



Java UI Testing

Technology and real experience

Alexandre Iline,
Maria Tishkova,
Alexander Kouznetsov

Agenda

- **Who are we**
- Introduction to testing
- Tools
- Automation effectiveness
- The mantra and the two approaches
- Misconceptions
- Conclusion

Who are we

Work for SUN Microsystems

Some products we test:

- > NetBeans

and NetBeans packs:

- > Enterprise Pack (Java Studio Enterprise)

- > Mobility Pack

- > Visual Web Pack (Java Studio Creator)

- > C/C++ Development Pack, Sun Studio

What we're going to talk about

How to make UI testing ...

- **less expensive**
by doing test automation, but doing it smart
- **more useful**
shorter test cycles, earlier bug detection
- **more fun**
there is no fun in testing UI manually

Agenda

- Who are we
- **Introduction to testing**
- Tools
- Automation effectiveness
- The mantra and the two approaches
- Misconceptions
- Conclusion

Testing is ...

- ... something which ensures product quality
- ... something which helps development to verify against regressions
- ... activities which have to be done repeatedly
- ... needed to be done for any supported configurations

UI Testing is ...

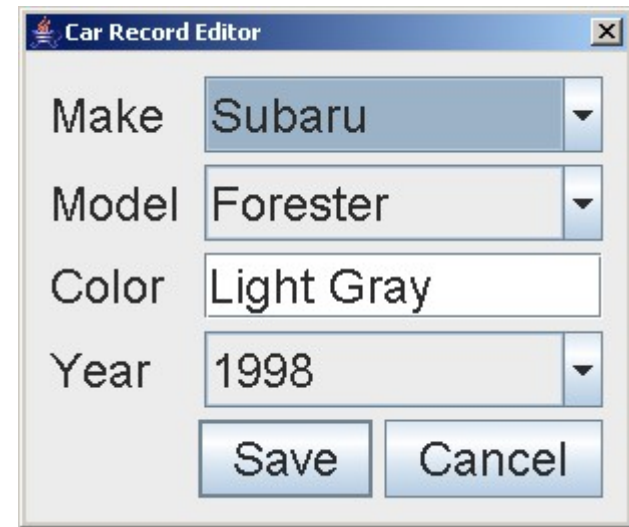
- ... a testing
 - As such, it has to be done repeatedly for each and every release and also every supported configuration
- ... a tedious job
 - of clicking through tons of screens, menus and buttons
- ... very expensive
 - if not done right, full testing takes a huge amount of time

Consider an example ...

UI Testing example

Vehicle ordering

- > Open car setting dialog
- > Change make (“Subaru”)
- > Change model (“Forester”)
- > Change color (“Light Gray”)
- > Change year (“1998”)
- > Submit
- > Verify that the information has been submitted
- > Verify what information has been submitted



Test automation

- **Reduces price of testing**
Eliminates (partially) need of repeating manual test cycles
- **Shortens test cycle**
Automated tests work faster and could be executed simultaneously for different configurations
- **Ensures earlier bug detection**
Automated tests could be executed as often as needed
- **Deeper level of testing**
With automated tests it is possible to go as deep into product as needed.
- **Makes testing fun!**

Terminology

- **Test**
a (small) program which verifies tested product functionality through product interface
- **Suite**
a set of tests which are executed together
- **Harness**
a (set of) auxiliary tool managing test execution, test result representation, storage, and sometimes failure analysis

Approaches to UI automation

- Recording/coding
 - > User actions could be recorded into test
 - > Test could be coded in some language
- Language
 - > XML
 - > scripting languages
 - > high-level languages
- Coordinates/components
 - > Test operates in terms of event coordinates
 - > Test operates in terms of UI components and actions

Agenda

- Who are we
- Introduction to testing
- **Tools**
- Automation effectiveness
- The mantra and the two approaches
- Misconceptions
- Conclusion

Tools

- Native vs. Java
Native tools are not good enough
- Commercial vs. open-source
No real offering from commercial tools: free tools provide everything the commercial ones do.
- Rest is:
 - > Jemmy
 - > Abbot
 - > JFCUnit

Jemmy

- Open-source (<http://jemmy.netbeans.org>)
- Java library

```
new JComboBoxOperator(new JDialogOperator("Car"))
    .selectItem("Green");
```
- Covers all Swing and AWT
- A lot of synchronization work is done behind the scene
- Uses `java.awt.Robot` or event dispatching
- Easy to extend (<http://jellytools.netbeans.org>)

Agenda

- Who are we
- Introduction to testing
- Tools
- **Automation effectiveness**
- The mantra and the two approaches
- Misconceptions
- Conclusion

Automation effectiveness formula

$$E_A = \frac{T_M * N_R * N_C}{T_D + T_S * N_R * N_C}$$

E_A - test automation effectiveness

Number of ...

