Reverse Engineering of Graphical User Interfaces

Inês Coimbra Morgado (coimbra.ines@fe.up.pt)
Ana C. R. Paiva (apaiva@fe.up.pt)
João Pascoal Faria (jpf@fe.up.pt)

Agenda

- Motivation and Context
- Implementation
- Results
- Conclusions & Future Work
Reverse Engineering

“Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction”

Motivation

- The model of a system is useful for:
  - Checking the system’s properties
  - Changing platforms
  - Testing
Problem

- Model-Based Testing (MBT) requires a formal model
- Most of models are not updated or are simple representative schematics
- The manual construction of a model is a too time consuming and error prone process
Goals

- Diminish the effort of obtaining part of the GUI model
  - Reverse Engineering

- Make a contribution to the automation of GUI testing

- Make a contribution to increase the adoption of MBT techniques

State of the Art

- **Static Reverse Engineering**
  - Source Code Analysis
  - Code parsing is a common technique
  - Examples:
    - GUISurfer, ManSART, Ciao, Bouillon et al., Vanderdockt et al.

- **Dynamic Reverse Engineering**
  - Application in Run Time
  - Information on concurrency and memory management
  - Examples:
    - GUIRipper, Shehady et al., VESP
State of the Art

- **Dynamic vs Static**
  - Information on concurrency and memory management
  - No source code needed
  - Detection of modifications of the GUI in run time
  - Extraction in run time (like testing)

- **Existing Dynamic**
  - Lack navigation map
  - Lack information on dependencies between elements

ReGUI
Architecture

Exploration Process

- Phase 1
  - Initial Structure
  - Initial State

- Phase 2
  - Interaction
  - Structure Extraction
  - Behaviour Extraction (navigation, dependencies)
Problems

- Element Identification
- Exploration Order

Exploration Order

![Menu Options](image-url)
Problems

- Elements Identification
- Exploration Order
- Synchronisation
- Closing Windows

Outputs

- ReGUI Tree
- Window Graph
- Navigation Graph
- Disabled Graph
- Dependency Graph
- Spec# Model
ReGUI Tree

UI Automation Tree

Untitled - Notepad
File
New
Open...
Save
Save As...
Page Setup...
Print...
Exit
Edit
Format
View
Help

UI Automation Tree

Untitled - Notepad
File
Edit
Undo
Cut
Copy
Paste
Delete
Find...
Find Next
Replace...
Go To...
Select All
Time/Date
Format
View
Help

ReGUI Tree

outputs

• ReGUI Tree
• Window Graph
• Navigation Graph
• Disabled Graph
• Dependency Graph
• Spec# Model
Window Graph

Outputs

- ReGUI Tree
- Window Graph
- **Navigation Graph**
- Disabled Graph
- Dependency Graph
- Spec# Model
Navigation Graph

Outputs

- ReGUI Tree
- Window Graph
- Navigation Graph
- **Disabled Graph**
- Dependency Graph
- Spec# Model
Disabled Graph

Outputs

- REGUI Tree
- Window Graph
- Navigation Graph
- Disabled Graph
- **Dependency Graph**
- Spec# Model
Dependency Graph

Outputs

- ReGUI Tree
- Window Graph
- Navigation Graph
- Disabled Graph
- Dependency Graph
- Spec# Model
Spec# Generation Rules

Window

Rule 1. □ - windowHeight
Spec:
namespace windowHeight;
var windowHeight = 1; // if main window
var windowHeight = 3; // if other windows

Window to Menu

Rule 2. □ → △ / ▲
Spec:
namespace MenuOptionName;
var menuOptionName = 1; // if △ and
var menuOptionName = 2; // if △ and
var menuOptionName = 3; // if △

// Apply Rule 1 to WindowName
var menuOptionName = 1; // if main window
var menuOptionName = 2; // if other windows
[Action] MenuOptionName()
requires menuOptionName == 1; { }

Window to Button

Rule 3. □ → ○
Spec:
namespace ButtonName;
var buttonName = 1; // if main window
var buttonName = 3; // if other windows
[Action] ButtonName()
requires buttonName == 1; { }

Menu to Menu

Rule 4. △ → △ / ▲
Spec:
namespace MenuOptionName1, MenuOptionName2
var menuOptionName2 = 3;
[Action] void MenuOptionName1()
requires menuOptionName1 == 1;
menuOptionName2 = 1; // or 2 if ▲
[Action] void MenuOptionName2()
requires menuOptionName2 == 1; { }

Menu to Window

Rule 5. △ → □
Spec:
namespace WindowName
var WindowName, windowHeight = 1;
var menuOptionName = 3;
[Action] void MenuOptionName()
requires menuOptionName == 1;
WindowName.windowsName = 1;
WindowName.menuOptionName = 3;
// Rule 1 for WindowName
// if not yet constructed

Button to Window

Rule 6. ○ → □
Spec:
namespace ButtonName, WindowName
var buttonName = 1;
[Action] void ButtonName()
requires buttonName == 1;
WindowName.windowsName = 1;
buttonName = 3;
// Rule 1 for WindowName
// if not yet constructed
Spec# Model

```csharp
namespace WindowUntitled__Notepad;
var windowUntitled__Notepad = 1;
var menu_itemFile_menu_barApplication_windowUntitled__Notepad = 1;
var menu_itemOpen_menu_itemFile_windowUntitled__Notepad = 3;

[Action] Menu_itemFile_menu_barApplication_windowUntitled__Notepad()
requires menu_itemFile_menu_barApplication_windowUntitled__Notepad == 1;

[Action] Menu_itemOpen_menu_itemFile_windowUntitled__Notepad()
requires menu_itemOpen_menu_itemFile_windowUntitled__Notepad == 1;

namespace WindowOpen;
var windowOpen = 3;
var buttonClose_windowOpen = 3;

[Action] ButtonClose_windowOpen()
requires buttonClose_windowOpen == 1;
```

Conclusions

- Generates graphs for easy results visualisation and interpretation
  - The navigation graph allows us to easily analyse usability related issues
  - The window graph allows us to rapidly verify if the windows are correctly connected
  - All graphs enable an easy identification of specification related issues
- Eases checking the available actions in a certain state of the application
Conclusions

- Diminish the effort of building a model for MBGT
  - Generates a Spec# Model

- Extracts important information on structure and behaviour

- Lack of standards was an obstacle

Future Work

- Simulate more user actions
- Interact with open windows
- Extract more dependencies
- Improve the Spec# generation
Reverse Engineering of Graphical User Interfaces

Inês Coimbra Morgado (coimbra.ines@fe.up.pt)
Ana C. R. Paiva (apaiva@fe.up.pt)
João Pascoal Faria (jpf@fe.up.pt)

Thank You!