

Com S 417

Software Testing

Fall 2017 – Week 11, Lecture 19

Announcements

- Research Project.
 - You *must* register your team and topic with Rumesh. We will not approve more than one request per tool, and in some cases only one request per *class* of tool.
- Exam 2 will be Nov. 2 in class.
- Lab 4 is available and due Oct. 31.
- We *will* have 5 labs.

Topics

- Project teams and topics.
- GUI testing technology
- Profiler

GUI Test Tools

- The following slides (with white background) were prepared by S. Mitra.

Capture and Replay

- Often used for regression test development
 - Tool used to capture interactions with the system under test.
 - Inputs must be captured; outputs may also be recorded and (possibly) checked.
 - Examples:
 - Telelogic Tau: save command history in simulator
 - GUI testing tools
- Capture requires a working system to be available already!

Testing on Replay

- During the replay of events, there are various output checking possibilities:
 - Manual: user has to watch the system for anomalies
 - Complete: all outputs were recorded during capture, and system must reproduce them "exactly".
 - "Exactly": to the level of detail of the recording.
 - Checkpoints: system output is only checked at certain points for specified values.
 - These could be inserted manually or automatically.

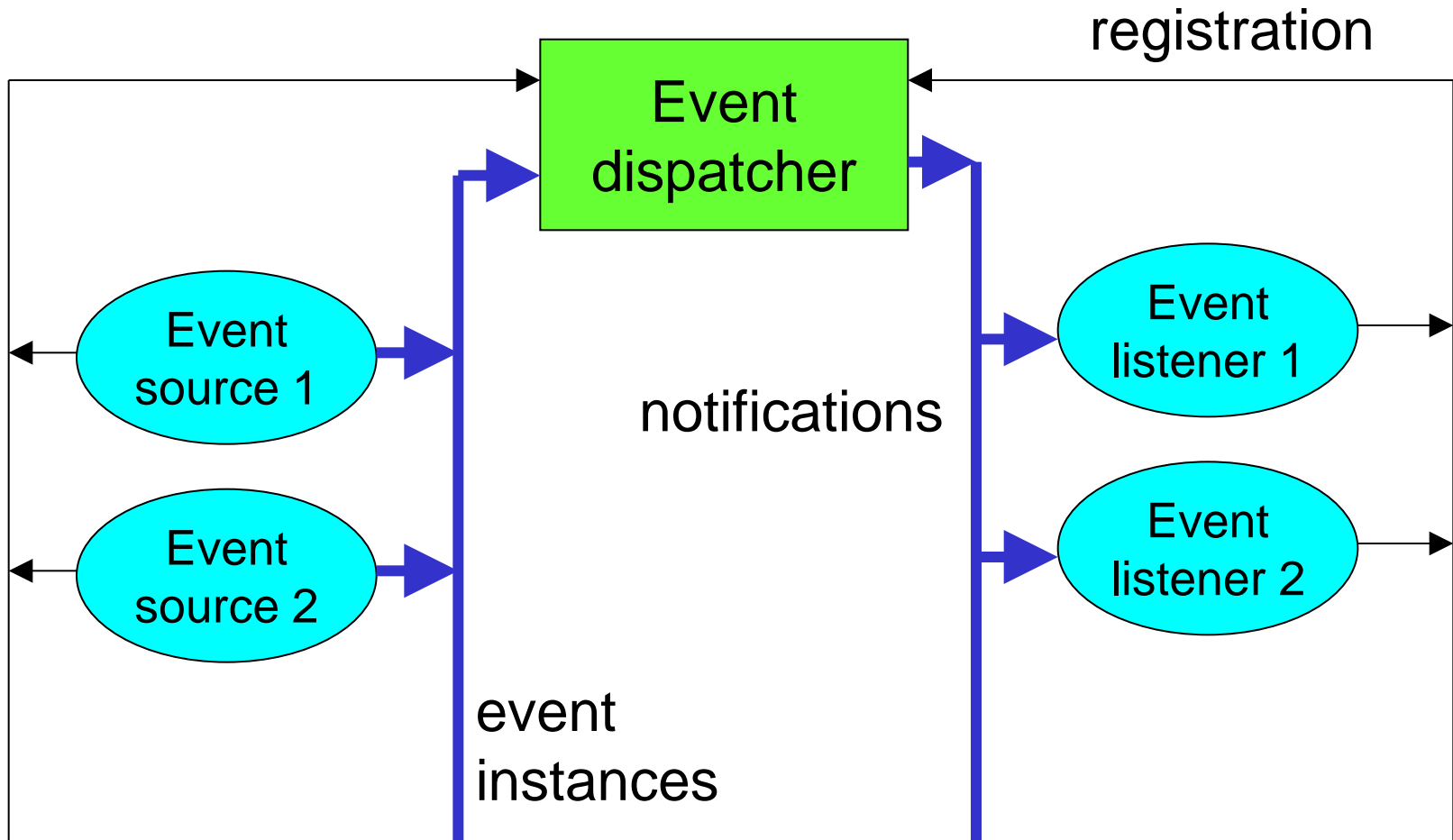
The capture record

- Inputs, outputs, and other information needed to reproduce a session with the system under test need to be recorded during the capture process.
- Examples:
 - General information: date/time of recording, etc.
 - System start-up information
 - Events from test tool to system
 - Point of control, event
 - Events from system to test tool
 - Checkpoints / expected outputs
 - Time stamps

Event-driven architecture

- Basis of GUI frameworks
 - Various controls create events when they are created, activated, modified, deactivated, or disposed.
- Input devices create events as per their functions: key pressed, key released, mouse moved, ...
- Events are sent to an event dispatcher
- "Listeners" register with the event dispatcher to receive events
 - Listeners specify the type of events in which they are interested, and which method should be called when the event occurs
- When an event is received by the event dispatcher, notifications are sent to each registered listener for the type of event.
 - Each listener has their event notification method called in turn.

