Software Testing

Lesson 11
Test Tools
V1.1

Uwe Gühl

Winter 2013 / 2014

Contents (1/2)

- Tool Support for Testing
  - Terms
  - Types of Test Tools
    - Tool Support for Testing
    - Test Tool Classification
    - Tool Support for Management of Testing and Tests
    - Tool Support for Static Testing
    - Tool Support for Test Specification
    - Tool Support for Test Execution and Logging
    - Tool Support for Performance and Monitoring
    - Tool Support for Specific Testing Needs

Contents (2/2)

- Tool Support for Testing (… cont'd)
  - Test Driven Development with JUnit
  - Effective Use of Tools: Potential Benefits and Risks
  - Introducing a Tool into an Organization

Terms

Meanings of the term “test frameworks”:

1. Reusable and extensible testing libraries that can be used to build testing tools (synonym: “test harnesses”)
2. A type of design of test automation, like
   - data-driven,
   - keyword-driven

ISTQB does NOT use the alternative meaning “Overall process of execution of testing”.

Types of Test Tools
Tool Support for Testing

- Test tools support testing activities:
  1. Tools directly used in testing
     For example test execution tools, test data generation tools, or result comparison tools.
  2. Tools helping in test management
     These support managing of tests, test results, test data, requirements, incidents, and reporting and monitoring test execution
  3. Tools supporting exploration testing
     For example tools that monitor file activity for an application
  4. Any tool that aids in testing
     In this context a spreadsheet is also a test tool

Types of Test Tools
Tool Support for Testing

Depending on context

- Purposes of tool support for testing:
  - Improve the efficiency of test activities by
    - automating repetitive tasks,
    - supporting manual test activities like test planning, test design, test reporting and monitoring.
  - Automate activities that
    - require significant resources when done manually, e.g. static testing,
    - cannot be executed manually like large scale performance testing of client-server applications.
  - Increase reliability of testing, e.g. by automating large data comparisons or simulating behaviour.

Types of Test Tools
Test Tool Classification

- Different tools support different aspects of testing – consider even every little tiny tool to help to make testing life easier
- ISTQB classifies tools according to the testing activities that they support.
- Other possible classification criteria:
  - Purpose,
  - commercial, free, open-source, or shareware … see for example opensourcetesting.org [Ope14]
  - technology used.

Types of Test Tools
Test Tool Classification

- Clear mapping is not always possible:
  - Some tools clearly support one activity
  - Other tools support more than one activity, for example test management tools
    - often include a requirements module and defect management.
    - offer interfaces to a test automation or load testing tool.
  - Tools from a single provider may be bundled into one package.
- ISTQB classifies such tools under the activity with which they are most closely associated.

Types of Test Tools
Test Tool Classification

- Intrusive test tools
  - can affect the actual outcome of the test, e.g.
    - The actual timing may be different due to the extra instructions that are executed by the tool.
    - Different measure of code coverage.
  - ⇨ Known as “probe effect”.
- Some test tools offer support more appropriate for developers, typically used during
  - component testing,
  - component integration testing.

Types of Test Tools

Tool Support for Management of Testing and Tests

Test Management Tools

- support
  - quantitative analysis,
  - reporting of the test objects,
  - tracing the test objects to requirement specifications.
- provide interfaces for
  - managing requirements,
  - executing tests,
  - tracking defects.
- might include version control or offer an interface.

Types of Test Tools

Tool Support for Management of Testing and Tests

Test Management Tools

- Commercial solutions typically offer a basic solution with module extending testing capabilities
  - Test management
    Integration of requirements management tool, defect management, special functionalities
  - Test automation
    Integration of capture / replay tools
  - Load and performance testing
    Integration of load and performance test modules
  - More … e.g. basic security tests

Types of Test Tools
Tool Support for Management of Testing and Tests

Test Management Tools – Selections

- Commercial tools
  - HP QC Quality Center,
    HP ALM Application Lifecycle Management [HP14],
  - Rational Quality Manager by IBM [IBM14],
  - Silk SilkCentral Test Manager [Bor14],
  - Visual Studio [Vis14].
    Microsoft IDE with test support.
- Open Source / Free Tools
  - Overview opensourcetesting.org [Ope14],
  - TestLink [TI14],
  - XStudio [XQ14].

