Reverse Engineering of Graphical User Interfaces

Universidade do Porto
Faculdade de Engenharia
FEUP

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

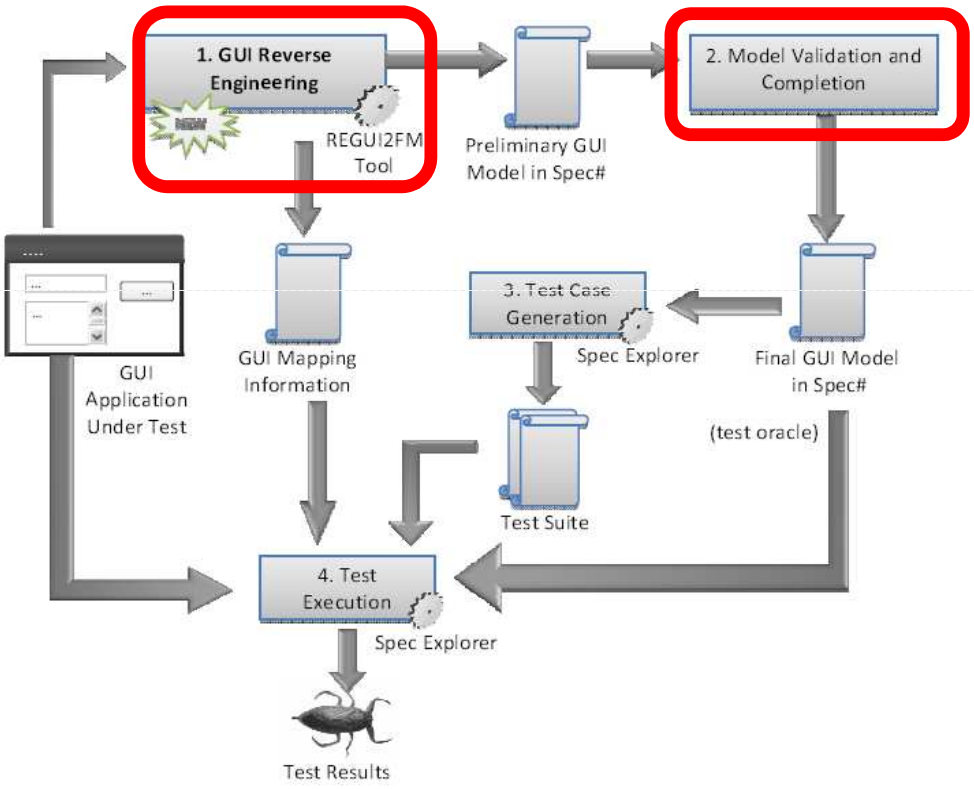
Reverse Engineering

"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

AMBER iTest



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

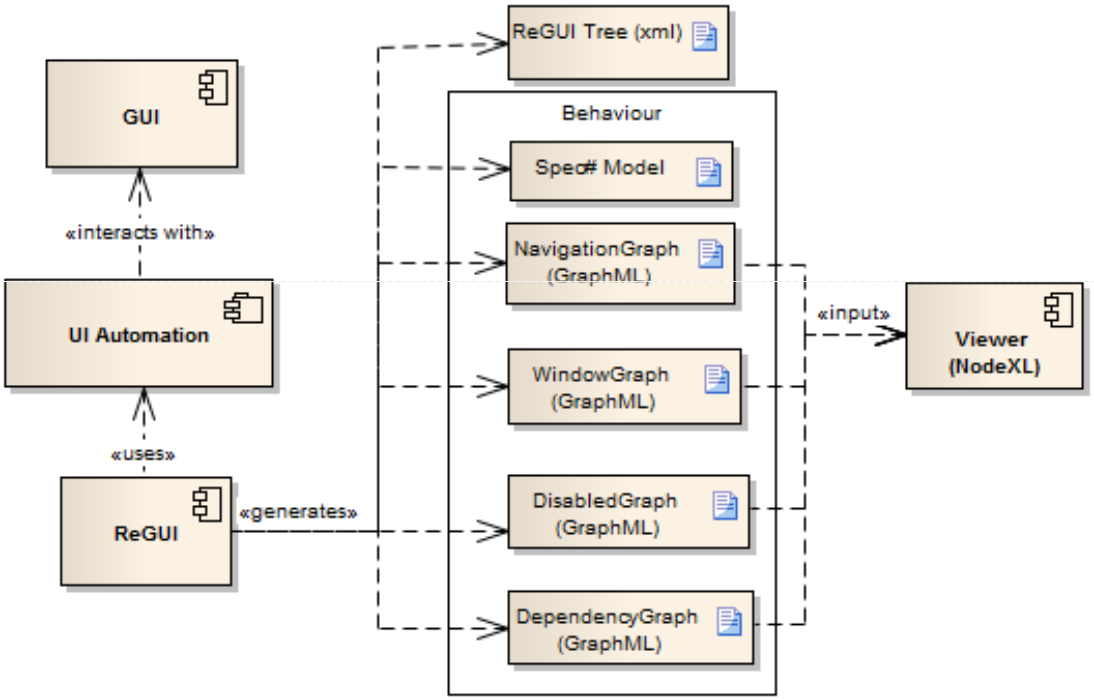
- Static Reverse Engineering
  - Source Code Analysis
  - Code parsing is a common technique
  - Examples:
    - GUISurfer, ManSART, Ciao, Bouillon et al., Vanderdockt et al.
- Dynamic Reverse Engineering
  - Application in Run Time
  - Information on concurrency and memory management
  - Examples:
    - 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



