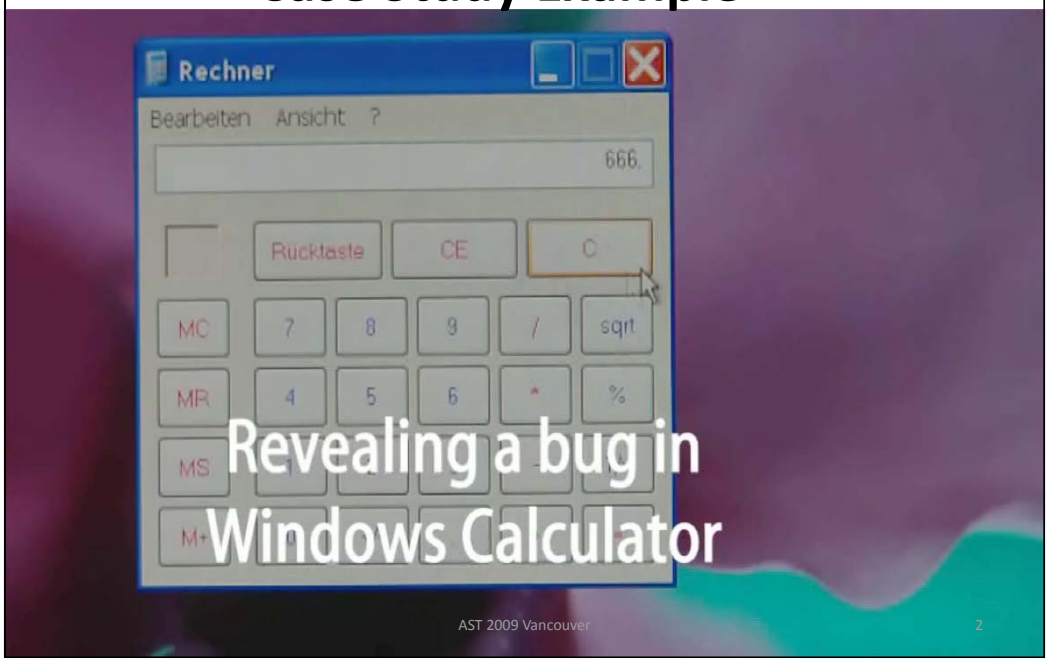
GUI Savvy End-to-End Testing with Smart Monkeys

Birgit Hofer, Bernhard Peischl and **Franz Wotawa**

Technische Universität Graz

Institute for Software Technology

Case Study Example



Content

- Introduction
  - GUI Testing
  - Monkey Testing
- Smart Monkeys, Modeling
- Case Study

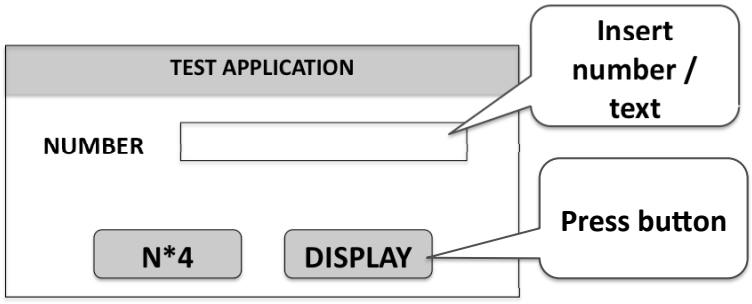
INTRODUCTION

Motivation

- **Shortcomings of traditional testing methods:**
  - Repeating the same sequences of actions
  - Mostly access through programming interface instead of the graphical user interface
  - GUI Testing usually done manually

GUI Testing

1. Testing the GUI
2. Testing the system (via the GUI)
   - ako system test!!!



GUI Testing

- Automation
  - Capture Replay Tools
  - Direct access to the GUI controls (buttons, text fields, …)
- GUI changes require changes in the test cases!