Types of Test Tools

Tool Support for Management of Testing and Tests

- Which commercial tool to use? [SG08]:
  "Forrester evaluated 6 functional testing solutions — tool suites with support for manual testing, test automation, and test management — across 96 criteria.
  HP's* leadership of the functional testing market continues unabated since its 2006 acquisition of Mercury Interactive.
  IBM has expanded its solution's support for packaged applications, and its road map for the future looks promising indeed. **Borland Software** and **Compuware** have doubled down on serving their target users: more and less technical testers, respectively.
  Empirix and **Seapine Software** offer less-costly but also less-capable solutions, with notably limited support for applications and technologies ..."

* Products **HP QC Quality Center, HP ALM Application Lifecycle Management**

Types of Test Tools

Tool Support for Management of Testing and Tests

Requirements Management Tools

- manage requirements with attributes like priority,
- support tracing requirements to tests,
- may help with identifying inconsistent or missing requirements.

Types of Test Tools

Tool Support for Management of Testing and Tests

Incident / Defect Management Tools

- store and manage incident / bug reports, e.g.
  - defects,
  - failures,
  - change requests,
  - support issues.
- help in managing the bug life cycle, optionally with support for statistical analysis.

Types of Test Tools

Tool Support for Management of Testing and Tests

Configuration Management Tools

- not strictly test tools.
- necessary for storage and version management of testware and related software.
- important if there are different hardware / software environments concerning
  - operating system versions,
  - compilers,
  - browsers.
- Comparison available [Wik14].

Types of Test Tools
Tool Support for Static Testing

Static testing tools

- make it possible to find more defects early in the development process.
  ⇨ saving costs.
- help developers and testers find defects prior to dynamic testing.

Types of Test Tools
Tool Support for Static Testing

Review Tools

- Used to
  - store and communicate review comments,
  - report on defects,
  - report on effort.
- Support with
  - review processes,
  - check lists,
  - review guidelines,
  - online reviews for large or geographically dispersed teams.

Types of Test Tools
Tool Support for Static Testing

Static Analysis Tools

- help to find defects by
  - providing support for enforcing coding standards – including secure coding,
  - analysis of structures and dependencies.
- can help in planning or risk analysis by providing metrics for the code like complexity.

Types of Test Tools
Tool Support for Static Testing

… for developers

Static Analysis Tools

- **Special considerations**
  - Static analysis tools can enforce coding standards,
  - Lot of rework possible, if applied to existing code
    → Discuss: High quantity of warning messages
    - do not stop the code from being translated into an executable program.
    - should be addressed to reduce effort for maintenance of the code in future.
  - Idea: Gradual implementation of the analysis tool with initial filters to exclude some messages.

Types of Test Tools
Tool Support for Static Testing

… for developers

Modelling Tools

- used to validate software models by
  - enumerating inconsistencies,
  - finding defects.

  Example: Validation of a physical data model for a relational database.
- may generate test cases based on the model.

Types of Test Tools
Tool Support for Test Specification

Test Design Tools

- to generate
  - test inputs,
  - executable tests,
  - test oracles.

  based on
  - requirements,
  - graphical user interfaces,
  - design models (state, data or object),
  - code.

Types of Test Tools
Tool Support for Test Specification

Test Data Preparation Tools

- help to set up test data
  These could be used during the execution of tests to ensure security with data anonymity.
- therefore manipulate
  - databases,
  - files, or
  - data transmissions.

Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools

- enable tests to be executed automatically, or semi-automatically.
- Areas: Regression test, smoke test, setup tests, configuration tests, non-GUI tests (interfaces).
- use scripting language(s) or GUI-based configuration e. g. to parametrize data.
  ⇨ Technical expertise required.
- use stored inputs and expected outcomes.
- usually provide a test log for each test run.

Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools

- often require significant effort
  in order to achieve significant benefits.
- Classical approach: Capture & replay
  - A captured script is
    - a linear representation with specific data and actions as part of each script,
    - might be unstable when unexpected events occur.
  - Does not scale to large numbers of automated test scripts.

Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools

- Data-driven testing approach
  - separates out the test inputs (the data), usually into a spreadsheet,
  - uses a more generic test script that can
    - read the input data,
    - execute the same test script with different data.
  - Testers can then create the test data for these predefined scripts.
  - Instead of defined data in a spreadsheet, data could be generated by an algorithm / configuration as well.

Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools

- Keyword-driven testing approach
  - a spreadsheet contains
    - keywords describing the actions to be taken, and
    - test data.
  - Testers can then define tests using the keywords, which can be tailored to the software under test.

Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools

Automated tests should be [MSA03] (1/2):

- Concise – As simple as possible and no simpler.
- Self Checking – Test reports its own results; needs no human interpretation.
- Repeatable – Test can be run many times in a row without human intervention.
- Robust – Test produces always same result. Tests are not affected by changes in the external environment.
- Sufficient – Tests verify all the requirements of the software being tested.
- Necessary – Everything in each test contributes to the specification of desired behaviour.

Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools

Automated tests should be [MSA03] (2/2):

- Clear – Every statement is easy to understand.
- Efficient – Tests run in a reasonable amount of time.
- Specific – Each test failure points to a specific piece of broken functionality;
  unit test failures provide “defect triangulation”.
- Independent – Each test can be run by itself or in a suite with an arbitrary set of other tests in any order.
- Maintainable – Tests should be easy to understand and modify and extend.
- Traceable – To and from the code it tests and
  to and from the requirements.

Types of Test Tools
Tool Support for Test Execution and Logging

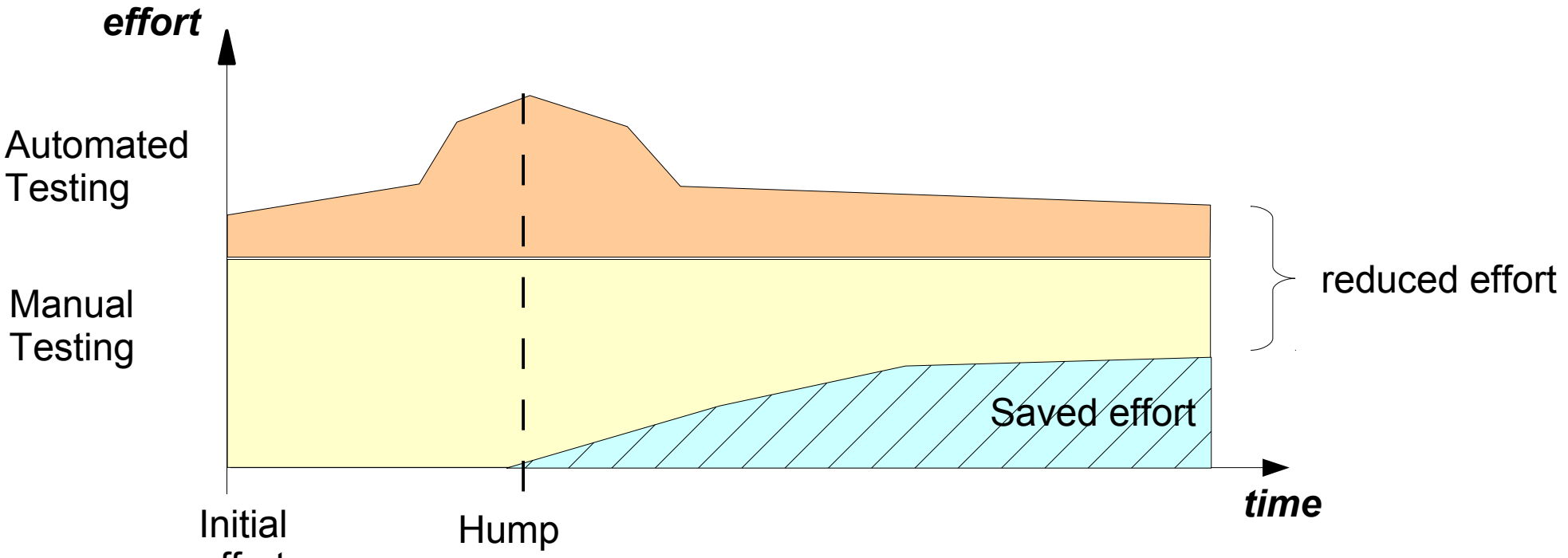
Test Execution Tools – Considerations

- **Benefits**
  - Could save costs in reducing manual test effort.
  - Useful for regression tests and large number of similar test with different data sets, environmental parameters.
  - Already defined tests could be executed fast, time independent, for example during night.
  - Could increase trust into software under test with regular repetitive automated test execution.

Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools – Considerations

- **Benefits**
  - The cost of automation is offset by the savings received from automation [Mes11].



Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools – Considerations

- **Risks**
  - Need initial investment:
    Experts, tool investment costs.
  - Need configuration, maintenance.
  - Introduce new possible defect sources.
  - Tool specific
    - Proprietary scripting languages,
    - Access to GUI elements directly or via position,
    - Possibility to skip GUI steps.

Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools – Selections

- Commercial tools
  - HP UFT Unified Functional Testing [HP14a] (was HP QTP (Quick Test Professional)),
  - Rational Test Workbench by IBM [IBM14a],
  - Silk Test™ [Bor14a].

Types of Test Tools
Tool Support for Test Execution and Logging

Test Execution Tools – Selections

- Open source tools overview [Ope14].
- Open source tools for testing web applications:
  - Canoo webtest [Can14],
  - Selenium [Sel14],
  - A comparison between Selenium and Canoo webtest [Gui07],
  - Siege [Dog12],
  - Watir [Wat14].

Types of Test Tools
Tool Support for Test Execution and Logging

Test Harness /
Unit Test Framework Tools

- What? Facilitates the testing of
  - components or
  - parts of a system.
- How?
  - simulate the environment in which that test object will run,
  - use of mock objects as stubs or drivers.

Types of Test Tools
Tool Support for Test Execution and Logging

Test Harness /
Unit Test Framework Tools

- Tool example for Continuous Integration: Jenkins [Jen14].
- Tool examples for Java Unit Tests:
  - JUnit [Jun14],
  - TestNG [Tes14].

Types of Test Tools
Tool Support for Test Execution and Logging

Test Comparators

- determine differences between
  - files,
  - databases, or
  - test results.
- may use a test oracle, especially if it is automated,
- typically parts of test execution tools.

Types of Test Tools

Tool Support for Test Execution and Logging

Coverage Measurement Tools

- could be intrusive or non-intrusive,
- measure the percentage of specific types of code structures that have been exercised by a set of tests, for example
  - statements,
  - branches or decisions,
  - module or function calls.

Types of Test Tools
Tool Support for Test Execution and Logging

Security Testing Tools

- evaluate the security characteristics of software.
- evaluate the ability of the software to protect
  - data confidentiality,
  - integrity,
  - authentication,
  - authorization,
  - availability, and
  - non-repudiation.
- focus often on defined technology, platform, and purpose.

Types of Test Tools

Tool Support for Performance and Monitoring

Dynamic Analysis Tools

- find defects during software execution, such as
  - time dependencies or
  - memory leaks.
- typically used in
  - component testing,
  - component integration testing, and
  - testing middleware.

Types of Test Tools
Tool Support for Performance and Monitoring

Load and Performance Testing Tools

- Performance testing measures how quickly a system responds under various workloads:
  Given load X, how fast will the system return a result Y?
- Performance testing tools monitor and report on how a system behaves under a variety of simulated usage conditions in terms of
  - number of concurrent users,
  - their ramp-up pattern,
  - frequency, and
  - relative percentage of transactions.

Types of Test Tools
Tool Support for Performance and Monitoring

Load and Performance Testing Tools

- Load Test
  - Determines a system's behaviour under various (high) workloads.
  - Given a certain load, how will the system behave?
- Load is simulated by virtual users.
- Virtual users
  - carry out a selected set of transactions,
  - spread across various test machines commonly known as load generators.

Types of Test Tools
Tool Support for Performance and Monitoring

Load and Performance Testing Tools

- Stress Test
  - A test that increases the workload on a system until the system fails.
  - Under what load will the system fail and how does it fail?

