Software-Qualitätssicherung (QS-VU)
LV 188.911

An Introduction to Automated Quality Assurance

Univ. Lektor DI Mag. Roland Breiteneder

Vienna University of Technology
Research Group for Industrial Software (INSO)

roland.breiteneder@inso.tuwien.ac.at

Agenda

- Introduction
- Tool support for testing levels
- Automation with Continuous Integration
- Issue Tracking
- Refactorings

Motivation

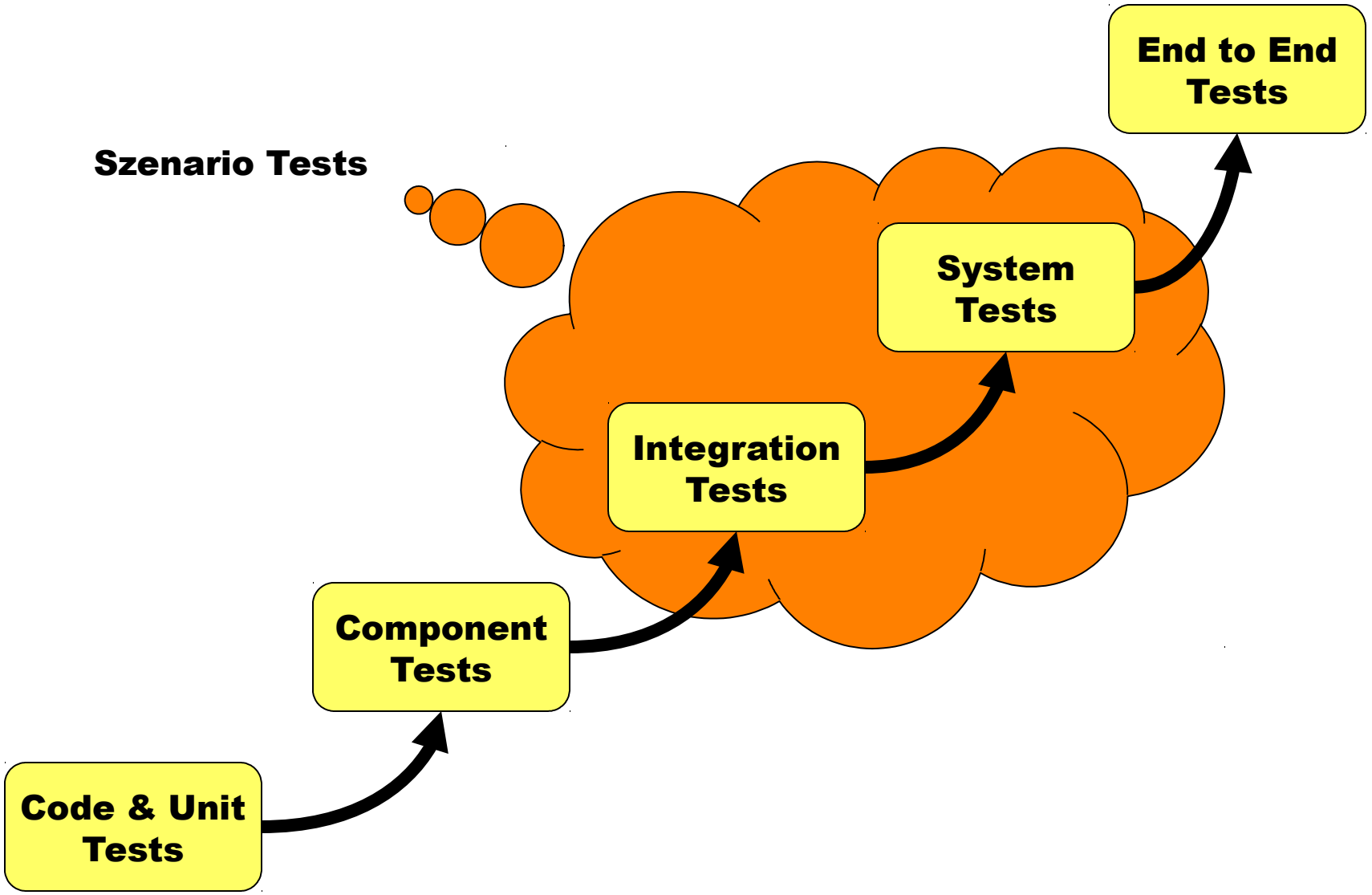
- A main goal in Software Engineering (SE) is the construction of high-quality and testable software products.
- Important key approaches:
  - Software processes (e.g., Life-Cycle Model, V-Modell XT, Scrum) define the sequence of steps, *when* to build *which* products.
  - Constructive methods, e.g., Model-Driven Development, Test-Driven Development, and pair programming support creating new (pieces of) software products (*how to build*).
  - Analytical methods, e.g., inspection and testing help assessing product and process quality (*how to evaluate, verify, validate, etc*).
- Quality Assurance (QA) supports engineers in evaluating products in every stage of software development and must be an integrated part of the development process.
- Manual QA activities vs. automated QA activities?!
- How is it possible to automate QA activities within the SE development cycle?

Quality Assurance „Pattern“

- Manual QA activities (e.g. reviews) includes a high effort (e.g., regression testing after every change)

- Automation and other selective QA approaches help reducing effort and increasing product quality:
  - Reviews of mission-critical test cases:
    Test case generation during review to focus on key requirements and to identify potential problems early (reuse of test cases)
  - Test-Driven Development: Test case generation in prior or parallel to software construction.
  - Pair Development and systematic QA.
  - Requirements Tracking
  - Automated Code Quality Checks (e.g., comments, code complexity)
  - Test Case Coverage
  - Etc.

Categorisation by Granularity



Examples of Tests of different “Granularity”

- Unit Tests with JUnit (see SE)
- Database-Support with DBUnit
- Mocking with JMock
- Load & Functional Tests with
  - JMeter
  - HTML/HTTP-Unit

Categorisation by Focus of Test

- Functional Requirements (e.g. backend)
- Integration of components into a specific environment
- User interface test
- Non-Functional requirements like performance, failover, time-to-restart, scalability
- User acceptance (i.e., UI design, UI testing)
- Others...?