N_R – tested releases

N_C – tested configurations

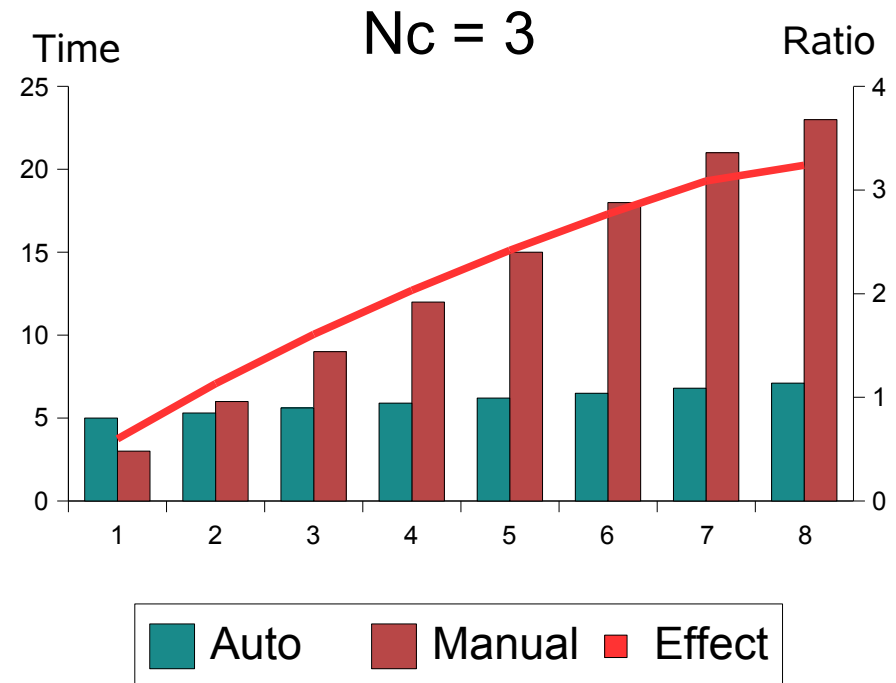
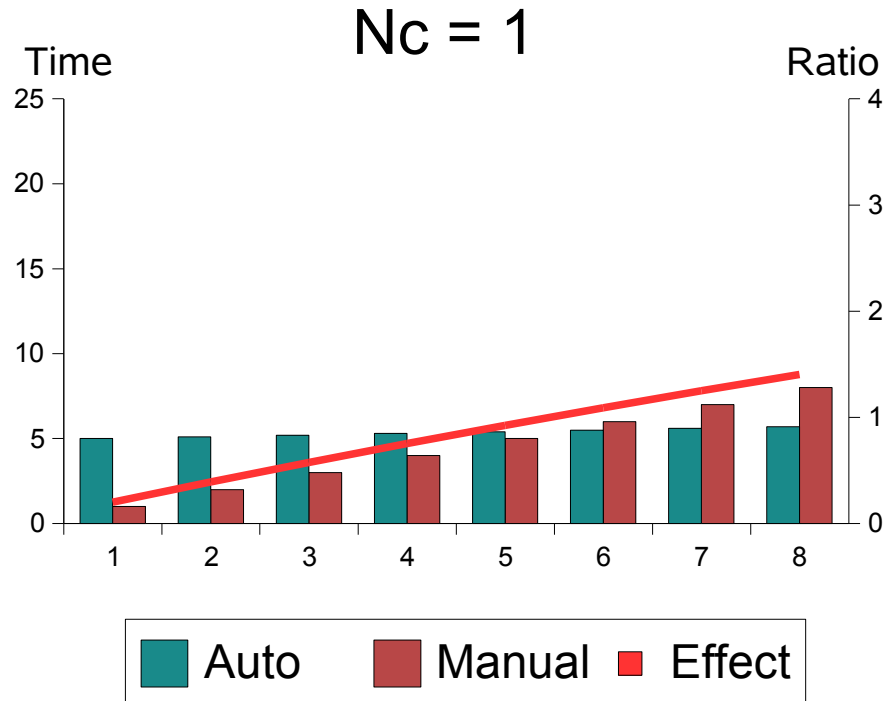
Time for ...