Types of Test Tools
Tool Support for Test Execution and Logging

Load and Performance Testing Tools – Selections

- Commercial tools
  - HP LoadRunner [HP14b],
  - Rational Performance Test Server by IBM [IBM14b],
  - Silk Performer [Bor14b].
- Open source tools could be used as well
  - Overview [Ope14],
  - Presentation about usage of open source performance testing tools at google [Bjo06],
  - Apache JMeter [Apa14].

Types of Test Tools
Tool Support for Performance and Monitoring

Monitoring Tools

- continuously focus on specific system resources, they
  - analyze,
  - verify and
  - report on usage.
- give warnings of possible service problems.

Types of Test Tools
Tool Support for Specific Testing Needs

Data Quality Assessment

- There are projects focusing on data like
  - data conversion projects,
  - migration projects,
  - data warehouse applications.
- Tools are requested for data quality assessment to ensure that processed data is
  - correct,
  - complete and
  - complies with a context-specific standard.

Types of Test Tools
Tool Support for Specific Testing Needs

Usability Testing Tools

- To support usability testing, there exist several usability testing tools [Cha10], [Tom09].
- These tools support e.g. in using usability evaluation methods.
- Main usage:
  Conduct tests and attempt to identify problem areas on websites.

TDD with JUnit

- Test Driven Development (TDD)
  is a software development process that relies on the repetition of a very short development cycle [Wik14a]:
  - First write an (initially failing) automated test case that defines a desired improvement or new function.
  - Second produce the minimum amount of code to pass that test.
  - Finally refactor the new code to acceptable standards.

TDD with JUnit

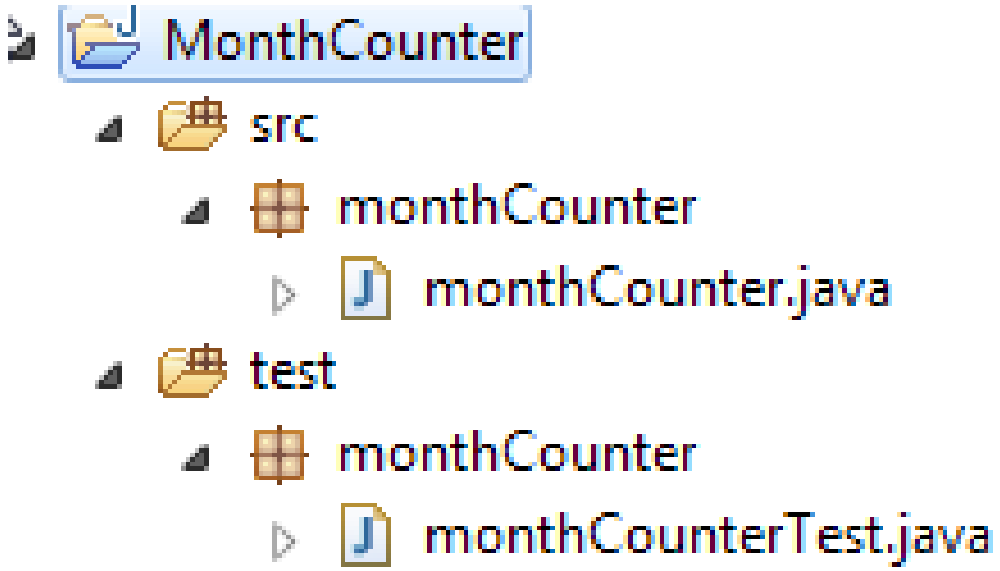
- JUnit [Jun14]
  - introduced by Erich Gamma and Kent Beck.
  - slim unit testing framework for the Java programming language.
  - to write automated repeatable unit tests
  - widely used, especially in TDD and XP (eXtreme Programming).
- Alternative: TestNG [Tes14]

TDD with JUnit

- Tests are “living documentation” [Dic12]
  - One test – one feature.
  - Tests describe
    - requirements as expectations to the software,
    - functional behaviour of the application,
    - clearly interfaces.
  - Typical: 3:1 ratio of test code to production code.
  - Goal: Clean test code and clean designed production code
    → Refactor tests as well.
  - increases confidence and understanding of code.