- *... with increasing complexity to automate (functional requirements, e.g. classes can be tested with unit-test frameworks. User acceptance on the other hand can hardly by tested automatically)*

JUnit and “automated Database Configuration”

- Unit-Testing applications that use a database is often tricky
  - One problem is, to get the database into a clear initial state
  - Several database instances might exist (from DBUnit documentation):
    - Production database (you would not want to test against this one!)
    - Local development database
    - Populated development database (with “realistic” data for developers)
    - Test database
- DBUnit is a “helper” project, that allows to set the database to a defined state before unit-tests are executed
- Plugins for e.g. Maven exist
- Verification of table content is as expected

“Test-Helper” Objects

- Stub
  - “Replay” predefined values, might also record and replay; no “real” functionality
- Mock
  - Also no “real” functionality, but are programmed to react according to expectations

Mock Objects

- Object that “mimics” the behaviour of a “real” object that is
  - Not available at the time of testing (e.g., some complex legacy application)
  - Should not be invoked (operational system)
  - Is expensive or difficult to create
  - Is difficult to bring into a specific status

- Several Frameworks support developers in creating Mock objects, e.g. Mockito, JMock

HttpUnit / HtmlUnit

- Works with JUnit
- httpunit models http protocol
- htmlunit models html documents returned from a web-application
- Emulates relevant browser behaviour including Javascript, authentication, cookies, automatic page redirection...
- Check if response from server is as expected

HtmlUnit Example

```
public void testHomePage() throws Exception {
    final WebClient webClient = new WebClient();
    final URL url = new URL("http://htmlunit.sourceforge.net");
    final HtmlPage page = (HtmlPage)webClient.getPage(url);
    assertEquals( "htmlunit - Welcome to HtmlUnit", page.getTitleText() );
}
```

- These two examples from the htmlunit website show two simple applications for htmlunit.
- On this slide: read a webpage and check, if the page title is as expected.
- On the next slice: read a webpage and fill out a form and send the result back to the server.

Htmlunit Example: Form Handling

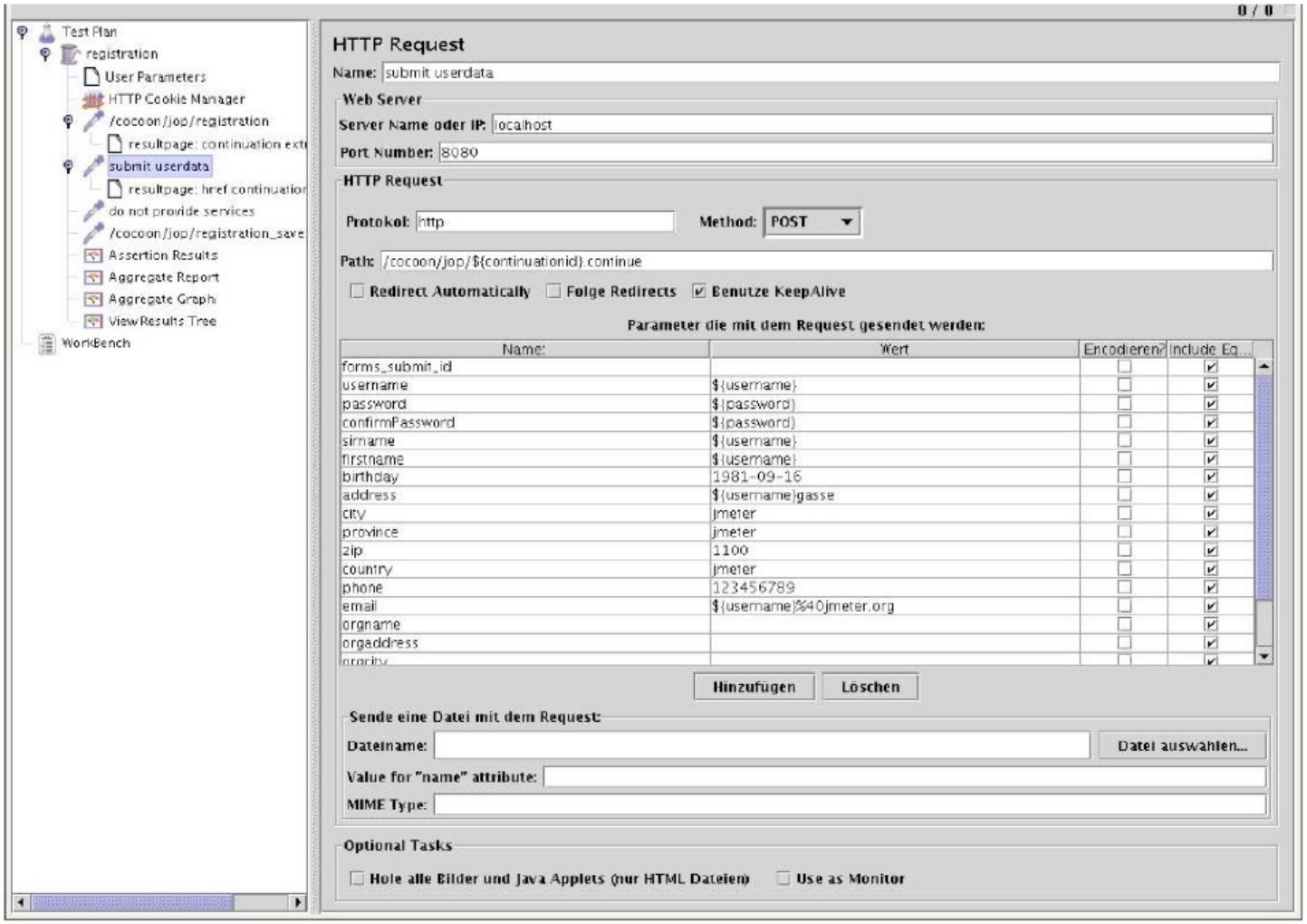
```
public void testHomePage() throws Exception {
    final WebClient webClient = new WebClient();
    final URL url = new URL("http://htmlunit.sourceforge.net");

    // Get the first page
    final HtmlPage page1 = (HtmlPage)webClient.getPage(url);

    // Get the form that we are dealing with and within that form,
    // find the submit button and the field that we want to change.
    final HtmlForm form = page1.getFormByName("myform");
    final HtmlSubmitInput button
        = (HtmlSubmitInput)form.getInputByName("submitbutton");
    final HtmlTextInput textField
        = (HtmlTextInput)form.getInputByName("userid");

    // Change the value of the text field
    textField.setValueAttribute("root");

    // Now submit the form by clicking the button and get back the
    // second page.
    final HtmlPage page2 = (HtmlPage)button.click();
}
```

Load and Functional Testing of (Web/Network)-Applications: e.g. JMeter

- Load and Functional testing of web-applications (generally UIs) is not a trivial tasks

- Tools like Apache JMeter allow to
  - Make load and performance tests on servers
    - http
    - JDBC
    - ftp
  - Make functional tests according to defined scripts
  - Test Web-UI (Frontend Tests)
  - Record/Replay (working as proxy in record mode)

JMeter Test Plan



Code Quality Checks

- Some types of code-quality checks can be performed automatically and can give hints about potential problems in the code
- Specialised tools can check unit test code coverage (e.g. Clover, JCover)
- Several tools exist to support such checks
  - IDEs
  - Specialised tools like Checkstyle
- Tools can be embedded in Build automation and can provide integrated reporting

Example: Checkstyle

- Config file specifies the coding standard (i.e. Sun Code Conventions)
- Standard Checks include:
  - Javadoc Comments
  - Naming Conventions
  - Headers
  - Imports (*)
  - Size Violations
  - Parenthesis
  - Modifier Order
  - Coding (30 Options)
    - Modified control variable (for...)
    - Simplify boolean expressions
  - **Duplicate Code**
  - Cyclomatic Complexity (minimal path through codepart)
  - NPath Complexity (number of possible execution paths through a function)
  - Search for TODO: Comments