T_D – tests development

T_S – tests support

T_M – manual tests execution

Automation effectiveness charts



Assumptions: $T_M = 1$ engineer*week

$T_S = 0.1 * T_M$

$T_D = 5 * T_M$

$N_R = 8$

Agenda

- Who are we
- Introduction to testing
- Tools
- Automation effectiveness
- **The mantra and the two approaches**
- Misconceptions
- Conclusion

The mantra

Multiple releases and supported configurations – test support expenses is what we care about the most.

Hence, the mantra:

“No more than one change in test code for one change in product code!”

There is only one way to do so: **organize code into a library**

Two approaches to library creation

- Interface oriented

Library's provides coverage for are UI objects of the tested products such as frames, dialogs, custom components, etc.

- Concept oriented (COT)

Library covers concepts and logic of the product business model.

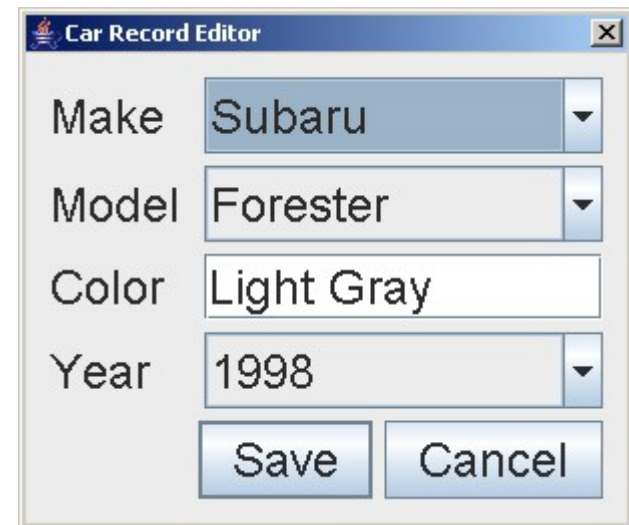
Interface oriented approach

library

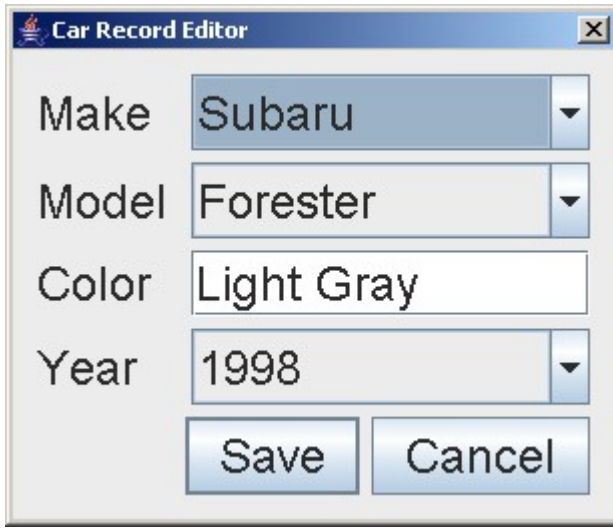
```
public class CarRecordDialogOperator {  
    public void enterColor(String color) {  
        //find the text field  
        ...  
        //type in the color  
        ...  
        //click the ok button.  
        ...  
    }  
    public void enterModel(String model) {  
        ...  
    }  
    ...  
}
```

test

```
class MyTest() {  
    public void testSetGreenColor() {  
        new CarRecordDialogOperator()  
            .enterColor("Light Gray");  
    }  
}
```



Some changes in UI



Car Record Editor

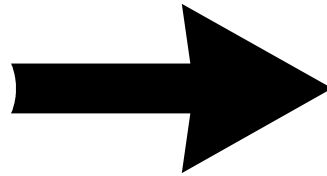
Make: Subaru

Model: Forester

Color: Light Gray

Year: 1998

Save Cancel



Car Record Editor

Make: Subaru

Model: Forester

Color: Light Gray ...

Year: 1998

Save Cancel

This is OK!
No need to change existent tests

More changes...



Car Record Editor

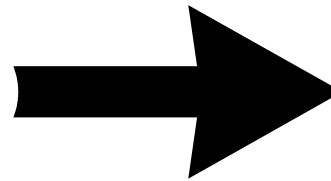
Make: Subaru

Model: Forester

Color: Light Gray

Year: 1998

Buttons: Save, Cancel



Car Record Editor

Make: Subaru

Model: Forester

Color: Light Gray

Year: 1998

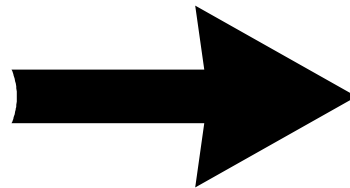
Buttons: Save, Cancel

Need to change the library? - **YES**

Need to change tests? - NO

How many changes it requires? - **Only one** (enterColor() method)

Yet more changes ...