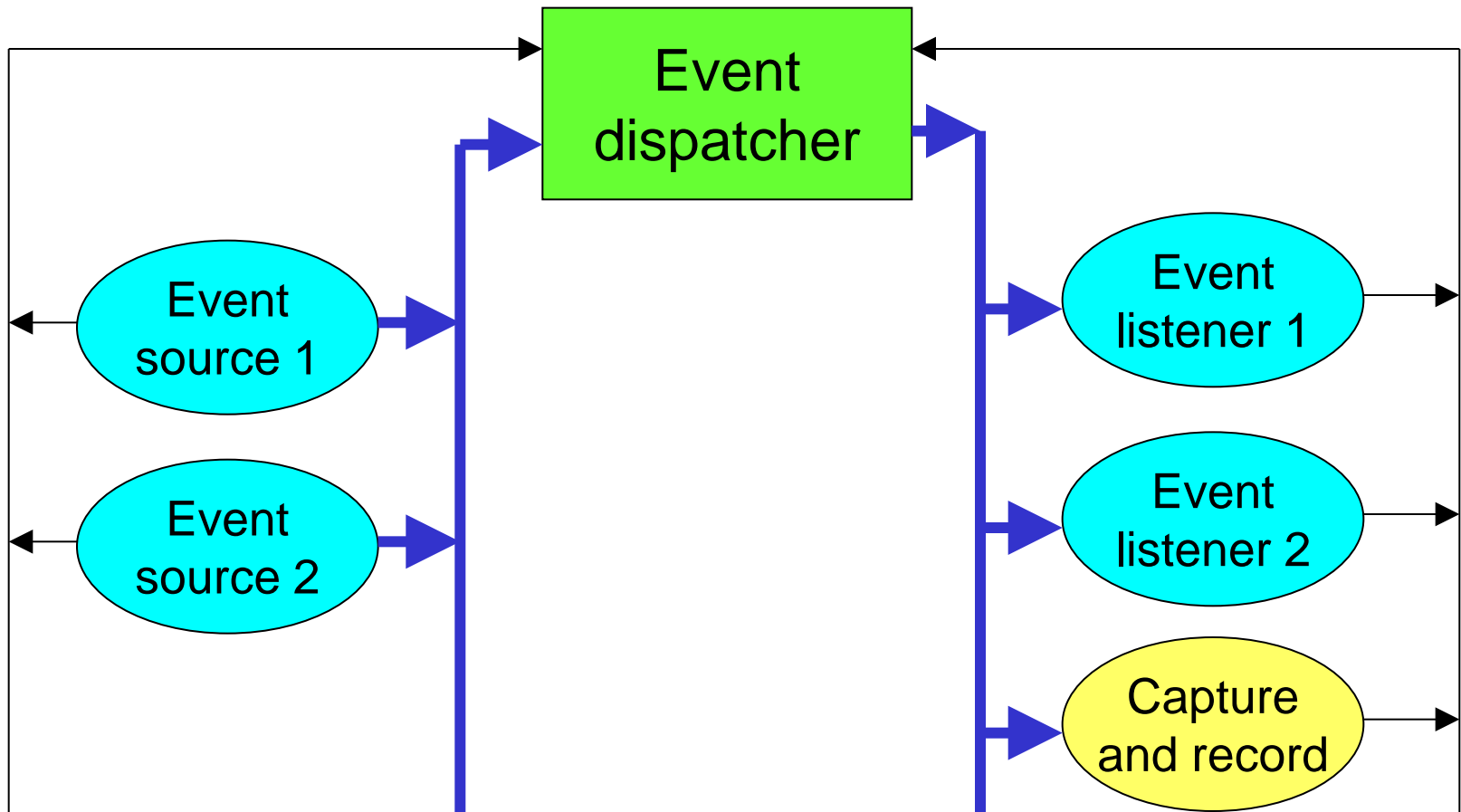
Event-driven architecture



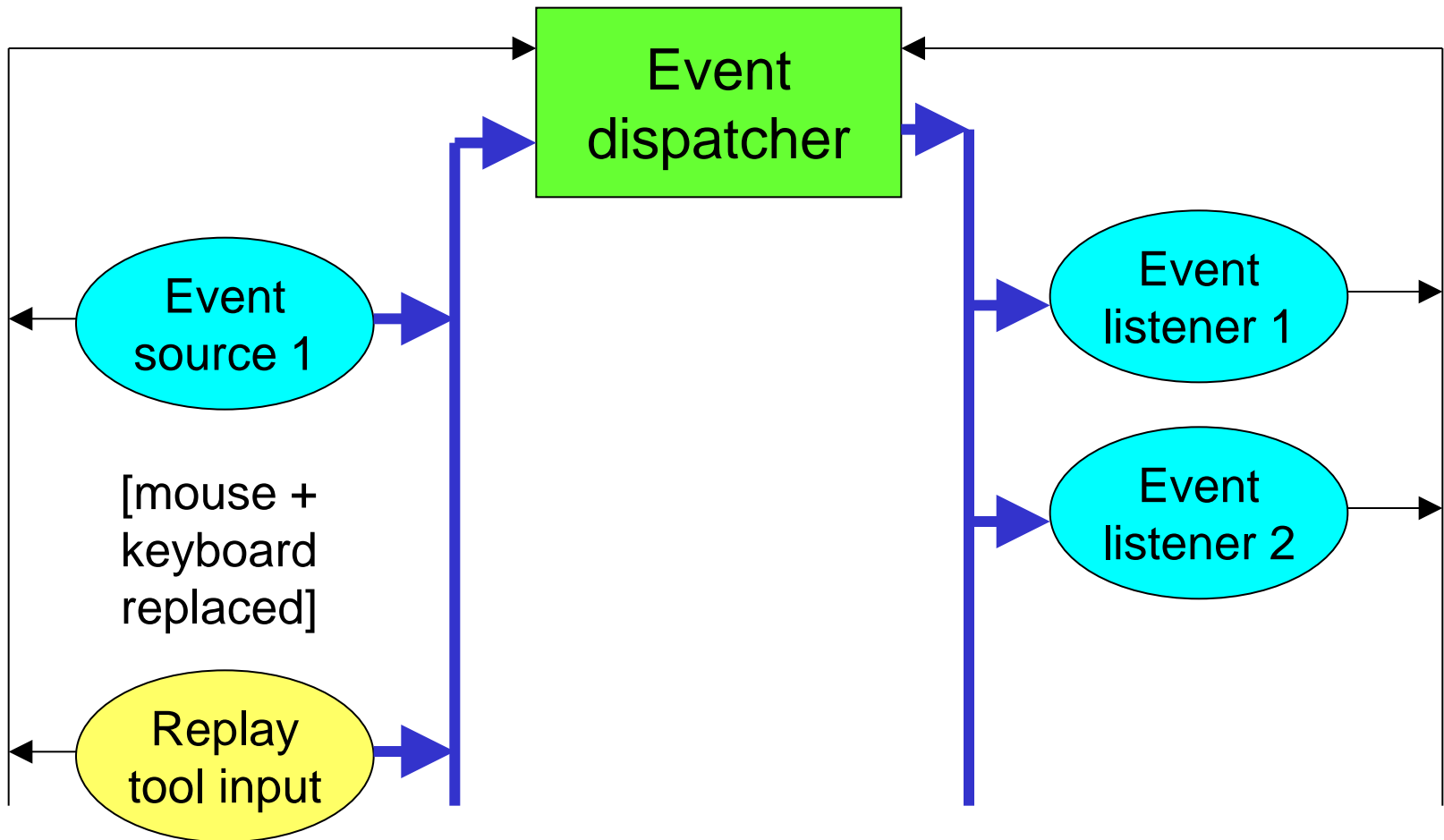
Integrating a Capture and Replay tool

- During the capture process, the tool will register as an event listener
 - Event notification method for the tool will record the details of all events that occurred.
- During the replay process, the tool will register as an event source (possibly also as a listener)
 - For mouse and keyboard events, the tool has to substitute for the actual devices as the event source.
 - Replay events should be initiated at the same relative time as during the capture.
 - Other controls issue events as usual (e.g. GUI button deactivated)

Capture



Replay



Integration Alternatives

- Use the UI event framework associated with an environment:
 - Example: Swing, SWT, etc.
- Create native system events
 - Example: The Robot class in Java will actually move the mouse cursor on the underlying platform, instead of just recreating the Java framework event that would be generated.
- Analyze screen images
 - A screen capture of a tool's GUI can be stored while the tool is running, analyzed for widgets, and then used for replay.

The Java `Robot` Class

- Class to send requests to native platform to generate system events.
- Functions
 - Keyboard: `keyPress`, `keyRelease` events
 - Mouse: `mouseMove`, `mousePress`, `mouseRelease`, `mouseWheel` events
 - Screen: request screen capture
 - Timing: delay specified duration

Tests

- Alternatives:
 - During the capture process, the tool may record events, and then compare the results during playback.
 - At test checkpoints, the tool can ask various controls for their state and record it.
 - Examples:
 - Contents of text fields
 - GUI radio button properties:
 - enabled / disabled
 - selected / not selected

Integration options

- Compile tool classes and application together
 - Similar to JUnit approach
 - User has to determine object instances that should be monitored, and then register instances of tool classes as event source and listeners
 - Connections are also set up to query controls as to their states.
 - Provides better integration, more stable replays
 - Tester has to learn how to use tool classes
- Use virtual machine registration (when there is a VM)
 - Tool has to “discover” objects of interest

Example: Jacareto

- Java-based open source GUI capture + replay tool
 - Authors: C. Spannagel et. al.
 - Web site: <http://jacareto.sourceforge.net>
- Tool sub-parts:
 - Jacareto: event record and playback
 - Picorder: command line interface
 - CleverPHL: graphical user interface (for Jacareto)