Example: FindBugs

- Based on the concept of bug patterns
- Plugin architecture for adding new bug patterns

- Found bugs include:
  - *Bad Practice:* Method may fail to close stream
  - *Bad Practice:* Class defines equals() but not hashCode()
  - *Correctness:* Useless assignment in return statement
  - *Correctness:* hasNext method invokes next
  - *Malicious code vulnerability:* Public static method may expose internal representation by returning array
  - *Multithreaded correctness:* Constructor invokes Thread.start()
  - *Performance:* Method invokes toString() method on a String
  - *Performance:* Method calls static Math class method on a constant value
  - *Security:* Empty database password
  - *Security:* HTTP cookie formed from untrusted input

Automation

- Try to embed testing into a automated build-environment,
  e.g. Ant, nAnt, Maven
- Execute tests as often as possible (after code changes)
  see *continuous integration*
- Create automated reports and notifications for development team
- Integrate development tools as good as possible (integration in the meaning of “tool-integration” but also “developer workflow integration” i.e.: educate the team and the customers) e.g.:
  - Build automation (Maven...)
  - Reporting
  - Code-Management (SVN)
  - **Bug Tracking**
- More complex tests like integration or end-to-end tests might require a more complex setup of the environment including mocking of legacy systems

Automation of Integration and Testing?

"The most fundamental part of the daily build is the 'daily' part. . . . Treat the daily build as the heartbeat of the project. If there's no heartbeat, the project is dead."

James McCarthy

- Modern software projects often use a set of complex frameworks and configurations and are embedded in a variety of platforms.
  "It works on my machine" is not sufficient!
- Development Teams develop distributed (on various modules), but sources have to be kept on a central sourcecode repository:
  "What cannot be found in the SVN is not existing in the project"
- *"It compiles, let's ship it"* is not an appropriate strategy: Various modules are interacting: do daily builds and test-cycles to reduce risk of failure, or to detect problems early

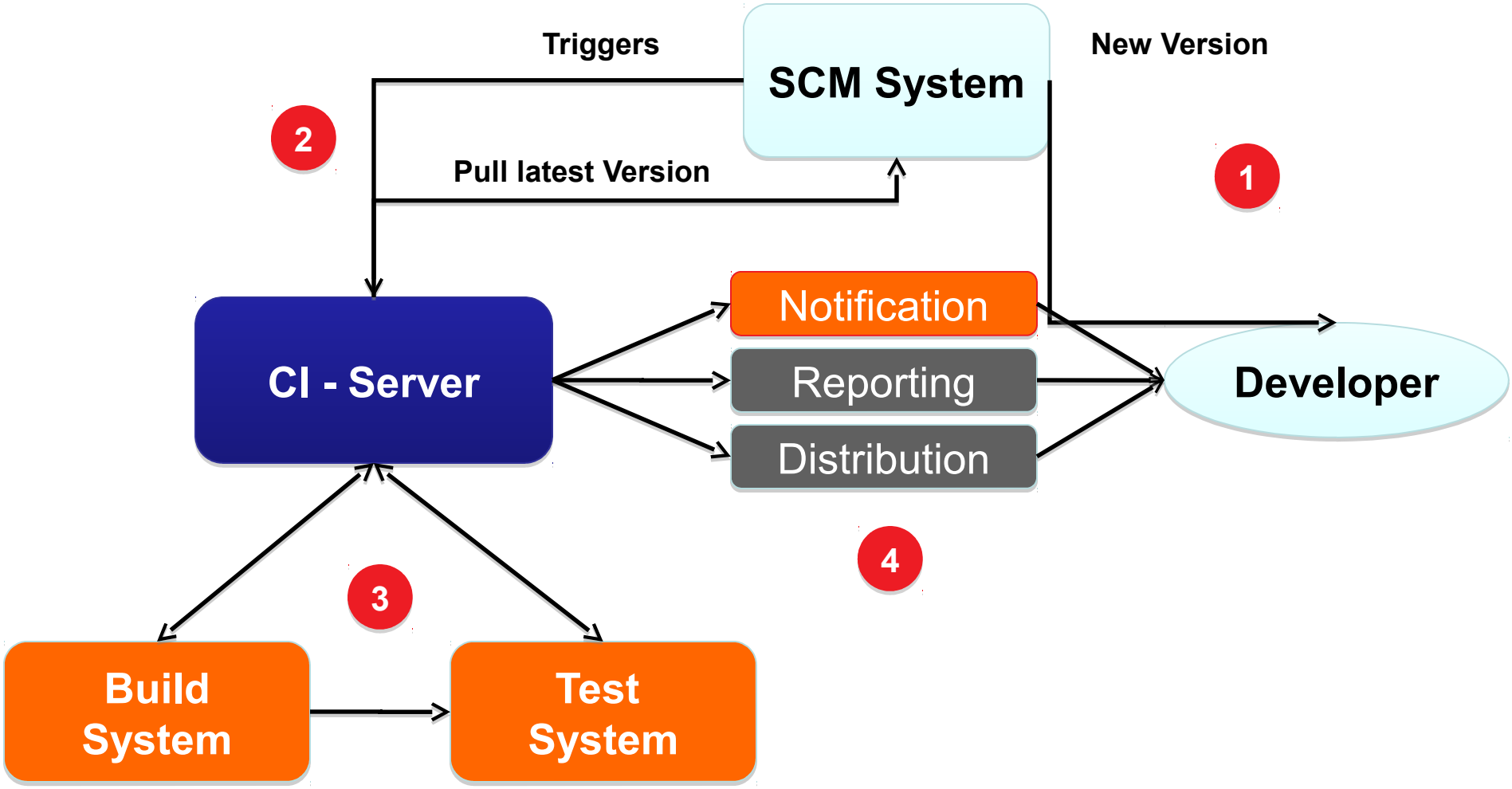
Continuous Integration

- Martin Fowler indicates the following best-practices:
  - Single Sourcecode Repository
  - Build Automation
  - Make build "Self-testing"
  - Daily (or more regular) commits by each developer!
  - Keep the build fast
  - *Test in a clone of the production environment*
  - **Transparency**
    - Easy access to latest binary (see e.g. Apache Daily builds)
    - Everyone should see the status (web-reports, notifications)
  - Automate Deployment

Continuous Integration Tool Support

- Build automation tools like Maven are great, but are still “developer-centric”
- Continuous Integration tools like Apache Continuum or Hudson allow server based automated integration
  and execution of tests
- Automation includes
  - Invocation based on events or scheduler (e.g. daily)
  - Build using build system
  - Execution of Tests
  - Creation of Reports
  - Sending of Notifications