Monkey testing

- Instance of random testing
  - Randomly executing a program
  - Random sequences of interactions and inputs
- Completely automated
- Explore the software in a new way at each test run

Dumb monkeys

- General knowledge about how to access GUIs
- No application specific knowledge
- Can only detect
  - Memory leaks
  - Access violations
  - Program hang-ups
- Inexpensive to build

SMART MONKEYS

Smart monkeys

- Application specific knowledge
  - Model of the application
- Can find customer-relevant bugs
  - Verification of the response of an application
- More expensive to construct

Modeling for Smart Monkeys

- Modeling language:
  - Decision-based state machine (DBSM)
  - (S, s_0, p, T, E, V)
    - S .. States with start state s_0
    - p .. Mapping of states to properties
    - T .. Transition relation (with conditions, events, and changes on variables)
    - E .. Events
    - V .. Variables

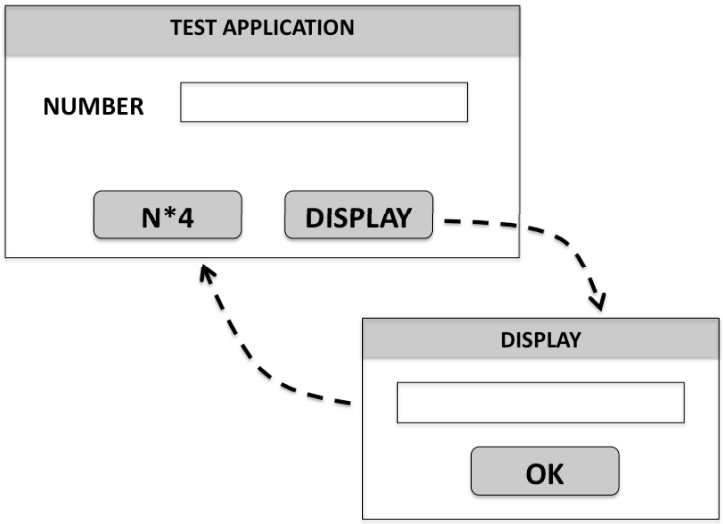
Important predicates

- **Properties** as oracle information
  - Is not hung
  - Window caption
  - All clickable controls enabled
  - Textfield value
  - …

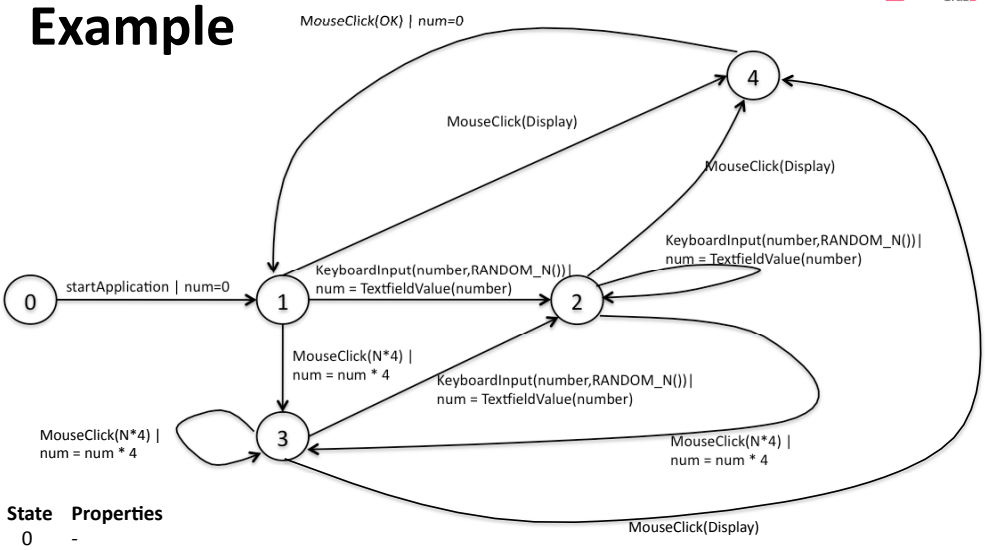
Important functions

- **Assignment** of values
  - Random_N (randomly generated numbers)
  - Random_S (randomly generated strings)
- **Access** to the values of the GUI
  - Textfield and checkbox values
- **Events** to communicate with the application
  - Start of the application
  - Mouse click
  - Keyboard input
  - …

Example



Example



State	Properties
0	-
1	*TextfieldProperty(number, 0) ∧ NotIsHungProperty()*
2	*NotIsHungProperty()*
3	*TextfieldProperty(number,num) ∧ NotIsHungProperty()*
4	*WindowCaptionProperty(DISPLAY) ∧ NotIsHungProperty()*