Jacareto

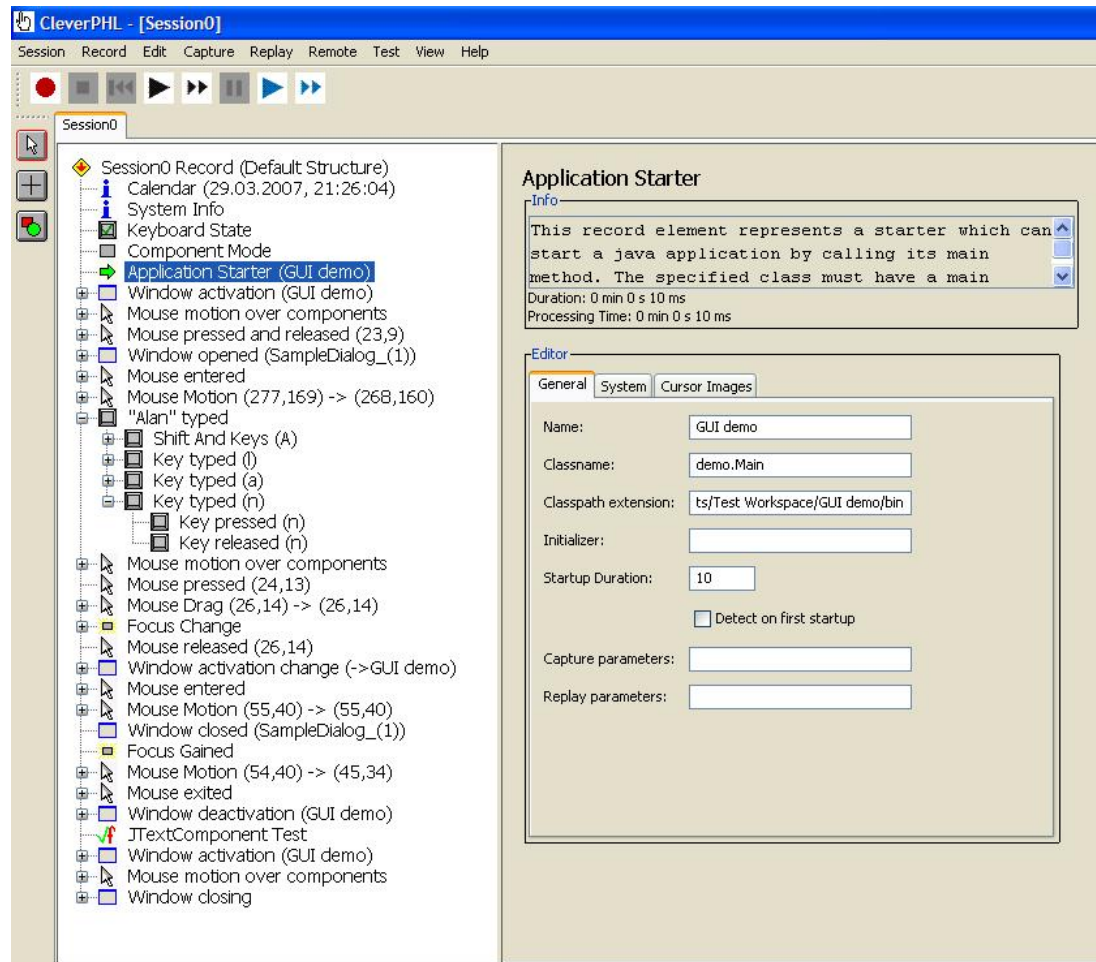
- Uses virtual machine registration
- Batch file and XML script used to start Java virtual machine, start the test tool, and load the application's classes.
 - Main method of application called by test tool as required.
- Jacareto has to discover all instances of subclasses of Component (including JComponent for Swing)

Event records

- Stored as XML
- Information recorded:
 - Type of event: mouse move, key pressed, etc.
 - Includes event-specific information: X-Y location of mouse cursor, which key was pressed, ...
 - Relative time from previous event
 - Needed to keep events synchronized
 - Example:

```
<KeyEvent procTime="0" duration="240"  
  source="SampleDialog_(1).JRootPane_(1).JLayeredPane_  
  (1).JPanel_(1).JTextField_(1)"  
  class="javax.swing.JTextField"  
  uuid="89a95465-8cae-467b-bc48-6422a63bcfc4" ID="401"  
  component="null" root="SampleDialog_(1)" xPos="0"  
  yPos="0" width="0" height="0" when="1175219983088"  
  isConsumed="false">  
  <KeyInfo keyCode="65" keyChar="A" modifiers="1" />  
</KeyEvent>
```