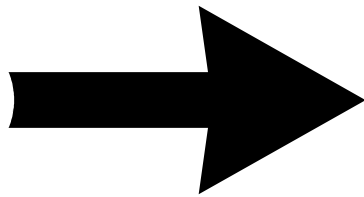
Need to change the library? - **YES**

Need to change tests? - basically, **YES**. Could be "hacked".

How many changes it requires? - **Plenty**, if not "hack"

And finally....

The same functionality implemented without “Car chooser” dialog



Model	Make	Color	Year
Subaru	Forester	Light Gray	1998

Need to change the library? - **YES**

Need to change tests? - **YES**

How many changes it requires? - **Plenty** - **EVERY** test

Interface approach: bottom line

- Good approach, but...
 - > Tests depend on UI so they are **subject to change** when UI changes
- We need another level of abstraction
 - > UI independent (next slide)

Concept oriented approach

Test

```
class MyTest() {  
    public void testSetGreenCarColor() {  
        new CarRecord().setColor(new CarColor("Green"));  
    }  
}
```

While there are **Green** cars could be entered in the system test does not need to be changed

Agenda

- Who are we
- Introduction to testing
- Tools
- Automation effectiveness
- The mantra and the two approaches
- **Misconceptions**
- Conclusion

Some misconceptions about UI automation testing

- Testing is simple
- Recording is better
- Automation frees from work
- Tests is all you need
- No manual testing anymore
- Automation goes first

“Testing is simple”

- Testing is not very different from development.
 - > Same technical complexity
 - > Own development life-cycle
 - > Highly technical types of testing
code coverage, specification coverage, static testing, etc.
 - > Outcome is just as important
- Bug tracking system maintenance
- Tools, libraries support

“Recording is better”

- Hardly (or non) maintainable tests
 - > Good coding significantly reduces maintenance time
 - > Tests are sensitive to slightest UI behavior changes
- Tests are not accurate enough
 - > Recorder is not able to recognize some actions
- Tests are created too late
 - > Only after GUI is stabilized
- Not so much of time effectiveness in test creation
 - > You could code several similar tests at once

“Automation frees from work”

Things still need to be done:

- Libraries, framework creation and maintenance
- Test creation
- Test sustaining and maintenance
- Test execution management
- Failure results analysis
- Some manual testing

“Tests is all you need”

- Test harness...
 - > to run several tests at once (test suits)
 - > to start and stop an application being tested
 - > to ensure clean initial state
 - > to provide test data
 - > to catch the faults and successes
- We use XTest with our extensions

“Tests is all you need” cont.

- Infrastructure...
 - > to schedule and execute test runs
 - on different platforms and configurations
 - > to get the latest builds
 - of application being tested
 - of tests and test framework
 - > to set up an application and its environment
 - > to collect and store results
 - > to perform results analysis
- We use Test4U and scripts

“No manual testing anymore”

- Do manual testing when...
 - > human eye is required
 - > number of runs is limited
- Automate only...
 - > tedious tests
 - > repeating tests, regression tests
 - > tests that hard to run manually

“Automation goes first”

- Don't start too early
 - > Product could significantly change
- Don't start too late
 - > Tests development takes time
 - > Time is needed to gain from the automation efforts
- What is the right moment?
 - > It depends ... (next slide).

What's the right moment

- Test framework, harness
 - > Once it is decided to go for automation
- Performance tests
 - > As early as possible
- Regular functional tests:
 - > Not before feature freeze
 - > With COT even earlier – once functional specification is ready

Agenda

- Who are we
- Introduction to testing
- Tools
- Automation effectiveness
- The mantra and the two approaches
- Misconceptions
- **Conclusion**

Quality work. A problem area

- A bottleneck
Some formal testing needed for every release.
- A pain for development
“Shoot the tester” - heard it many times.
- Most consider it boring
Lower requirement to skills of quality engineer
- Really hard to do it right
No (or almost no) scientific research around it

Automation:

Chipper!

Faster!!

More Fun!!!



Java UI Testing

Technology and real experience

Alexandre Iline,
Maria Tishkova,
Alexander Kouznetsov

Links

- NetBeans
<http://www.netbeans.org>
- Jemmy
<http://jemmy.netbeans.org>
- Jellytools
<http://jellytools.netbeans.org>
- Java GUI Testing Yahoo group