Continuous Integration Workflow



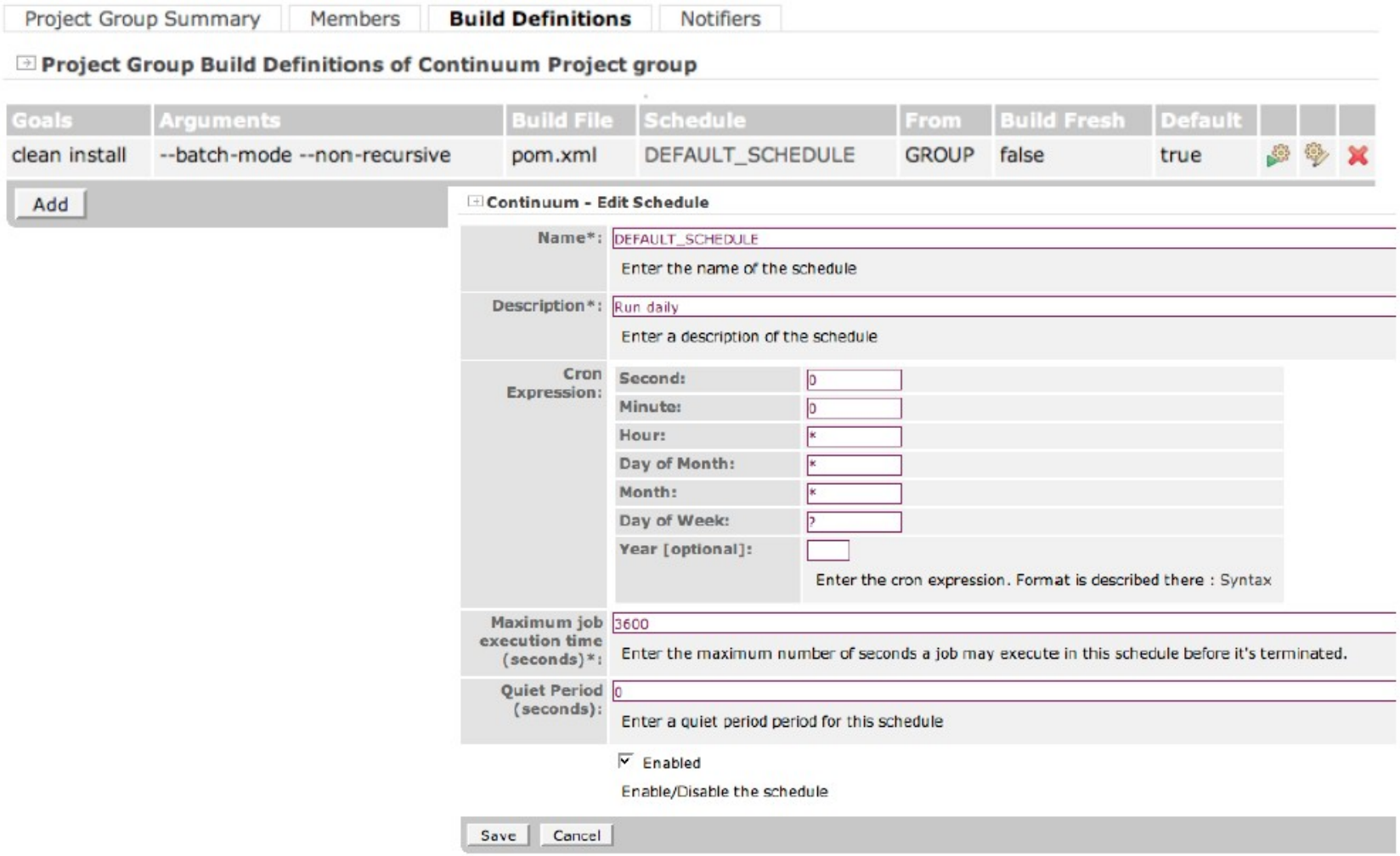
Apache Continuum

- Maven Subproject
- Installation on Server
- Definition of series of projects
  - Maven 1
  - Maven 2
  - Ant
  - Shell
- Integration with several SCM (source code management) systems like SVN, CVS
- Web-Frontend for working with projects & reporting
- Schedule Definition for Build automation
- Reporting
- Notification
- *See Example Screeshots on next slides*

Result List

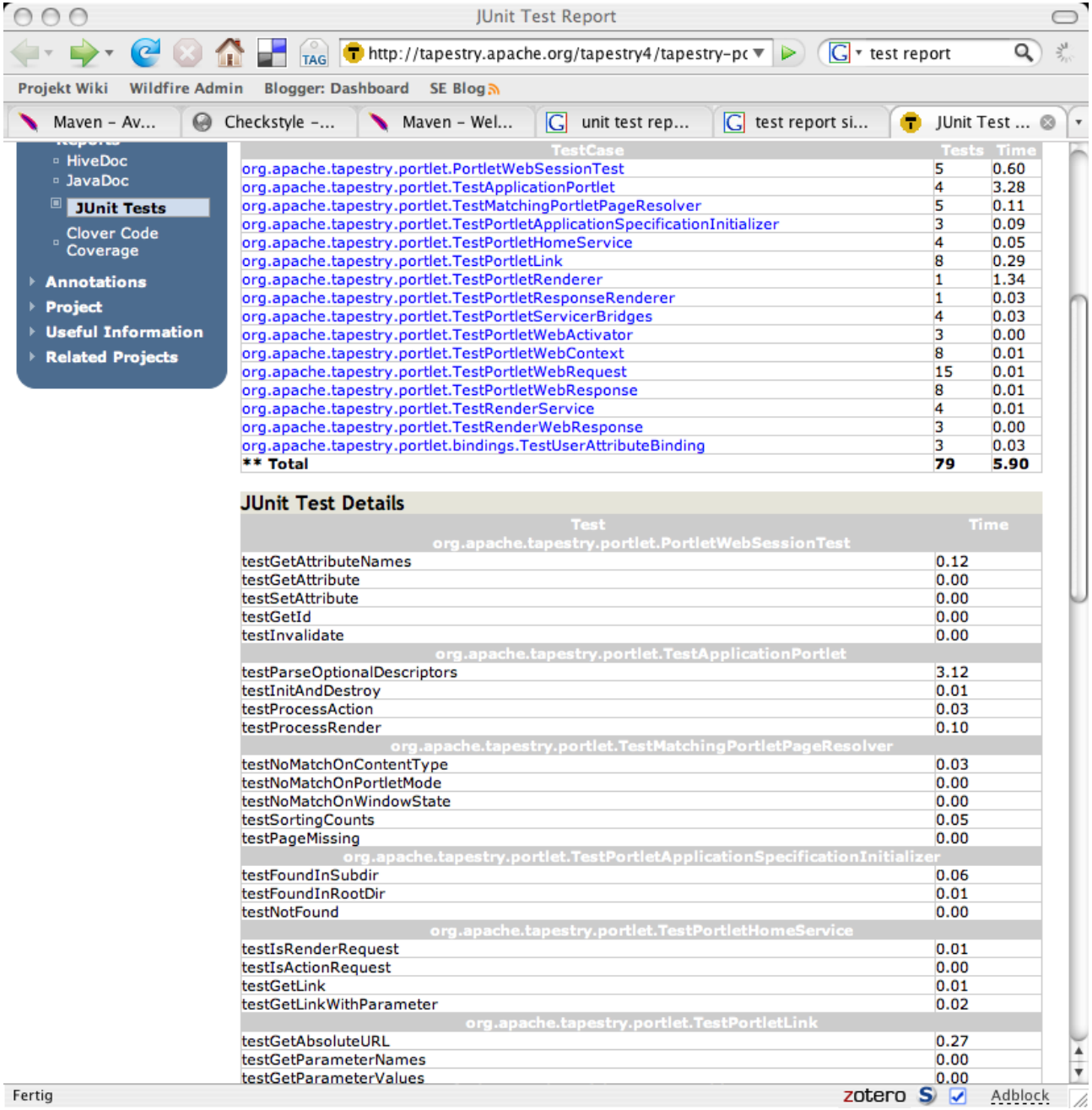
	Project Name	Version	Build	Group	
	Continuum Project	1.1-SNAPSHOT	1	Continuum Project	
	Continuum API	1.1-SNAPSHOT		Continuum Project	
	Continuum Security	1.1-SNAPSHOT	1	Continuum Project	
	Continuum Core	1.1-SNAPSHOT		Continuum Project	
	Continuum Model	1.1-SNAPSHOT		Continuum Project	
	Continuum Notifiers	1.1-SNAPSHOT	1	Continuum Project	
	Continuum Notifier API	1.1-SNAPSHOT		Continuum Project	
	Continuum IRC Notifier	1.1-SNAPSHOT		Continuum Project	
	Continuum Jabber Notifier	1.1-SNAPSHOT		Continuum Project	
	Continuum MSN Notifier	1.1-SNAPSHOT		Continuum Project	