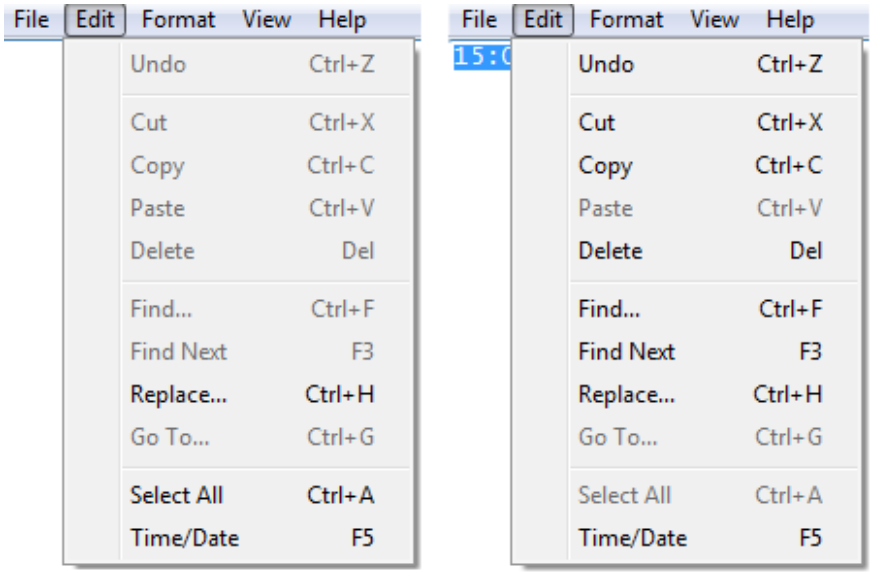
Exploration Process

- Phase 1
  - Initial Structure
  - Initial State
- Phase 2
  - Interaction
  - Structure Extraction
  - Behaviour Extraction (navigation, dependencies)