XML record displayed for user

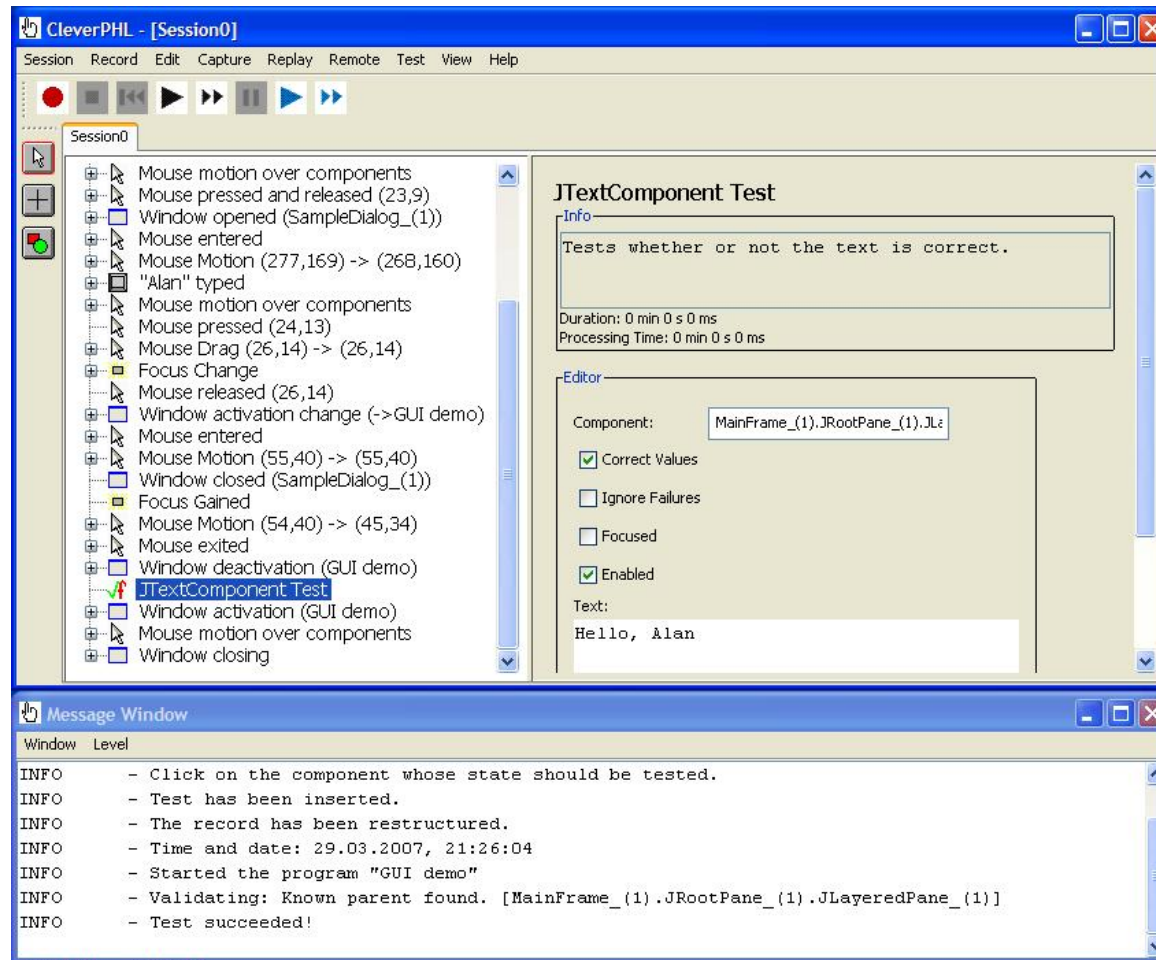


Tests

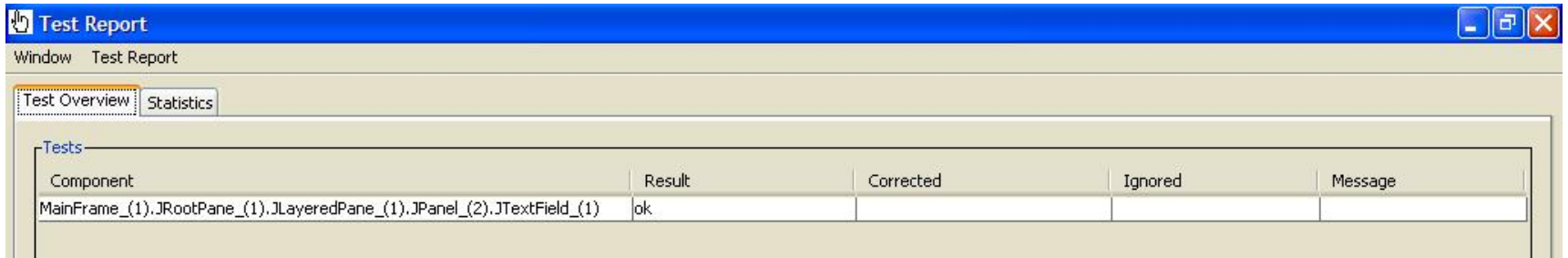
- At various user-defined checkpoints, the state of a control can be tested.
- Example: Test if a text field contains the specified text
 - Also specify that we want to stop if there is an error (i.e. ignore error is false), and that we do not want to correct the state of the control if it does not conform to the record.

```
<JTextComponentTest
component="MainFrame_(1).JRootPane_(1).JLayered
Pane_(1).JPanel_(2).JTextField_(1)"
isRegExp="false" isCorrecting= "false
isEnabled="true" isIgnoring="false"
hasFocus="false">Hello, Alan
</JTextComponentTest>
```

UI for Test creation



Test reports



The image shows a screenshot of a 'Test Report' window. The window has a blue title bar with the text 'Test Report' and standard window controls. Below the title bar is a menu bar with 'Window' and 'Test Report'. There are two tabs: 'Test Overview' (selected) and 'Statistics'. The main area is titled 'Tests' and contains a table with the following data:

Component	Result	Corrected	Ignored	Message
MainFrame_(1).JRootPane_(1).JLayeredPane_(1).JPanel_(2).JTextField_(1)	ok			



The image shows a screenshot of the 'Test Report' window with the 'Statistics' tab selected. The statistics are as follows:

Successes:	1
Failures:	0
Corrections:	0
Ignored Failures:	0

Example: Marathon

- Capture and playback tool for testing Java/Swing graphical user interfaces.
 - www.marathontesting.com
- Uses the “Jython” (Python for Java) scripting language to store test cases.
- Test cases can be captured interactively, or written as Jython scripts.
- Assertions can be added to check the state of various components.
- Uses test runner that is a non-standard version of JUnit.