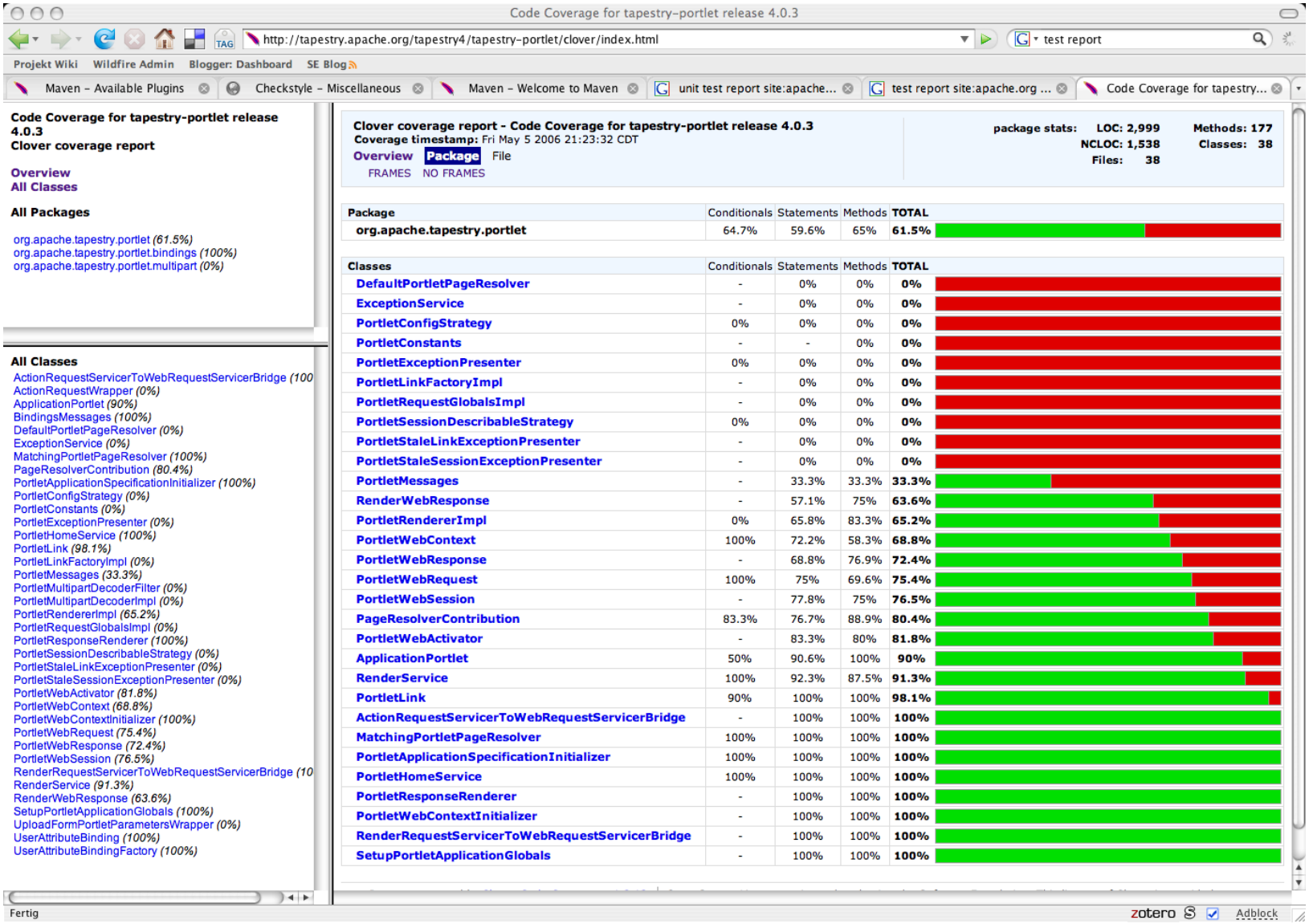
Definition of Schedulers for Build Automation



Automation, Reporting and Documentation

- Documentation and Reporting are integral aspects of automated testing
- Most tools produce test reports in generic formats, that can be reused
  - Text
  - XML
  - HTML
- Integration into Documentation Systems e.g., with Maven is recommended!