Problems

- Element Identification
- Exploration Order

Exploration Order



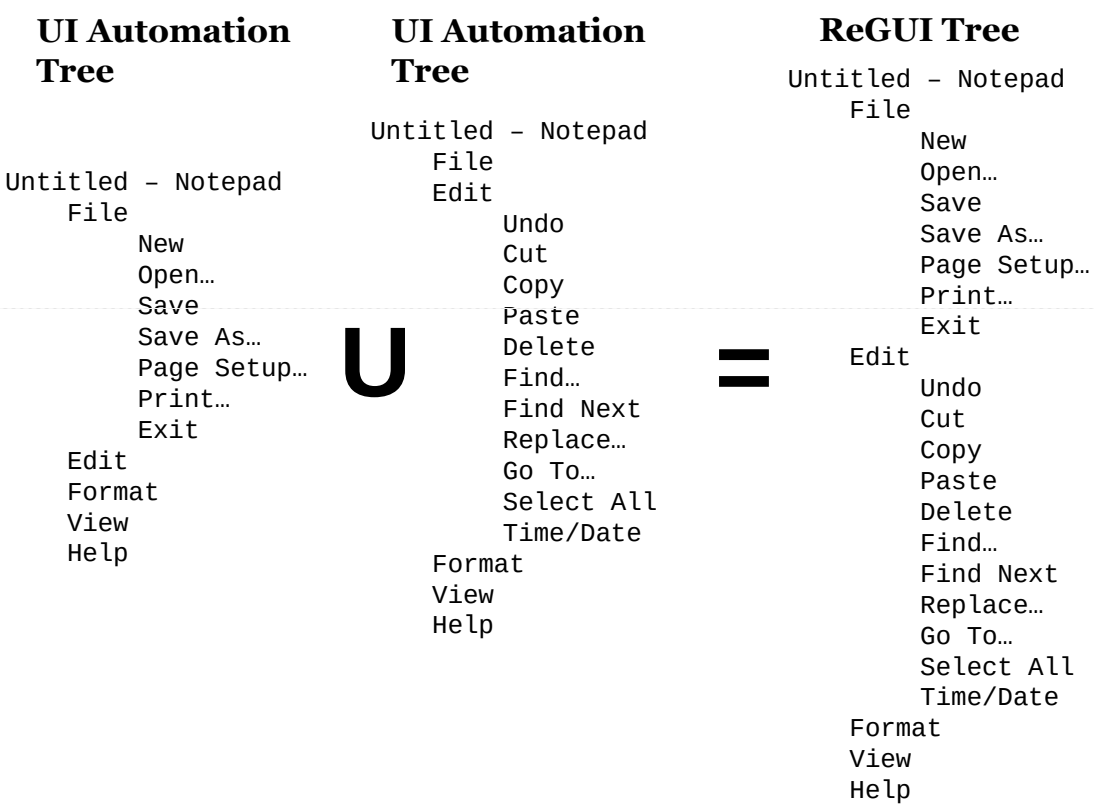
Problems

- Elements Identification
- Exploration Order
- Syncronisation
- Closing Windows

Outputs

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

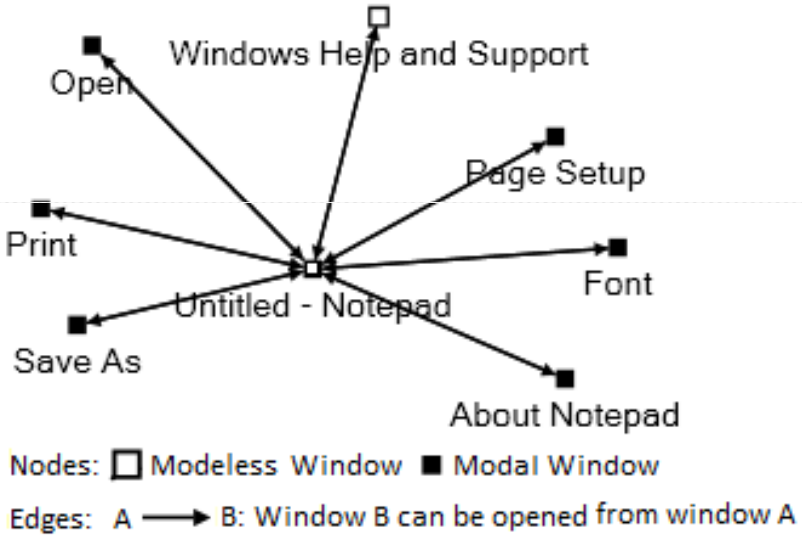
ReGUI Tree



Outputs

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

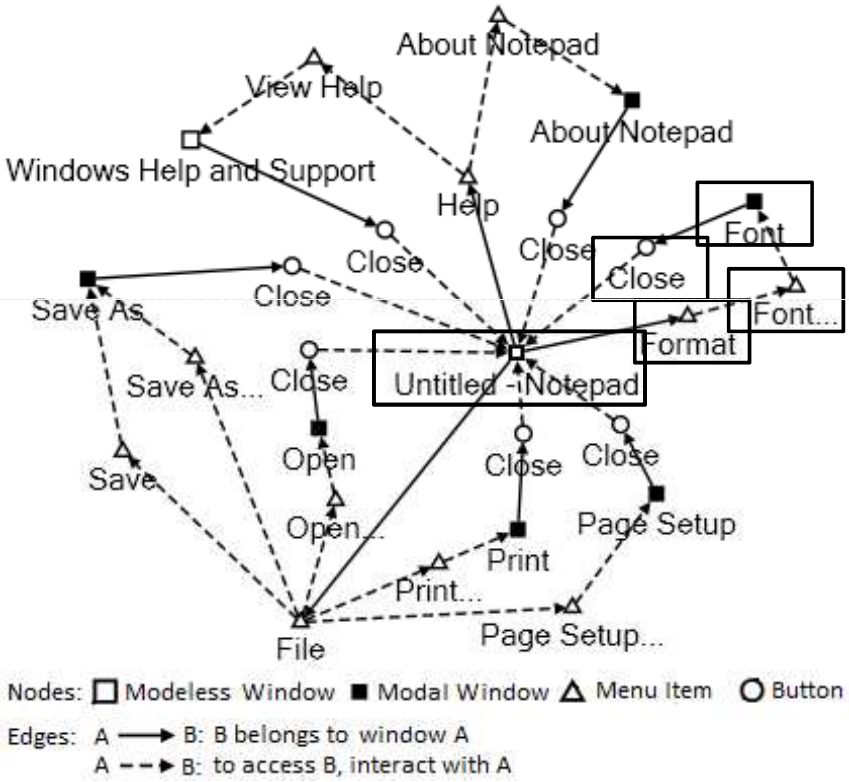
Window Graph



Outputs

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

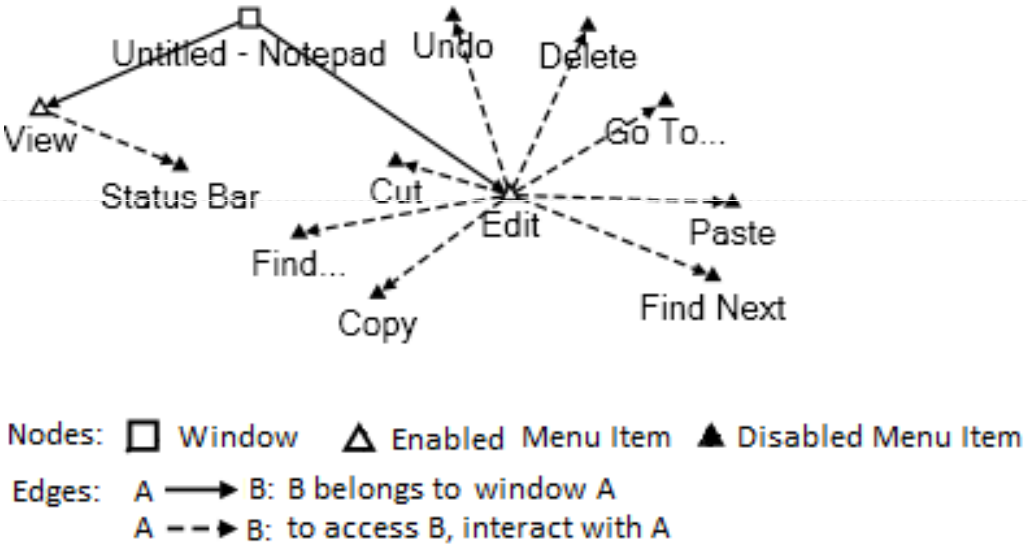
Navigation Graph



Outputs

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

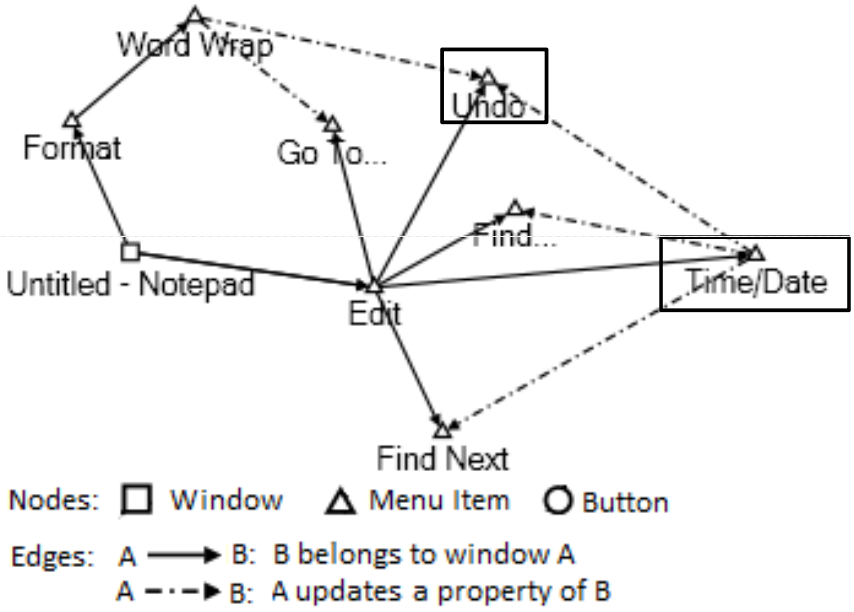
Disabled Graph



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 □ - windowName

Spec#:

```
namespace windowName;
var windowName = 1; //if main window
var windowName = 3; //if other windows
```

Window to Menu

Rule 2 □ ——————→ Δ / ▲

windowName MenuOptionName

Spec#

```
// Apply Rule 1 to WindowName
var menuOptionName = 1; //if Δ and
                        //main window
var menuOptionName = 2; //if ▲ and
                        //main window
var menuOptionName = 3; //if other windows