Sample Jython test script

```
useFixture(default)

def test():

    java_recorded_version = '1.5.0_14'

    if window('Greetings'):

        select('Click Me', 'true')

        if window('Enter name'):

            select('TextField', 'Alan')

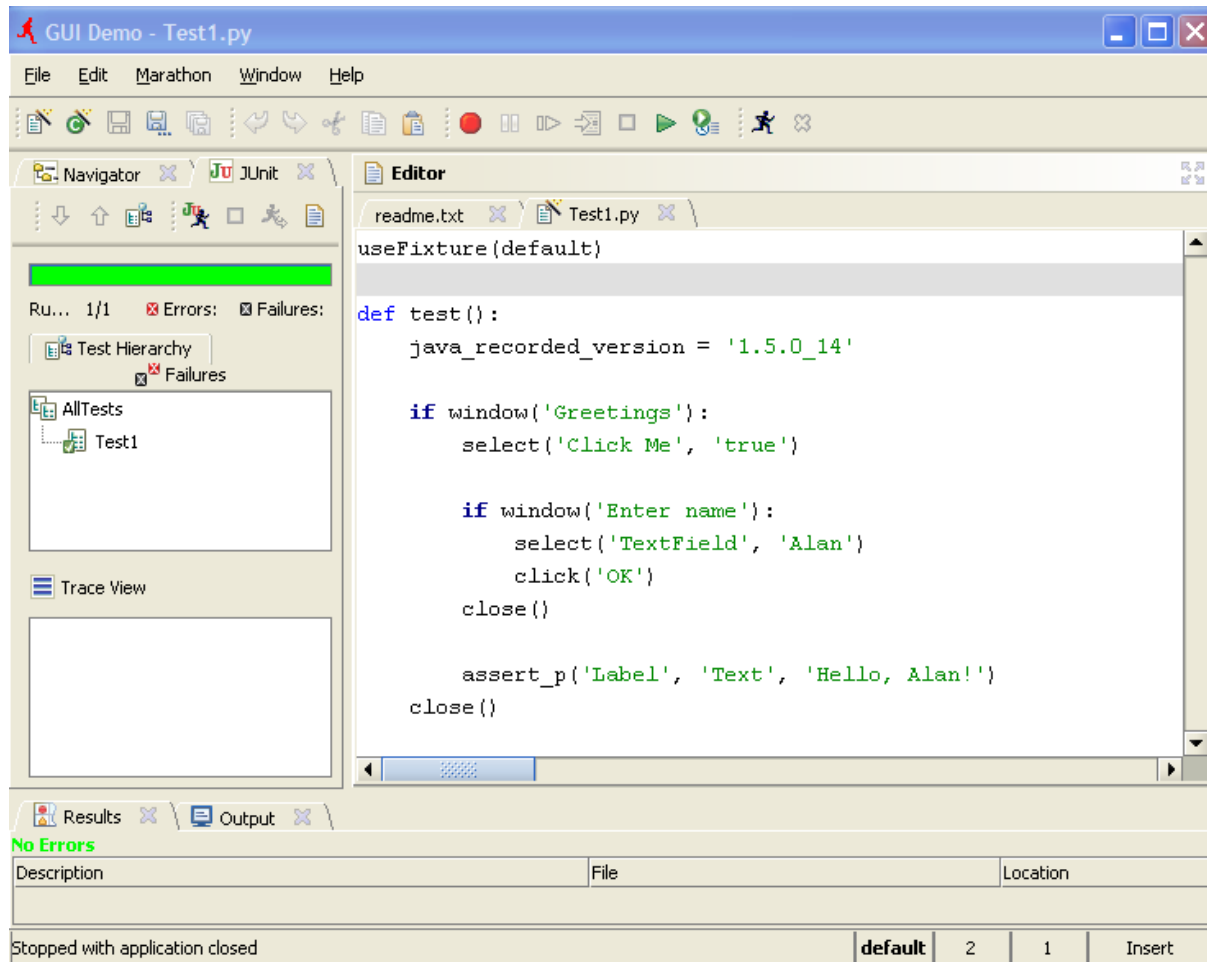
            click('OK')

        close()

    assert_p('Label', 'Text', 'Hello, Alan!')

close()
```

Running a Marathon test



Test report

Marathon Acceptance Test Results - Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///C:/Documents%20and%20Settings/Alan/Local%20Settings/Temp/marathon36854.html

GUIDemo - Results

Generated on **Thursday Mar 29 23:28:33 2007**

Summary

Tests	Failures	Errors	Success rate	Time
1	0	0	100.00%	4.947

Packages

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

Name	Tests	Failures	Errors	Time(s)
AllTests	1	0	0	4.947

Test AllTests

Name	Status	Type	Time(s)
Test1	Success		4.947

GUI Tech and Web-Testing

- The Browser is a local application. The recorder/playback mechanism control it.
- Instead of “registering” with the JVM, the test tool connects to the browser via an application programming interface such as NSAPI, Web Extensions API, or the Microsoft COM interface.
- In some cases, the tool may require a special version of the browser.
- Selenium Web-Driver is an example of a purpose specific API built on top of the extension api.

