CSE 403
Lecture 15

UI Automation / Functional Testing

Reading:
How to Break Software, Ch. 2, Whittaker

slides created by Marty Stepp
http://www.cs.washington.edu/403/
Recall: Kinds of testing

- **unit testing**: looks for errors in objects or subsystems

- **integration testing**: find errors when connecting subsystems

- **system testing**: test entire system behavior as a whole, with respect to scenarios and requirements
  - functional testing: test whether system meets requirements
  - performance testing: nonfunctional requirements, design goals
  - acceptance / installation testing: done by client
Functional testing

• **ad-hoc**: Just run the product and click things.

• **UI automation**: Simulate usage of a product's UI in code.
  – "record" usage and play back later
  – or write code to simulate mouse clicks

• Many developers rely too much on ad-hoc testing.
  – pro: Simple; fast; does not require specialized knowledge
  – con: Inaccurate; must be repeated many times; poor at catching regressions; costs more and more time later in the project
  – The ideal is a mix of both kinds of UI testing.
Flush out error messages

- empty strings (or strings made entirely of spaces, etc.)
- invalid strings (too short, too long; special characters)
- 0 or negative numbers
- settings that do not make sense in combination
Input buffer overflows

• When prompted for input, try to put in a very long string
  – Will it be accepted, leading to a strange appearance on the UI?
Overflow data structures

- Whenever a UI shows a page or list, try to add to that list until it overflows, causing crashes, errors, or awkward appearance.
Violate app's assumptions

• What does the app's GUI do if:
  – The file it is using gets externally modified or deleted?
  – The network goes down (or just slows down) unexpectedly?
  – The OS amount of memory available drops?
  – The processor becomes busy and the app slows down?
Repeat/duplicate inputs

- Try the same input multiple times to expose bugs:
  - re-add an existing user
  - create a file that already exists
  - delete a file that is already deleted or that does not exist
  - click the button to perform an action multiple times
    - "Buy", "Order", "Check Out"
    - Will the customer be charged twice?
  - web apps: click "Back" and then try an action again
    - Was the developer expecting this?
Cause invalid outputs

• Most GUIs stop you from supplying bad input.
  – But maybe you can still cause bad output.

• Example: Set calendar to an invalid date:
  – The UI properly restricts you to Feb 1-28.
  – Choose a leap year, then select Feb 29.
  – Change year back to a non-leap year.
  – Feb 29 will still be shown as a valid choice.

• Example: TextPad "Block Select" feature
  – toggle on, copy text, toggle off, paste
Test moving / sizing

• Many UI designers don't consider what their screen or page will look like when resized to extremes
  – try resizing the window or adding input to grow a window's size
    • does the window add scrollbars?
    • do some controls disappear or overlap?
    • does text begin to wrap in odd ways?
Test enabling / disabling

- Enable/disable elements to indicate whether they can be used.

- Test the enabling/disabling of all UI elements.
  - Do elements disable/re-enable when they are supposed to?
  - Is it ever possible to click an element that shouldn't be clickable, or impossible to click an element that should be clickable?
Android testing

• Google recommends creating an entire separate test Eclipse project to store your unit tests for an Android app

• put in tests/ subdir of main app
  – MyProject/
    • AndroidManifest.xml
    • res/ ... (resources for main app)
    • src/ ... (source code for main app) ...
  • tests/
    – AndroidManifest.xml
    – res/ ... (resources for tests)
    – src/ ... (source code for tests)
Android UI testing

- **uiautomatorviewer**
  - allows you to inspect current state of an on-screen UI

- **UiAutomatorTestCase**
  - a specialized JUnit test that can construct and interact with UI controls

- **UI Automatker Monkey**
  - simulates pseudo-random UI interaction to test UI robustness and stress testing
import com.android.uiautomator.core.*;
import com.android.uiautomator.testrunner.*;

public class LaunchSettings extends UiAutomatorTestCase {
    public void testDemo() throws UiObjectNotFoundException {
        getUiDevice().pressHome();

        // simulate a user bringing up the All Apps screen
        UiObject allAppsButton = new UiObject(new UiSelector().description("Apps"));
        allAppsButton.clickAndWaitForNewWindow();

        // simulate the user bringing up the Apps tab
        UiObject appsTab = new UiObject(new UiSelector().text("Apps"));
        appsTab.click();

        // simulate a user swiping until they come to the Settings app icon
        UiScrollable appViews = new UiScrollable(new UiSelector().scrollable(true));
        appViews.setAsHorizontalList();