[Action] MenuOptionName()
requires menuOptionName == 1; { }
```

Window to Button

Rule 3 □ ——————→ ○

WindowName ButtonName

Spec#:

```
// Apply Rule 1 to WindowName
var buttonName = 1; //if main window
var buttonName = 3; //if other windows
[Action] ButtonName()
requires buttonName == 1; { }
```


Spec# Generation Rules

Menu to Menu

Rule 4 Δ - - - - - - - - - - - - - - → Δ / ▲

MenuOptionName1 MenuOptionName2

Spec#:

```
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 Δ - - - - - - - - - - - - - - → □

MenuOptionName WindowName

Spec#:

```
[Action] void MenuOptionName()
requires menuOptionName == 1; {
  WindowName.windowName = 1;
  menuOptionName = 3;
}
//Rule 1 for WindowName
//if not yet constructed
```

Button to Window

Rule 6 ○ - - - - - - - - - - - - - - → □

ButtonName WindowName

Spec#:

```
[Action] void ButtonName()
requires buttonName == 1; {
  WindowName.windowName = 1;
  buttonName = 3;
}
//Rule 1 for WindowName
//if not yet constructed
```

Spec# Model

```
namespace WindowUntitled___Notepad;                                    //Rule 1
var windowUntitled___Notepad = 1;
var menu_itemFile_menu_barApplication_windowUntitled___Notepad = 1;     //Rule 2