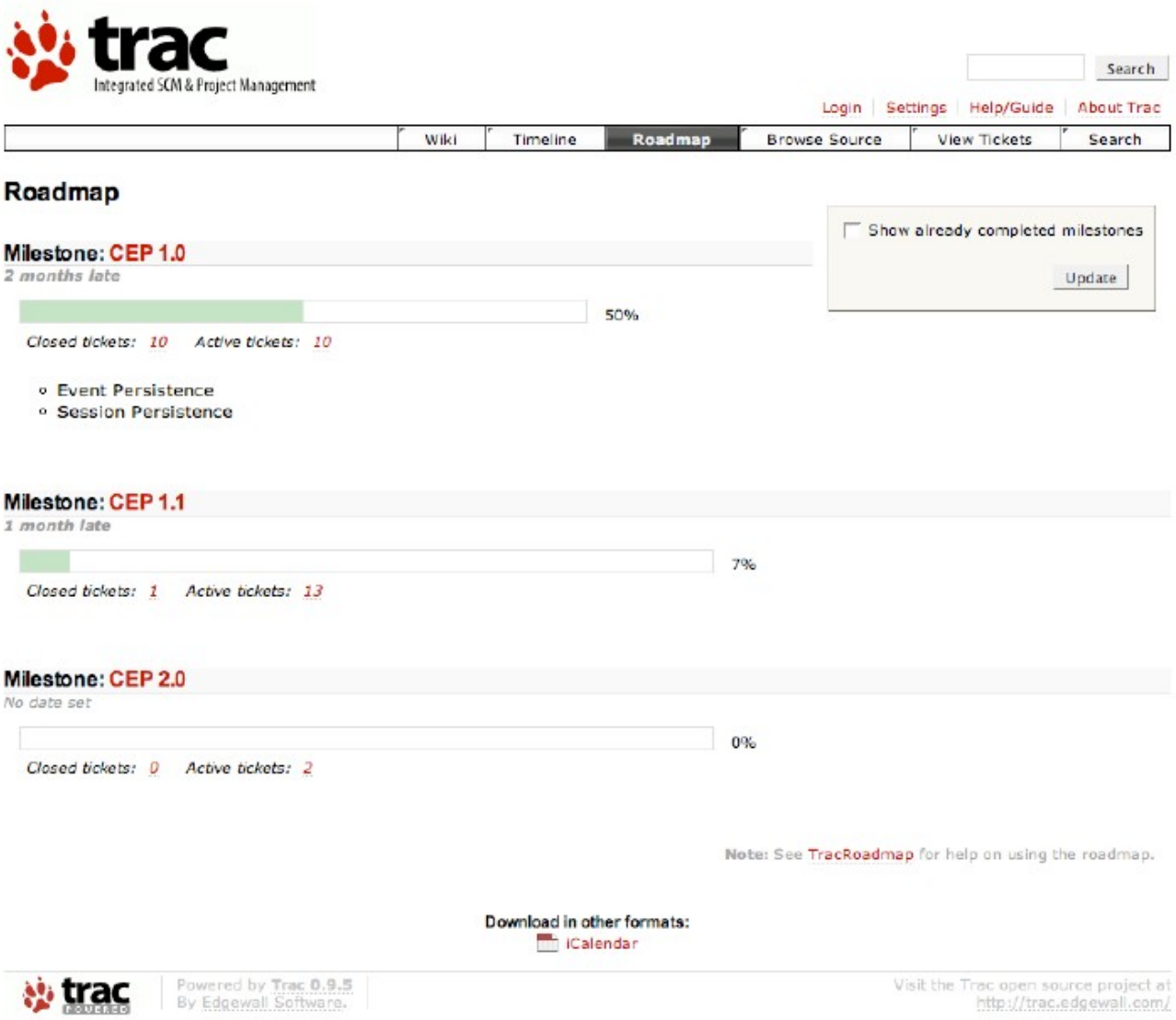

JUnit Test Report
http://tapestry.apache.org/tapestry4/tapestry-pc
test report
Projekt Wiki
Wildfire Admin
Blogger: Dashboard
SE Blog
Maven – Av...
Checkstyle –...
Maven – Wel...
unit test rep...
test report si...
JUnit Test ...
HiveDoc
JavaDoc
JUnit Tests
Clover Code Coverage
Annotations
Project
Useful Information
Related Projects
TestCase Tests Time
org.apache.tapestry.portlet.PortletWebSessionTest 5 0.60
org.apache.tapestry.portlet.TestApplicationPortlet 4 3.28
org.apache.tapestry.portlet.TestMatchingPortletPageResolver 5 0.11
org.apache.tapestry.portlet.TestPortletApplicationSpecificationInitializer 3 0.09
org.apache.tapestry.portlet.TestPortletHomeService 4 0.05
org.apache.tapestry.portlet.TestPortletLink 8 0.29
org.apache.tapestry.portlet.TestPortletRenderer 1 1.34
org.apache.tapestry.portlet.TestPortletResponseRenderer 1 0.03
org.apache.tapestry.portlet.TestPortletServicerBridges 4 0.03
org.apache.tapestry.portlet.TestPortletWebActivator 3 0.00
org.apache.tapestry.portlet.TestPortletWebContext 8 0.01
org.apache.tapestry.portlet.TestPortletWebRequest 15 0.01
org.apache.tapestry.portlet.TestPortletWebResponse 8 0.01
org.apache.tapestry.portlet.TestRenderService 4 0.01
org.apache.tapestry.portlet.TestRenderWebResponse 3 0.00
org.apache.tapestry.portlet.bindings.TestUserAttributeBinding 3 0.03
** Total 79 5.90
JUnit Test Details
Test Time
org.apache.tapestry.portlet.PortletWebSessionTest
testGetAttributeNames 0.12
testGetAttribute 0.00
testSetAttribute 0.00
testGetId 0.00
testInvalidate 0.00
org.apache.tapestry.portlet.TestApplicationPortlet
testParseOptionalDescriptors 3.12
testInitAndDestroy 0.01
testProcessAction 0.03
testProcessRender 0.10
org.apache.tapestry.portlet.TestMatchingPortletPageResolver
testNoMatchOnContentType 0.03
testNoMatchOnPortletMode 0.00
testNoMatchOnWindowState 0.00
testSortingCounts 0.05
testPageMissing 0.00
org.apache.tapestry.portlet.TestPortletApplicationSpecificationInitializer
testFoundInSubdir 0.06
testFoundInRootDir 0.01
testNotFound 0.00
org.apache.tapestry.portlet.TestPortletHomeService
testIsRenderRequest 0.01
testIsActionRequest 0.00
testGetLink 0.01
testGetLinkWithParameter 0.02
org.apache.tapestry.portlet.TestPortletLink
testGetAbsoluteURL 0.27
testGetParameterNames 0.00
testGetParameterValues 0.00
Fertig
zotero
Adblock