TDD with JUnit

- Separate test code from production code to prevent getting test code into a production environment.
  - src directory contains production code,
  - test directory contains test code.



TDD with JUnit

- Testing with JUnit

```
import org.junit.Test;

public class SomethingToTest {

   @Test
   public void testMethod() {

      // MyClass is tested
      MyClass tester = new MyClass();

      // check if method(0,0) returns 0
      assertEquals("result is zero", 0, tester.method(0, 0));
   }
}
```

@Test is followed by test methods that get executed and are checked

must be public, and must be void

Use assertEquals to define the expectations of a test

TDD with JUnit

- Annotations (excerpt) [Vog13]

Annotation	Description
@Test public void method()	The @Test annotation identifies a method as a test method.
@Test(expected = Exception.class)	Fails if the method does not throw the named exception.
@Test(timeout=500)	Fails if the method takes longer than 500 milliseconds.
@Before public void method()	This method is executed before each test, e.g. to prepare the tests
@After public void method()	This method is executed after each test for clean-up purposes.
@Ignore	Ignores the test method, e.g. if test case has to be changed.

Optional additional message, if test fails

TDD with JUnit

- Assert statements (excerpt) [Vog13]

Statement	Description
assertTrue([message], boolean condition)	Checks that the boolean condition is true.
assertFalse([message], boolean condition)	Checks that the boolean condition is false.
assertEquals([message], expected, actual)	Tests that two values are the same. Note: For arrays the reference is checked not the content of the arrays.
assertNull([message], object)	Checks that the object is null.
assertNotNull([message], object)	Checks that the object is not null.
assertSame([message], expected, actual)	Checks that both variables refer to the same object.
assertNotSame([message], expected, actual)	Checks that both variables refer to different objects.

TDD with JUnit

- Proceeding [Dic12]
  - If your test fails first, you know the test is valid.
  - Write the least amount of code possible to get your tests to pass.
  - Use your tests to drive out the interfaces of your production code; results in clean API easy to use.
  - Try to call the outcome of your tests before they run.
  - Refactor your tests if applicable.
  - Make tests easy to read and easy to understand to improve maintainability.

TDD with JUnit

- Test exceptions [Gie09]
  e.g. to cover boundary conditions

Different illegal exceptions could be addressed – can test for only one exception

```
import org.junit.Test;

public class SomethingNegativeToTest {

    @Test(expected=RuntimeException.class)
    public void testNegativeWithdraw() throws Exception {
        account = new BankAccount();
        account.deposit(1000);
        account.withdraw(1001);
    }
}
```

TDD with JUnit

- Refactor your tests [Dic12]
  … to make them easy maintainable

Rerun tests after changes to catch errors early

```
import org.junit.Test;

public class SomethingRefactoredToTest {
   private BankAccount account;

   @Before
   public void init() {
      account = new BankAccount();
   }
   @Test
   public void testSomething() throws Exception { … }
   @After
   public void cleanup() { … }
}
```

To generate common set-up states for all tests

To clean up the test environment

Effective Use of Tools Potential Benefits and Risks

- Simply purchasing or leasing a tool does not guarantee success with that tool.
- “The goal of test automation should be to reduce the number of tests that need to be run manually, not to eliminate manual testing entirely” (Bret Pettichord) [Pet01].
- Each type of tool may require additional effort to achieve real and lasting benefits.
- Consider both with the use of tools in testing:
  - **Potential benefits** and opportunities,
  - **Risks**.

Effective Use of Tools Potential Benefits and Risks

Potential benefits:

- Finding defects in regression testing
  - because of side effects
  - because of wrong builds
- Reduced repetitive work concerning
  - run of regression tests,
  - re-entering same test data,
  - checking against coding standards.