var menu_itemOpen_menu_itemFile_windowUntitled___Notepad = 3;           //Rule 4

[Action] Menu_itemFile_menu_barApplication_windowUntitled___Notepad()   //Rule 2
requires menu_itemFile_menu_barApplication_windowUntitled___Notepad == 1;{
    menu_itemOpen_menu_itemFile_windowUntitled___Notepad = 1;           //Rule 4
};

[Action] Menu_itemOpen_menu_itemFile_windowUntitled___Notepad()
requires menu_itemOpen_menu_itemFile_windowUntitled___Notepad == 1;{
    menu_itemOpen_menu_itemFile_windowUntitled___Notepad = 3;           //Rule 5
    WindowOpen.windowOpen = 1;
};

namespace WindowOpen;                                                  //Rule 1
var windowOpen = 3;
var buttonClose_windowOpen = 3;                                        //Rule 3

[Action] ButtonClose_windowOpen()
requires buttonClose_windowOpen == 1;{
    buttonClose_windowOpen = 3;                                        //Rule 6
    WindowUntitled___Notepad.windowUntitled___Notepad = 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

Universidade do Porto
Faculdade de Engenharia
FEUP

Thank You!

Inês Coimbra Morgado (coimbra.ines@fe.up.pt)

Ana C. R. Paiva (apaiva@fe.up.pt)
João Pascoal Faria (jpf@fe.up.pt)