Code Coverage for tapestry-portlet release 4.0.3
http://tapestry.apache.org/tapestry4/tapestry-portlet/clover/index.html
test report
Projekt Wiki
Wildfire Admin
Blogger: Dashboard
SE Blog
Maven – Available Plugins
Checkstyle – Miscellaneous
Maven – Welcome to Maven
unit test report site:apache...
test report site:apache.org ...
Code Coverage for tapestry...
Code Coverage for tapestry-portlet release 4.0.3
Clover coverage report
Overview
All Classes
All Packages
org.apache.tapestry.portlet (61.5%)
org.apache.tapestry.portlet.bindings (100%)
org.apache.tapestry.portlet.multipart (0%)
All Classes
ActionRequestServicerToWebRequestServicerBridge (100
ActionRequestWrapper (0%)
ApplicationPortlet (90%)
BindingsMessages (100%)
DefaultPortletPageResolver (0%)
ExceptionService (0%)
MatchingPortletPageResolver (100%)
PageResolverContribution (80.4%)
PortletApplicationSpecificationInitializer (100%)
PortletConfigStrategy (0%)
PortletConstants (0%)
PortletExceptionPresenter (0%)
PortletHomeService (100%)
PortletLink (98.1%)
PortletLinkFactoryImpl (0%)
PortletMessages (33.3%)
PortletMultipartDecoderFilter (0%)
PortletMultipartDecoderImpl (0%)
PortletRendererImpl (65.2%)
PortletRequestGlobalsImpl (0%)
PortletResponseRenderer (100%)
PortletSessionDescribableStrategy (0%)
PortletStaleLinkExceptionPresenter (0%)
PortletStaleSessionExceptionPresenter (0%)
PortletWebActivator (81.8%)
PortletWebContext (68.8%)
PortletWebContextInitializer (100%)
PortletWebRequest (75.4%)
PortletWebResponse (72.4%)
PortletWebSession (76.5%)
RenderRequestServicerToWebRequestServicerBridge (10
RenderService (91.3%)
RenderWebResponse (63.6%)
SetupPortletApplicationGlobals (100%)
UploadFormPortletParametersWrapper (0%)
UserAttributeBinding (100%)
UserAttributeBindingFactory (100%)
Clover coverage report - Code Coverage for tapestry-portlet release 4.0.3
Coverage timestamp: Fri May 5 2006 21:23:32 CDT
Overview Package File
FRAMES NO FRAMES
package stats: LOC: 2,999 NCLOC: 1,538 Files: 38
Methods: 177 Classes: 38
Package Conditionals Statements Methods TOTAL
org.apache.tapestry.portlet 64.7% 59.6% 65% 61.5%
Classes Conditionals Statements Methods TOTAL
DefaultPortletPageResolver - 0% 0% 0%
ExceptionService - 0% 0% 0%
PortletConfigStrategy 0% 0% 0% 0%
PortletConstants - - 0% 0%
PortletExceptionPresenter 0% 0% 0% 0%
PortletLinkFactoryImpl - 0% 0% 0%
PortletRequestGlobalsImpl - 0% 0% 0%
PortletSessionDescribableStrategy 0% 0% 0% 0%
PortletStaleLinkExceptionPresenter - 0% 0% 0%
PortletStaleSessionExceptionPresenter - 0% 0% 0%
PortletMessages - 33.3% 33.3% 33.3%
RenderWebResponse - 57.1% 75% 63.6%
PortletRendererImpl 0% 65.8% 83.3% 65.2%
PortletWebContext 100% 72.2% 58.3% 68.8%
PortletWebResponse - 68.8% 76.9% 72.4%
PortletWebRequest 100% 75% 69.6% 75.4%
PortletWebSession - 77.8% 75% 76.5%
PageResolverContribution 83.3% 76.7% 88.9% 80.4%
PortletWebActivator - 83.3% 80% 81.8%
ApplicationPortlet 50% 90.6% 100% 90%
RenderService 100% 92.3% 87.5% 91.3%
PortletLink 90% 100% 100% 98.1%
ActionRequestServicerToWebRequestServicerBridge - 100% 100% 100%
MatchingPortletPageResolver 100% 100% 100% 100%
PortletApplicationSpecificationInitializer 100% 100% 100% 100%
PortletHomeService 100% 100% 100% 100%
PortletResponseRenderer - 100% 100% 100%
PortletWebContextInitializer - 100% 100% 100%
RenderRequestServicerToWebRequestServicerBridge - 100% 100% 100%
SetupPortletApplicationGlobals - 100% 100% 100%
Fertig
zotero
Adblock

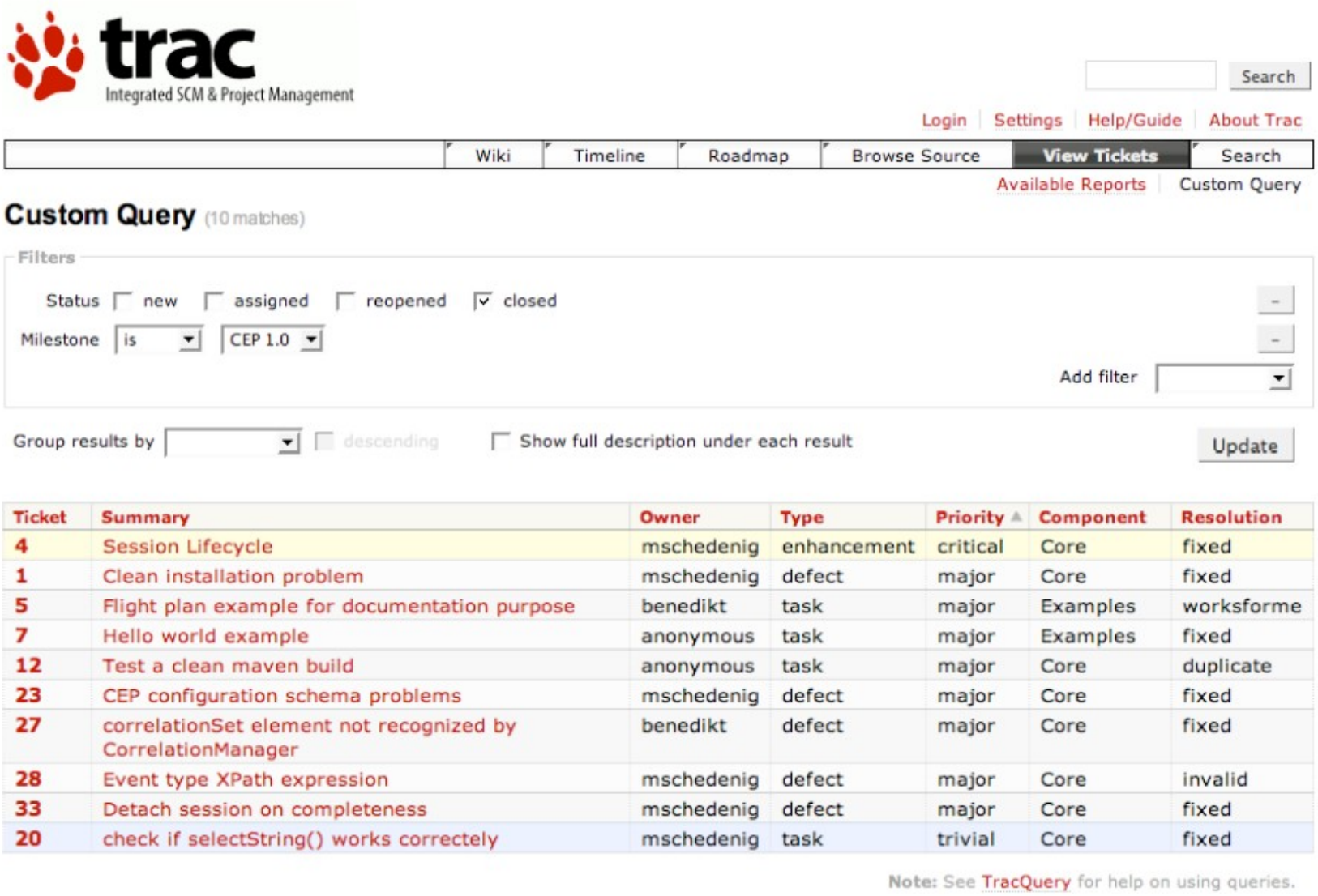
Issue Tracking & Tool Integration

- Automatic Testing is important, but many problems and issues are found by humans, i.e.
  - Developers themselves
  - Customers
  - Users
- A proper Bug/Issuetracking tool is imperative for a software project
- All stakeholders should have access to the tool
- Users should be able to add issues
- Select/configre tool so that also “end-users” are able to add issues!