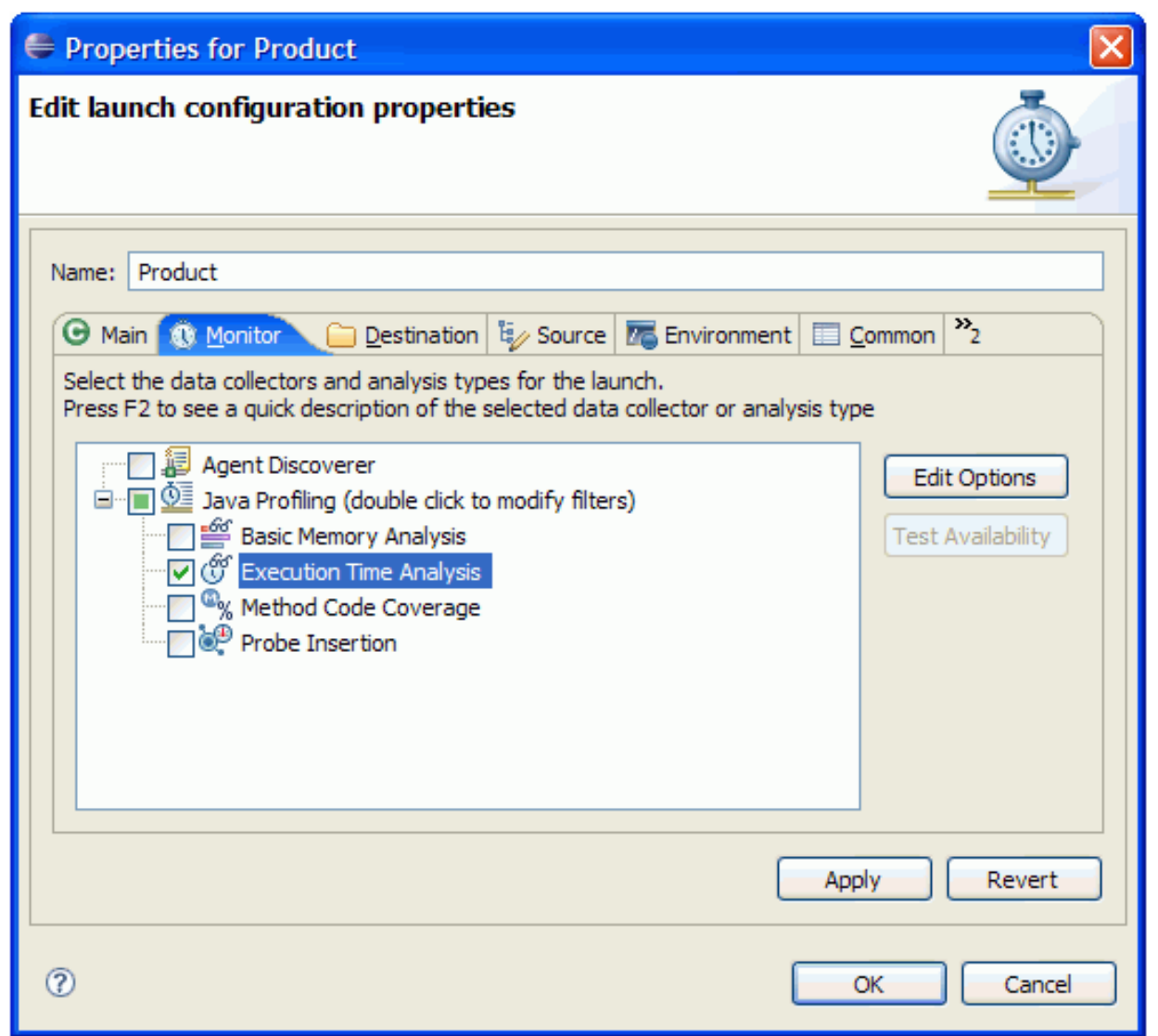
Profilers

- Profilers are used to monitor run-time usage of machine resources, especially CPU usage.
- Profilers are an example of externally added, general purpose test instrumentation.
- Depending on the tool and configuration, profilers can count instruction executions, measure time consumed in different methods and functions, count objects created and destroyed, and report memory usage by different categories (stack, heap, code, etc.)
- Java profilers tend to use a special JVM interface designed for that purpose. In non JVM environments, the profiler may “pre-compile” code to insert instrumentation hooks, or supply a special run-time library.

Profilers and testing

- Profilers are important for evaluating Non-functional requirements:
 - Can the app run reliably with certain memory limits?
 - Do critical network operations (for example) complete in a timely way?
- Profilers are extremely useful for troubleshooting performance problems and memory leaks.
- However, most profilers impose a performance overhead.

TPTP: an eclipse profiler plugin



Configuring what part of the app is to be monitored.

- Helps reduce instrumentation overhead.
- Makes reports easier to use.

Edit Profiling Set

Filter Set
Select and define the filter set that will be used for profiling.

Select a filter set:

- ☐ Default
- ☒ ProductFilterSet
- ☐ WebSphere J2EE
- ☐ WebSphere Studio

Buttons: Add..., Rename..., Remove

Contents of selected filter set:

Class	Method Name	Rule
com.sample.product*	*	INCLUDE
*	*	EXCLUDE

Buttons: Add..., Edit..., Remove, Up, Down

Note: By default, all classes are profiled.
Use a filter set to filter out and customize profiled data.
Supported wildcard usage on a filter is: '*' , '<pattern>*', '*<pattern>'.

Buttons: < Back, Next >, Finish, Cancel

Execution Statistics

Profiling and Logging - ProductCatalog.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Working Sets

Console Execution Statistics Method Invocation Details

Execution Statistics - com.sample.product.Product at popescu011 [PID: 240] (Filter: No filter)