Semantics of DBSM - Traces

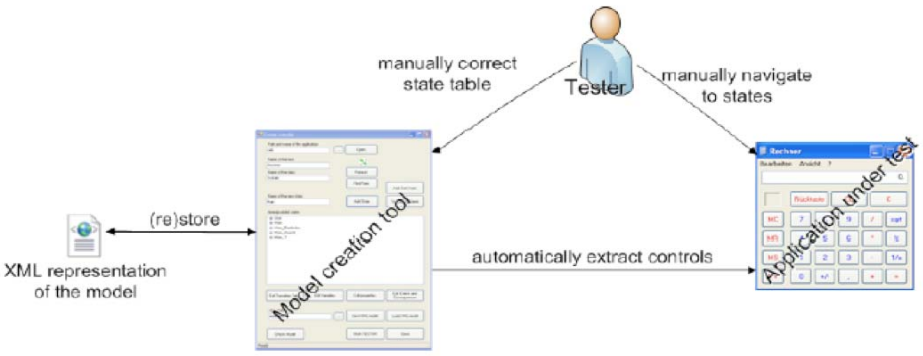
*Algorithm **trace (s, env)***

1. If **(s ∈ Σ)** then
   a) If **(eval(p(s), env) = FALSE**) then return **FAIL** .
   b) Randomly select one transition **(s, e, c, s') ∈ T**, call **trace(s', eval(c, env))** and store the result in **t**. Return the trace **e ⊕ t** as result.
2. If **(s ∈ D)**
   Call **trace(s', env)** for conditional transition **(s, co, s') ∈ T** where **eval(co, env) = TRUE**, and return the result.
3. In all other cases return the empty trace.

Technical realization

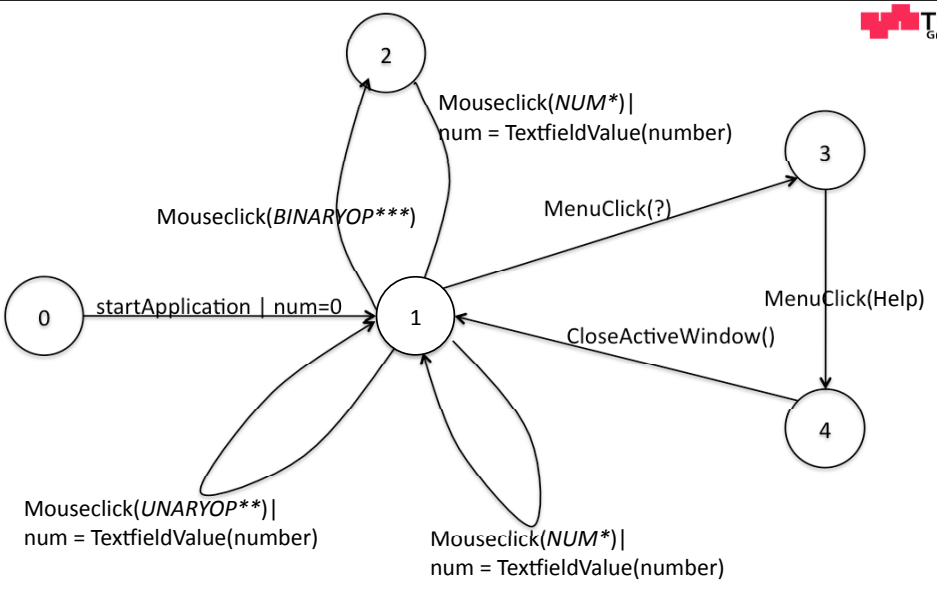
- Capture-Replay tool for extracting and accessing the elements of a GUI
  - Ranorex Automation Library (www.ranorex.com)
- Smart Monkey - Components
  - Semi-automated model creation
  - Model execution
  - Result evaluation