Effective Use of Tools
Potential Benefits and Risks

Potential benefits:

- Greater consistency and repeatability
  - tests executed by a tool in the same order with the same frequency
  - tests derived from requirements

Effective Use of Tools
Potential Benefits and Risks

Potential benefits:

- Objective assessment like
  - static measures,
  - coverage results.
- Easy access to information about tests or testing, for example
  - statistics and graphs about test progress,
  - incident rates, and
  - performance

Effective Use of Tools Potential Benefits and Risks

Risks:

- Lack of clear goals.
- Unrealistic expectations for the tool including functionality and ease of use.
- Underestimating
  - time, cost and effort for the initial introduction of a tool including training and external expertise.
  - time and effort needed to achieve significant and continuing benefits from the tool including
    - need for changes in the testing process and
    - continuous improvement of the way the tool is used.

Effective Use of Tools
Potential Benefits and Risks

Risks:

- Underestimating the effort required to maintain the test assets generated by the tool.
- Overrating a tool and doing wrong decisions like
  - replacement for test design,
  - use of automated testing where manual testing would be better
    Consider: With test automation no new defects in new functionality could be detected.

Effective Use of Tools Potential Benefits and Risks

Risks:

- Neglecting
  - relationships and interoperability issues between critical tools like
    - requirements management tools,
    - version control tools,
    - incident management tools,
    - other tools and tools from multiple vendors.
  - version control of test assets within the tool.

Effective Use of Tools
Potential Benefits and Risks

Risks:

- Tool vendor could go out of business, retiring the tool, or selling the tool to a different vendor.
- Poor response from vendor for support, upgrades, and defect fixes.
- Suspension of open-source / free tool project.
- Inability to support a new platform.

Introducing a Tool into an Organization

- Tool needs to fit into organization / processes
  That's why: First assessment
  - What's about organizational maturity, strengths and weaknesses?
  - How test processes could be improved by tools?
- Definition of clear requirements and objective criteria to evaluate a tool.
- Estimation of a cost-benefit ratio based on a concrete business case.

Introducing a Tool into an Organization

Proposal: Proceeding

- Notice requirements.
- Collection of information, play around with tools.
- Vendors are normally interested in presenting their tools. Often it is possible to use a tool a specific time for free.
- Try a proof of concept.
- Intensive evaluation of tools in a realistic environment following defined requirements.
- Comparison of tools following requirements / recommendation.
- Decision with documentation of reasons.

Introducing a Tool into an Organization

Proposal: Proof-of-concept

- Does the tool perform effective with the software under test?
- Could current infrastructure be used or adaptation needed?
- Evaluation of
  - the vendor including training, support and commercial aspects (commercial tools).
  - service support suppliers (non-commercial tools).
- Training for tool and / or general test automation skills.

Introducing a Tool into an Organization

Proposal: Pilot project; aims:

- Learn more detail about the tool.
- How does the tool fit with existing processes? What has to be changed?
- How to use and maintain the tool and the test assets?
  Example: Folder structure, naming convention.
- Assess whether the benefits will be achieved at reasonable costs.

Introducing a Tool into an Organization

Success factors to establish a tool within an organization:

- Roll out incrementally.
- Adapt and improve processes to fit with the use of the tool.
- Provide training and support for users.
- Define usage guidelines.
- Monitor tool use and benefits.
- Gather lessons learned from all users.

Sources (1/7)

- [Ape14] Apache JMeter, 2014, http://jmeter.apache.org/
- [Bjo06] Goranka Bjedov: Using Open Source Tools for Performance Testing, at Google London Test Automation Conference (LTAC), Sept. 8th, 2006, http://www.youtube.com/watch?v=k9h51BM2h4w
- [Bor14] Borland: Silk Central™, 2014, http://www.borland.com/products/silkcentral/
- [Bor14a] Borland: Silk Test™, 2014, http://www.borland.com/products/silktest/
- [Bor14b] Borland: Silk Performer™, 2014, http://www.borland.com/products/silkperformer/
- [Can14] Canoo: Canoo Webtest, 2014, http://webtest.canoo.com/webtest/manual/WebTestHome.html

Sources (2/7)

