Why UI testing is SO hard?
(and what we can do about it)

ivan.inozemtsev@xored.com
An abstract testing tool

• Control an application-under-test (start/stop)
• Manage test resources (create/modify/exchange)
• Maintain backward compatibility
• Execute tests (locate elements/send commands/perform assertions)
Testing Tool Runtime

- Identifying and locating elements
- Interacting with elements
- Asserting their state
How an application looks
How user sees it

- Package Explorer view
- Type Hierarchy view (inactive)
- (dirty) Java Editor
- Perspective switch buttons
- Collapsed text
- Markers (error and todo)
How the tool sees it
Locating elements
Locating elements
editors vs views
Locating elements
editors vs views

Editors

Views
Identifying caret position

StyledText.getCaretOffset() == 551
Identifying headless windows

Documentation

Proposals
Performing Actions
Clicking a context menu

- User: click “Refresh” menu item
- What can go wrong? Get a menu, find an item, send event to it
- item.getText() == Refresh\tF5
Setting text

- org.eclipse.swt.widgets.Text.setText("foo") automatically sends SWT.Verify and SWT.Modify. What can be more simple?
- Application: listens for KeyUp/Down events to perform some extra actions, affecting UI
- Fail
Checking an item

- Click or check/uncheck?
- How to get/set current value?
  - MarkerViewerContentProvider
  - CheckboxCellEditor
Asserting UI state
 Asserting Decorations

- ControlDecoration is not a control
- No references from control to decorations
- How to assert its presence and text?
Asserting Images

- Images contain a ton of useful information
- May be decorated by other images
- Don’t retain their origin – just a handle to OS resource

On a screenshot:
org.eclipse.jdt.ui/icons/full/obj16/field_private_obj.gif decorated with final_co.gif and static_co.gif
Asserting styled tree items

SWT Treeltem has no idea about those colors
Main points

- UI testing is really hard
- Blackbox testing is a myth (or dream)
- Someone has to be an expert in internals – but who? Tool developers or QA engineers?
RCP Testing Tool

- Focus on Eclipse technologies
- Support for SWT, JFace, Forms, Draw2D, GEF, GMF, Graphiti
- Control state of workspace, workbench, preferences
- Automatic tracking of jobs, Display execs, databindings
RCP Testing Tool – Demo

• Introduce a compile error in Java Editor
• Make sure it is displayed
  • In rulers
  • In editor (underlined)
  • In Package Explorer view
  • In Problems view
  • In Outline view
RCP Testing Tool – Demo

- State control
- Recording just works
- Human-readable identification of widgets
- OS-independent hotkeys
- Powerful assertions
with [get-editor "Program.java" | get-text-viewer] {
  set-caret-pos 3 44
  key-type BackSpace
}
get-button "Save (M1+S)" | click
Highlights

get-editor "Program.java"
  | get-left-ruler
  | get-ruler-column AnnotationColumn
  | hover-ruler -line 3
with [get-view Problems | get-tree] {
  get-item "Errors (1 item)/Syntax error.*"
  | get-property "image.path"
  | equals "org.eclipse.ui.ide/icons/full/obj16/error_tsk.gif"
  | verify-true

  get-item "Errors (1 item)/Syntax error.*"
  | get-property "values['Location']"
  | equals "line 3"
  | verify-true
}

Highlights
get-view Outline | get-tree
| get-item "Program/main(String[]) : void"
| get-property "image.decorations[1].path"
| equals "org.eclipse.jdt.ui/icons/full/ovr16/error_co.gif"
| verify-true
What's next?

• http://eclipse.org/rcptt
• http://www.eclipse.org/rcptt/documentation/userguide/getstarted/
• http://eclipse.org foraums/eclipse.rcptt
Thank you!

Visit http://eclipse.org/rcptt