Method	Class	Package	Base Time (sec...)	<Cumulative Ti...	Calls
main(java.lang.String[]) void	Product	com.sample.p...	0.05%	58.60%	0.02%
readData(java.lang.String) void	ProductCatalog	com.sample.p...	0.04%	58.13%	0.02%
createParser() javax.xml.parsers	ProductCatalog	com.sample.p...	0.07%	42.96%	0.46%
newSAXParser() javax.xml.parse	SAXParserFa...	org.apache.x...	0.07%	21.88%	0.46%
SAXParserImpl(javax.xml.parsers	SAXParserImpl	org.apache.x...	0.43%	21.81%	0.46%
newInstance() javax.xml.parsers	SAXParserFa...	javax.xml.pa...	0.07%	20.99%	0.46%
find(java.lang.String, java.lang.S	FactoryFinder	javax.xml.pa...	0.50%	20.87%	0.46%
SAXParser()	SAXParser	org.apache.x...	20.49%	20.49%	0.46%
findJarServiceProvider(java.lang.	FactoryFinder	javax.xml.pa...	18.88%	18.88%	0.46%
parseContent(java.io.File, javax.	ProductCatalog	com.sample.p...	0.07%	14.86%	0.46%
parse(java.io.InputStream, org.x	SAXParser	javax.xml.pa...	0.06%	13.03%	0.46%
parse(org.xml.sax.InputSource, i	SAXParser	javax.xml.pa...	0.12%	12.93%	0.46%
parse(org.xml.sax.InputSource)	AbstractSAXP...	org.apache.x...	9.36%	12.14%	0.46%
startElement(java.lang.String, ja	ProductCatalog	com.sample.p...	0.90%	2.78%	0.93%
FileInputStream(java.io.File)	FileInputStream	java.io	0.09%	1.70%	0.46%
open(java.lang.String) void	FileInputStream	java.io	1.57%	1.57%	0.46%
newInstance(java.lang.String, ja	FactoryFinder	javax.xml.pa...	1.02%	1.02%	0.46%

18M of 40M

Drill-Down view

Profiling and Logging - ProductCatalog.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Working Sets

Console Execution Statistics **Method Invocation Details**

Method Invocation Details: com.sample.product.util.ProductCatalog.createParser() javax.xml.parsers.SAXParser

Method Invocation Details

Selected method

>Calls	Method	Class	Package	Base Time (sec...)	Cumulative Tim...
24	createParser() javax.xml.parsers.SA...	ProductCatalog	com.sample.p...	0.000580	0.378272

Selected method is invoked by:

>Invoked by	Method	Class	Package	Base Time (sec...)	Cumulative Tim...	Calls
1	readData(java.lang.String) void	ProductCatalog	com.sample.p...	0.000391	0.511898	1

Selected method invokes:

Invokes	Method	Class	Package	Base Time (sec...)	<Cumulative Ti...	Calls
24	newSAXParser() javax.xml.parsers.S...	SAXParserFac...	org.apache.x...	0.000602	0.192656	24
24	newInstance() javax.xml.parsers.SA...	SAXParserFac...	javax.xml.par...	0.000652	0.184867	24
4	loadClass(java.lang.String) java.lang....	ClassLoader	java.lang	0.000034	0.002085	4
7	checkPackageAccess(java.lang.String...	ClassLoader	java.lang	0.000044	0.000046	7
24	setValidating(boolean) void	SAXParserFac...	javax.xml.par...	0.000005	0.000005	24

18M of 40M

Research Project

- Teams of 4 to 5 (slots for 16 to 20 teams).
- Written report.
- 15 minute Presentation (all team members have speaking role).
- Attendance is required for everyone during team presentations: Nov 16, 28, 30, Dec 5, and Dec 7.
- Teams and topics must be submitted ASAP. Your proposal must be approved Oct 31. Submit to Rumesh.
- First team to propose a topic gets it, so act quickly.
- Emphasis is on new or advanced tools and on recent research.

Suggested topics

- New and or widely used tools for
 - Generating combinatorial testing covering arrays.
 - Mutation testing.
 - Performance testing (application scope, e.g. TPTP)
 - Performance testing (web scope)
 - Test and Defect management (especially test to requirements traceability).
 - Test Instrumentation (especially non-interfering)
 - mediation systems
 - Test automation (especially GUI-related)
 - guitar
 - selenium and products built on-top of selenium web-driver
 - capture/playback products.

Suggested Topics (tools cont'd)

- Slicing tools for testing and debugging
- Reliability testing
- Usability testing
- Integration testing (spring integration?)
- Real-time and Concurrency Testing (GroboUtils and related)
- Advanced static analysis tools.
- Behavior Driven Testing (BDT tools)
- Mobile Application Testing
- Security Testing
- Product line testing

Suggested topics (cont'd)

- Recent research
 - on combinatorial testing.
 - on mutation testing.
 - on algorithmic program debugging.
 - on slicing
 - on usability testing
 - on security testing
 - on test standardization
 - empirical evaluation of agile test quality
 - on testing big-data applications

For Ideas:

- Antonia Bertolino, "Software Testing Research: Achievements, Challenges, Dreams"
- query "software _____ testing tools" on wikipedia.
- query "glenford myers" on google scholar and then limit results to those newer than 2016.
 - looks for recent papers that reference a seminal testing work.
- query "Offutt" + "software testing" on google scholar and then limit results to those newer than 2016.
 - Similar queries based on authors who have contributed to a field that interests you. (Check the bibliographic notes at the end of a related chapter in Ammann & Offutt or in Mathur.

Background Reading

HTTP: the web protocol

- Basics of HTTP messages:
 - https://www.tutorialspoint.com/http/http_messages.htm
 - https://www.tutorialspoint.com/http/http_requests.htm
 - https://www.tutorialspoint.com/http/http_responses.htm
 - https://www.tutorialspoint.com/http/http_methods.htm
 - https://www.tutorialspoint.com/http/http_status_codes.htm
 - https://www.tutorialspoint.com/http/http_url_encoding.htm
- Examples & network level tools
 - https://wiki.wireshark.org/Hyper_Text_Transfer_Protocol
 - <https://tools.ietf.org/html/rfc2616>