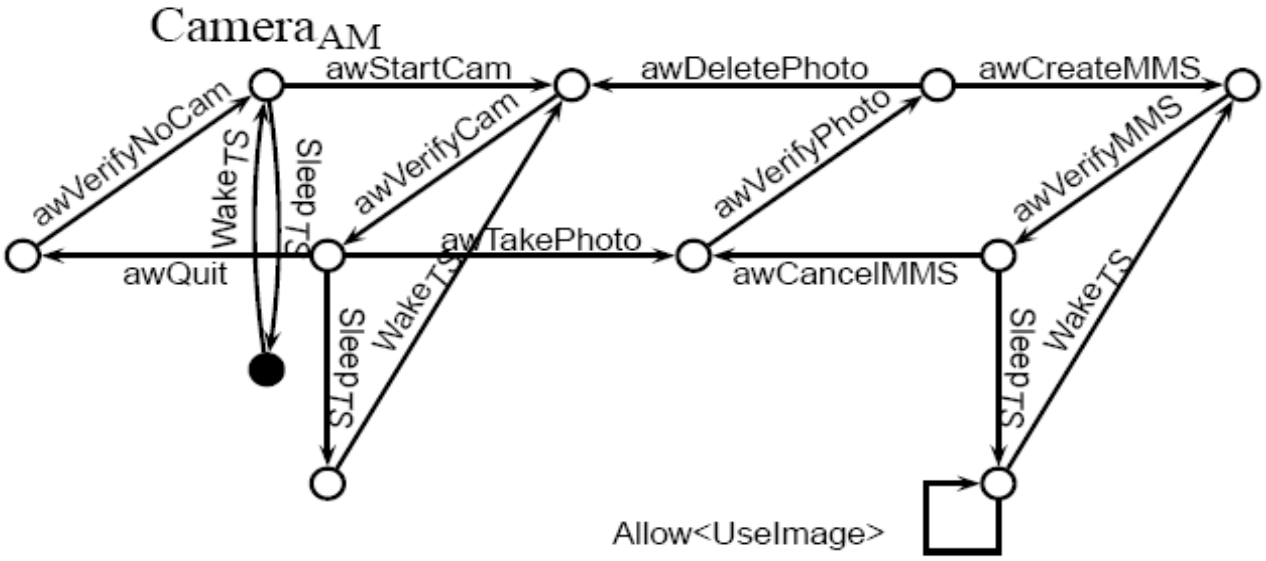


Illustration: Antti Kervinen/TUT

S60 Camera application, refinement machine

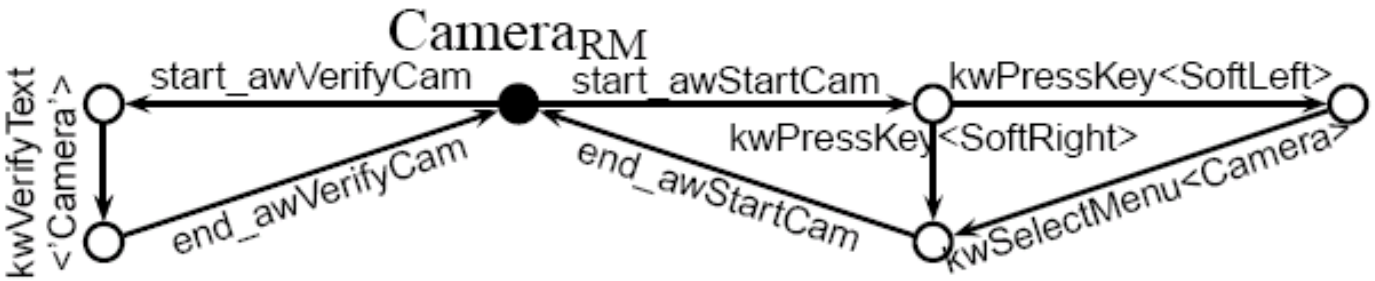


Illustration: Antti Kervinen/TUT

Domains Conquered

- We have been primarily focusing on mobile platforms, but our approach is also suited to desktop applications
- Our model library presently holds models for the following domains:
  - S60
  - Mobile Linux
  - Android
  - Java Swing
- Action machines (high-level models) have been reused for different domains

Test Generation

- The test model is composed on the fly during test execution
- We use various graph search algorithms, such as random and interaction-based algorithms
- The end user is only concerned with web GUI modes and their parameters
  - Use case, bug hunting, basic coverage and smoke testing modes

Conclusions

- Modeling typically uncovers more bugs and quirks than test execution itself
  - Reverse-engineering vs. use of system models
  - Lack of precise enough GUI specifications
    - Agile trend hasn't made matters easier….
- It is possible to find bugs in already well-tested applications
  - Mostly minor or cosmetic, but also serious (system errors, etc.)
  - Otherwise hard-to-find concurrency-related bugs have been found (major upside of our approach)
- A talented student was able to create the first version of the S60 model library in 2 months (+1 month for debugging & maintenance)
- Automatic GUI testing requires mature test automation support from the domain

Case: S60 (Project Starting Point)

- Built-in applications in S60 smartphones, such as Gallery, Music Player, Flash Player, Notes, Voice Recorder, Contacts and Messaging
- Keyword execution using proprietary and commercial test automation tools
  - Optical character recognition was used for verifications, which caused some reliability and maintenance issues
- 21 defects of different severities and priorities were found
  - Some of these defects existed in more than one smartphone model
  - The most severe of the defects caused the phone to hang with “System error” message on the display
  - About two thirds of the defects were discovered while modeling (reverse engineering), and the remaining third by execution (dynamic testing)
  - Most of the defects had already been previously found in traditional testing (both manual and automatic test execution), but they had not been fixed for some reason
    - However, there were also some that were totally new
  - Many of the defects were related to **concurrency issues**: performing some multimedia-related functionality in one application and then switching to another application caused unexpected behavior in some circumstances
  - In addition to defects found in applications, some were found in test automation tools, which was considered rather surprising, as these tools were quite mature

Case: Mobile Linux

- Media player application by Ixonos
  - New modeling challenge: real-time requirements
    - Playing videos, fast-forwarding, rewinding, pausing…
    - Although it was difficult, real-time support was eventually accomplished at model level
- Keyword execution using Linux accessibility features
  - API access to GUI components
  - Easier and more reliable than in S60 case
- Some minor bugs were found (both during modeling and execution)

Case: Android Phone

- Messaging, Contacts and Calendar applications
  - Action machines created for S60 were reused
  - Calendar was modeled with ATS4 AppModel and converted to TEMA models with an automatic converter
- Keyword execution was based on A-Tool by Symbio
- Optical character recognition was implemented with MS Office Imaging, which could have been more reliable
- Some bugs were found (both during modeling and execution)

Case: Android revisited (still ongoing)

- BBC News application
  - RSS reader optimized for BBC news feeds
- Self-made test automation for the emulator
  - Based on API access -> improved reliability
- Defects found so far:
  - 13 in total
  - 8 found during modeling
  - 5 found during execution
  - 2 of the bugs cause the application to crash
- Some Android built-in applications are currently being modeled

DEMO TIME!

Thank you!

TEMA TOOLSET NOW AVAILABLE!

Acknowledgements for financiers:
Tekes, Nokia, Ixonos, Symbio, Cybercom Plenware, F-Secure,
Qentinel, Prove Expertise, Academy of Finland (grant #121012)