- [Cha10] Graham Charlton: Ten free usability testing tools, 2010, http://econsultancy.com/us/blog/5932-ten-free-usability-testing-tools
- [Dic12] Jon Dickinson: Dive Into TDD, 2012, http://www.youtube.com/watch?v=PIWLC3dexSA
- [Dog12] Joe Dog Software: Siege, 2012, http://www.joedog.org/index/siege-home
- [Gie09] Adam Giemza: “SoftwareTests mit Junit”, Universität Duisburg-Essen, 2009, http://www.youtube.com/watch?v=8eeKvYurFU8
- [Gui07] Marc Guillemot: WebTest vs Selenium: WebTest wins 13 – 5, 2007, http://mguillem.wordpress.com/2007/10/29/webtest-vs-selenium-webtest-wins-13-5/
- [HP14] Hewlett Packard: HP Quality Center Enterprise, 2014, http://www8.hp.com/us/en/software-solutions/quality-center-quality-management/index.html

Sources (3/7)

- [HP14a] Hewlett Packard: HP Unified Functional Testing, 2014, http://www8.hp.com/us/en/software-solutions/unified-functional-testing-automated-testing/index.html
- [HP14b] Hewlett Packard: HP LoadRunner, 2014, http://www8.hp.com/th/en/software-solutions/loadrunner-load-testing/index.html
- [IBM14] IBM: Rational Quality Manager, 2014, http://www-03.ibm.com/software/products/en/ratiqualmana
- [IBM14a] IBM: Rational Test Workbench, 2014, http://www-03.ibm.com/software/products/en/rtw
- [IBM14b] IBM: Rational Performance Test Server, 2014, http://www-03.ibm.com/software/products/en/rpts

Sources (4/7)

- [ISTQB-CTFLS11] International Software Testing Qualifications Board: Certified Tester Foundation Level Syllabus, Released Version 2011, http://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html
- [ISTQB-GWP12] Glossary Working Party of International Software Testing Qualifications Board: Standard glossary of terms used in Software Testing, Version 2.2, 2012, http://www.istqb.org/downloads/glossary.html
- [Jen14] Jenkins: An extendable open source continuous integration server, 2014, http://jenkins-ci.org/
- [Jun14] JUnit.org – A programmer-oriented testing framework for Java. 2014, http://junit.org

Sources (5/7)

- [Mes11] Gerard Meszaros: Goals of Test Automation, 2011 http://xunitpatterns.com/Goals%20of%20Test%20Automation.html
- [MSA03] Gerard Meszaros, Shaun Smith, Jennitta Andrea: The Test Automation Manifesto, 2003 http://xunitpatterns.com/~gerard/xpau2003-test-automation-manifesto-paper.pdf
- [Ope14] open source software testing tools, news, and discussion, 2014, http://opensourcetesting.org
- [Pet01] Bret Pettichord: Success with Test Automation, Version of 28 June 2001.

Sources (6/7)

- [Sch11] Ina Schaefer, Software Systems Engineering, Technische Universität Braunschweig: Qualitätsmanagement, Software Engineering 1 WS 2011/2012, https://www.tu-braunschweig.de/Medien-DB/isf/sse/vl8_testen.pdf
- [Sel14] SeleniumHQ: Web application testing system, 2014, http://seleniumhq.org/
- [SG08] Carey Schwaber, Mike Gualtieri with Mike Gilpin, David D'Silva: The Forrester Wave™: Functional Testing Solutions, Q3 2008, http://www.forrester.com/
- [Tes14] TestNG, 2014, http://testng.org/
- [Tl14] TestLink, 2014, http://testlink.org/
- [Tom09] W. Craig Tomlin: 24 Usability Testing Tools, 2009, http://www.usefulusability.com/24-usability-testing-tools

Sources (7/7)

- [Vis14] Visual Studio, 2014, http://www.visualstudio.com/en-us/products/visual-studio-with-msdn-overview-vs
- [Vog13] Lars Vogel: Unit Testing with JUnit – Tutorial, Version 2.4, 2013, http://www.vogella.com/tutorials/JUnit/article.html
- [XQ14] XQual: XStudio, 2014, http://www.xqual.com/
- [Wat14] Watir.com: Web Application Testing in Ruby, 2014, http://watir.com/
- [Wik14] wikipedia.org: Comparison of open-source configuration management software, 2014, http://en.wikipedia.org/wiki/Comparison_of_open_source_configuration_management_software
- [Wik14a] wikipedia.org: Test-driven development, 2014, http://en.wikipedia.org/wiki/Test-driven_development