Automated Model Extraction and Testing of Graphical User Interface Applications

Pekka Aho

Open Universiteit
www.ou.nl

Abstract

The industry practice for test automation through graphical user interface (GUI) is script-based. In some tools the scripts can be recorded from the user (capture & replay tools) and in others, the scripts are written manually. The main challenge with script-based test automation is the maintenance effort required when the GUI changes.

One attempt to reduce the maintenance effort has been model-based testing that aims to generate test cases from manually created models. When the system under testing (SUT) changes, changing the models and re-generating the test cases is supposed to reduce the maintenance effort. However, creating the formal models allowing test case generation requires very specialized expertise.

Random testing (monkey testing) is a maintenance-free and scriptless approach to automate GUI testing. In random GUI testing, the tool emulates an end-user by randomly interacting with the GUI under testing by pressing buttons and menus and providing input to other types of GUI widgets. More recent research aims to automatically extract the GUI models by runtime analysis while automatically exploring and interacting with the GUI (or during random testing). Test case generation based on extracted models is problematic but by comparing extracted GUI models of consequent versions it is possible to detect changes on the GUI. This approach aims to address the maintenance problem with scriptless regression testing and change analysis.

Open Universiteit
www.ou.nl

Background: Automated GUI model extraction and test case generation from extracted GUI models

GUITAR tool by Atif Memon's team, University of Maryland, USA

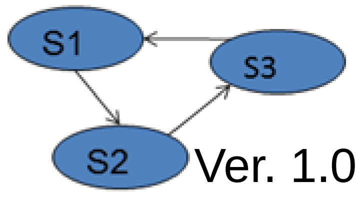
- Originally only for Java-based GUI applications, limited support on GUI widgets
- Using event-based graph models, problematic in capturing the behavior
- Plenty of publications, for example:
  - "GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing", Memon & al, 2003
  - https://sourceforge.net/projects/guitar/

GUI Driver tool by Pekka Aho, VTT, Finland

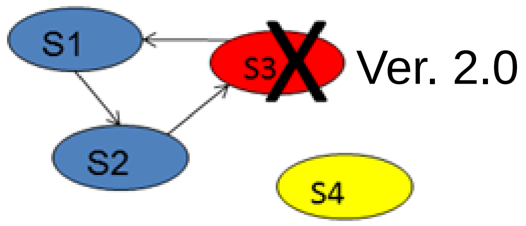
- Only for Java-based GUI applications
- Using state-based GUI models, recognizing a previously visited GUI state
  1. Automatically exploring Java GUI of the SUT to extract state-based model
  2. Saving the model in a format readable by GraphWalker tool to generate test sequences
  3. Executing the generated sequences on the SUT and comparing against the modeled behavior
- Publications:
  - Automated Java GUI modeling for model-based testing purposes, Aho & al, 2011
  - Enhancing generated Java GUI models with valid test data, Aho & al, 2011
  - Dynamic reverse engineering of GUI models for testing, Aho & al, 2013

Background: Shortcomings of test case generation from extracted GUI models

Generating regression test cases from extracted state models:



1. Extract GUI model from SUT version 1.0 – the reference model
2. Generate test scipts or test sequences from the extracted model
3. Executing the generated scripts on the later releases of the same SUT can detect changes in the behavior compared to version 1.0 and that way find regression bugs
4. BUT, when the SUT changes a lot, for example in version 2.0 the GUI view corresponding to state S3 is removed and new fuctionality as state S4 is added
   - Any new behavior is excluded from the models and therefore also from the test scripts.
   - All the test sequences including S3 will fail.



5. Extract a new GUI model from SUT version 2.0
   - The new reference model, cannot be used to detect changes of version 2.0
   - If the model is extracted everytime, what's the point in generating the scripts?

Background: Comparing extracted GUI models of consequent versions of an application to automatically detect changes between versions

Murphy tools by F-Secure Ltd, Finland:

- Originally developed as an internal GUI testing tool in a software company
  - https://github.com/F-Secure/murphy
- Based on Windows APIs and image recognition on screenshots of the GUI
- Uses virtual machines and works together with continuous integration (Jenkins)
    1. Iteratively create separate instructions (simple Python file) for Murphy tools and automatically explore the GUI of the SUT to create a model and to see if GUI coverage of the model is sufficient
    2. When the instructions are sufficient, set continuous integration to extract the GUI model from the latest version (for example 3 times a day)
    3. Automatically compare the latest model with the previous version to detect changes in the GUI
    4. Automatically report all changes to the test engineers to decide if the changes were intentional or bugs
- Provides a Web GUI for using the extracted models to design specific automated test cases
- Publications:
  - Industrial Adoption of Automatically Extracted GUI Models for Testing, Aho & al, 2013
  - Murphy tools: Utilizing extracted GUI models for industrial software testing, Aho & al, 2014
  - Making GUI Testing Practical: Bridging the Gaps, Aho & al, 2015

Background: Benefits of automatically comparing extracted GUI models to detect changes in the GUI

- Scriptless, no maintenance required (except if the tool can’t reach all parts of the GUI – then more instructions have to be given)
  - Any new part of the GUI will be explored automatically and reported as a change
  - Any removed part of the GUI will be reported as a change, but only once
- Fairly easy to extract a new model of each release through continuous integration (CI) but model extraction takes time (can be more than an hour)
  - CI + fully automated GUI change detection = maintenance-free regression testing
- Model comparison is relatively cheap (fast) and detects basically all changes in the GUI (depending on the level of abstraction on the models)
  - Also all the intentional changes are reported, but only once when the change happens the first time

Future research plans @ Open University

Research in TESTOMAT project

- Improving TESTAR tool with specific testing needs of the project partners
  - https://testar.org/
  - Improve the action selection mechanism to reach higher GUI and test coverage
    - Search-based genetic algorithms
    - Machine learning
    - Improve the test oracle mechanism
    - Test specification language
  - Improve how to measure the quality of testing
  - Re-use test scripts to derive TESTAR specifications and oracles
  - Parallel GUI execution to make GUI testing faster
  - Automated model extraction and change analysis + continuous integration
- Evaluating TESTAR and its benefits in real-life industrial setting

TESTOMAT project – a short introduction

- ITEA3 / TESTOMAT project - The Next Level of Test Automation
  - ITEA (Information Technology for European Advancement) is the EUREKA Cluster programme supporting innovative, industry-driven, pre-competitive R&D projects in the area of Software-intensive Systems & Services (SiSS).
  - Funding from the national funding agencies - Netherlands Enterprise Agency (Rijksdienst voor Ondernemend Nederland)
- Project web site: https://www.testomatproject.eu/
- October 2017 – September 2020
- 41 partners from 7 European countries
- Project topics:
  - WP3: Test Effectiveness
  - WP4: Test Prioritisation
  - WP5: Testing for Quality Standards
  - WP6: Test Automation Improvement Model
- Open Universiteit is leading the research of Task 3.5 - When less is more: Scriptless testing of graphical user interfaces

Belgium
Finland
Germany
Netherlands
Spain
Sweden
Turkey

TESTOMAT PROJECT
The Next Level of Test Automation

Open Universiteit
www.ou.nl