Modeling tool

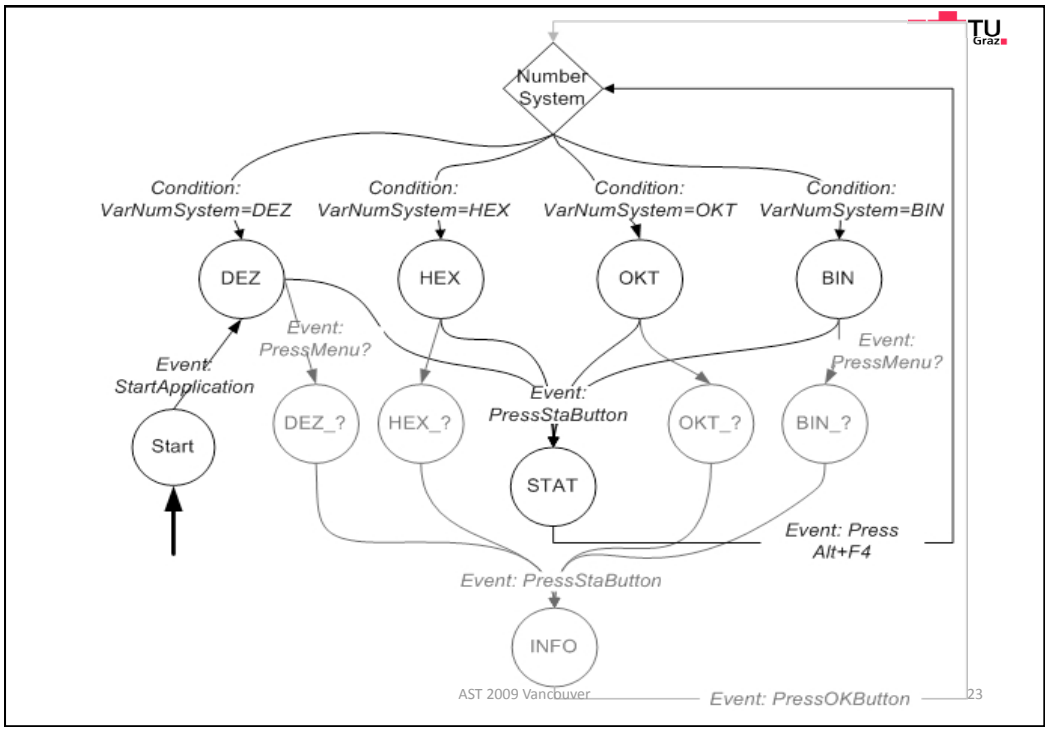


CASE STUDY – WINDOWS CALCULATOR

Windows Calculator Case Study



* NUM is one of the buttons {0,1,2,…,9}
** UNARYOP is one of the buttons {+/-, sqrt, %, 1/x, =}
*** BINARYOP is one of the buttons {,,/,, *,-,+,}



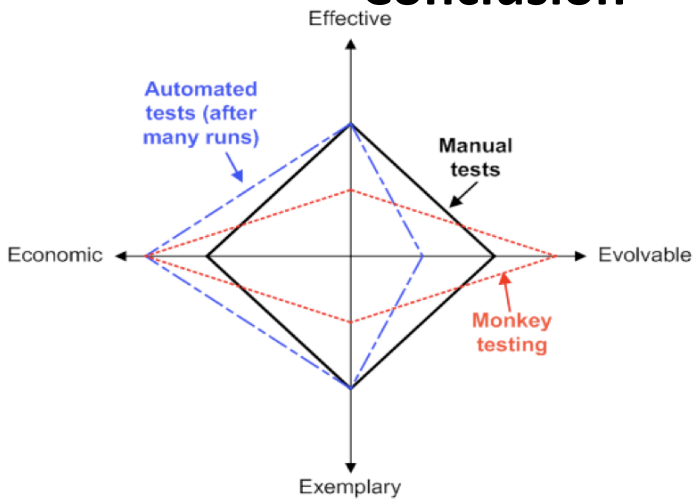
Errors found

- Event sequence for the 1st Error
  - The monkey produces a division by zero (e.g. *65 / 0*),
  - then it opens the menu item *?/Help*.
  - The value in the text field changes from the error message *'Division by 0 not possible'* to a number.
- Event sequence for the 2nd Error
  - The monkey produces a division by zero,
  - then it opens the menu item *?/Info*.
  - The info menu does not appear

Future work

- Comprehensive case study with an open source project
- Adaption of the monkey for web testing

Conclusion



→ Powerful supplement to systematic testing

Thank you for your attention!