        // simulate a user click to launch the app
        UiObject settingsApp = appViews.getChildByText(new UiSelector()
                .className(android.widget.TextView.class.getName()), "Settings");
        settingsApp.clickAndWaitForNewWindow();

        // validate that the package name is the expected one
        UiObject settingsValidation = new UiObject(new UiSelector()
                .packageName("com.android.settings"));
        assertTrue("Unable to detect Settings", settingsValidation.exists());
    }
}
// Start main activity of the application under test
mActivity = getActivity();

// Get a handle to Activity object's main UI widget, a Spinner
mSpinner = (Spinner) mActivity.findViewById(
    com.android.example.spinner.R.id.Spinner01);

// Set Spinner to a known position
mActivity.setSpinnerPosition(TEST_STATE_DESTROY_POSITION);

// Stop activity – onFinish() should save state of Spinner
mActivity.finish();

// Re-start Activity – onResume() should restore Spinner state
mActivity = getActivity();

// Get Spinner's current position
int currentPosition = mActivity.getSpinnerPosition();

// Assert that current position is same as the starting position
assertEquals(TEST_STATE_DESTROY_POSITION, currentPosition);
Robotium

- Robotium
  - UI test automation tool for Android apps
  - based on very popular Selenium web app UI test tool

- tutorials:
  - [http://www.youtube.com/watch?v=VYk1_kpSzQg](http://www.youtube.com/watch?v=VYk1_kpSzQg)
import com.jayway.android.robotium.solo.*;
public class EditorTest extends ActivityInstrumentationTestCase2<EditorActivity> {
    private Solo solo;

    public EditorTest() {
        super(EditorActivity.class);
    }

    public void setUp() throws Exception {
        solo = new Solo(getInstrumentation(), getActivity());
    }

    public void testPreferenceIsSaved() throws Exception {
        solo.sendKey(Solo.MENU);
        solo.clickOnText("More");
        solo.clickOnText("Preferences");
        solo.clickOnText("Edit File Extensions");
        assertTrue(solo.searchText("rtf"));
        solo.clickOnText("txt");
        solo.clearEditText(2);
        solo.enterText(2, "robotium");
        solo.clickOnButton("Save");
        solo.goBack();
        solo.clickOnText("Edit File Extensions");
        assertTrue(solo.searchText("application/robotium"));
    }

    public void tearDown() throws Exception {
        solo.finishOpenedActivities();
    }
}
Selenium

- Records and plays back automated "test cases" of walking through a web app's UI
- can **assert** various aspects of the web page state to make sure the page looks right
- tests can be saved as HTML
  - or can be written in:
    - Java
    - Ruby
    - Python
    - ...
Components of Selenium

- Selenium IDE - record/playback tool as Firefox add-on
  - produces Selenium Core test cases

- Selenium Core - HTML/JS framework that runs in any browser
  - for testing browser compatibility

- Selenium Remote Control (RC) - automation framework
  - for running tests on a schedule
  - used with Eclipse or a dedicated server
import com.thoughtworks.selenium.*;

public class NewTest extends SeleneseTestCase {
    public void setUp() throws Exception {
        setUp("http://www.google.com/", "*firefox");
    }

    public void testNew() throws Exception {
        selenium.open("/");
        selenium.type("q", "marty stepp");
        selenium.click("btnG");
        selenium.waitForPageToLoad("30000");
        assertTrue(selenium.isTextPresent("University of Washington"));
    }
}

import java.util.*;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.*;

public class GoogleSuggest {
    public static void main(String[] args) throws Exception {
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com/webhp?complete=1&hl=en");

        // Enter the query string "Cheese"
        WebElement query = driver.findElement(By.name("q"));
        query.sendKeys("Cheese");
        long end = System.currentTimeMillis() + 5000; // Sleep 5 sec
        while (System.currentTimeMillis() < end) {
            WebElement resultsDiv = driver.findElement(By.className("gac_m"));
            if (resultsDiv.isDisplayed()) break;
        }

        // And now list the suggestions
        List<WebElement> allSuggestions = driver.findElements(By.xpath("//td[@class='gac_c']"));
        for (WebElement suggestion : allSuggestions) {
            System.out.println(suggestion.getText());
        }
    }
}
Java Swing UI testing

• Abbot - Functional UI testing for Java desktop app GUIs
  – (not for Android apps)
  – works with Costello companion app
  – http://abbot.sourceforge.net/