Trac Issuetracking



Trac Tickets to Roadmap



Ticket	Summary	Owner	Type	Priority	Component	Resolution
4	Session Lifecycle	mschedenig	enhancement	critical	Core	fixed
1	Clean installation problem	mschedenig	defect	major	Core	fixed
5	Flight plan example for documentation purpose	benedikt	task	major	Examples	worksforme
7	Hello world example	anonymous	task	major	Examples	fixed
12	Test a clean maven build	anonymous	task	major	Core	duplicate
23	CEP configuration schema problems	mschedenig	defect	major	Core	fixed
27	correlationSet element not recognized by CorrelationManager	benedikt	defect	major	Core	fixed
28	Event type XPath expression	mschedenig	defect	major	Core	invalid
33	Detach session on completeness	mschedenig	defect	major	Core	fixed
20	check if selectString() works correctely	mschedenig	task	trivial	Core	fixed

Note: See TracQuery for help on using queries.

Trac Reports

Trac Ticket Details

Ticket #4 (enhancement)

Session Lifecycle

Opened 4 months ago
Last modified 3 months ago

Status: closed (fixed)

Reported by:	anonymous	Assigned to:	mschedenig
Priority:	critical	Milestone:	CEP 1.0
Component:	Core	Keywords:	
Cc:			

Implement session lifecycle handling, i.e. timeouts/TTL counters and automatic purging of outdated sessions.

Change History

02/07/07 18:46:13: Modified by anonymous

- **owner** set to *anonymous*.
- **component** set to *Core*.

02/18/07 20:08:20: Modified by benedikt

- **owner** changed from *anonymous* to *marian*.

02/18/07 21:59:55: Modified by mschedenig

- **owner** changed from *marian* to *mschedenig*.
- **status** changed from *new* to *assigned*.
- **description** changed.

02/21/07 18:28:23: Modified by anonymous

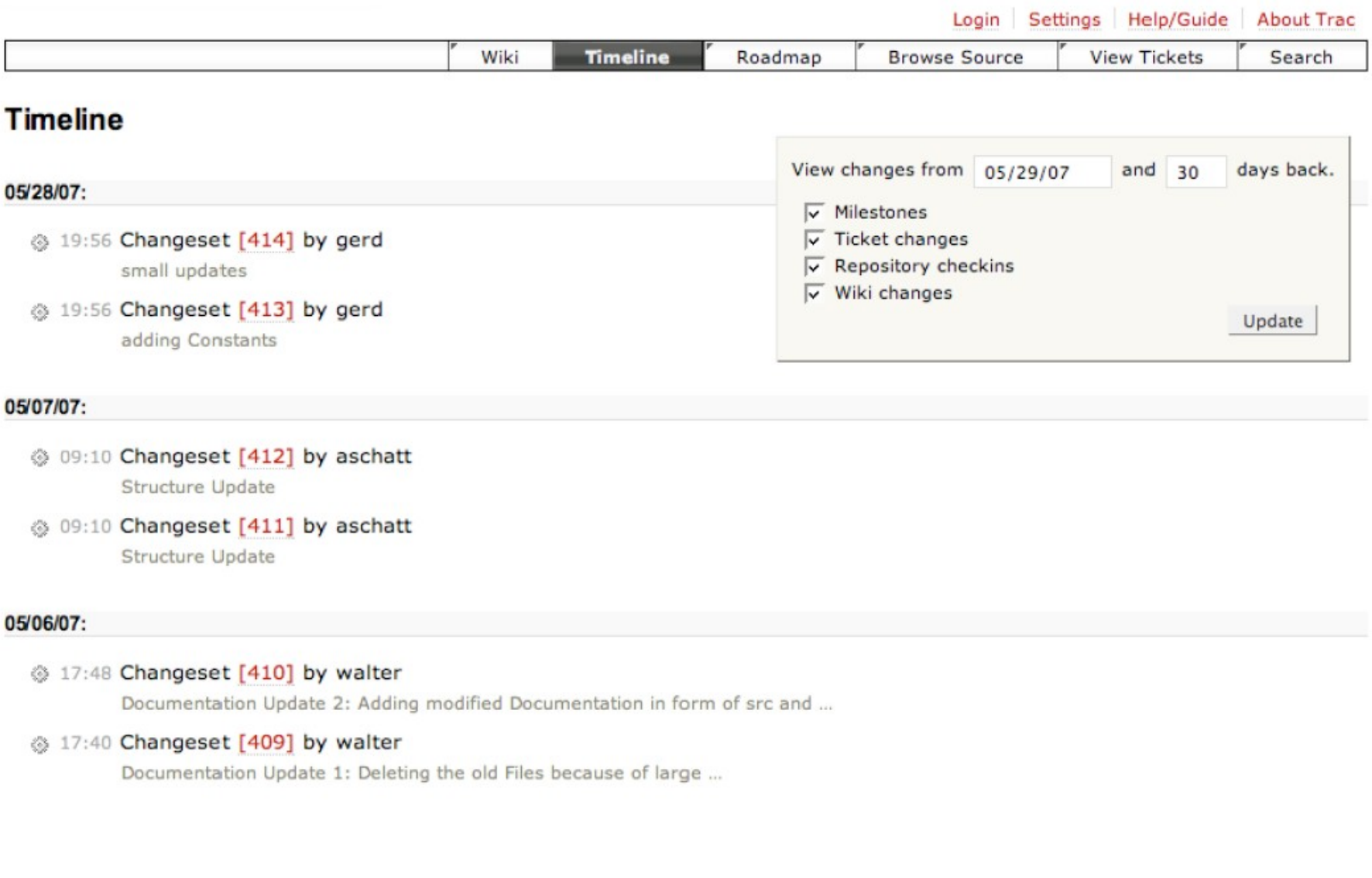
- **priority** changed from *major* to *critical*.

02/25/07 18:59:08: Modified by mschedenig

- **status** changed from *assigned* to *closed*.
- **resolution** set to *fixed*.

Sessions can now have a timeout value which can be set directly either to a specific time or by specifying the number of milliseconds until the session is to time out. Also, in the correlation definition, the default timeout duration for a session can be set (in seconds).

Trac Timeline



Tool Integration

- Ideally build-tools are highly integrated
- E.g.: Trac allows to connect to SVN using “post-commit” hooks, this means
  - When developers commit changes and add the Ticket ID to the commit info Trac gets informed about the commit
  - Then Trac shows the SVN info / sources with the ticket
- Continuous Build Tool
  - React on SVN commits
  - Notify developers (mail, IM)
- Requirements-Tracking
  - Attach Requirements to SVN, Issues, ...

Refactoring

- **Changes in the inner structure** of a System, that keep “external” behaviour the same
- **“Cleaning up”** code, but no “bug-fixing” or “improvements” in functionality, but improvement of
  - Understandability
  - Design/Architecture
  - Reusability
  - Helping to find bugs

Photo from flickr (doeth)

- Often weaknesses of the code—**“bad smells”**—show up during test-phase or even in production phase, e.g. when trying to add new functionality
- Refactoring is a **systematic approach** to enhance code quality

“Bad Smells”

- Martin Fowler describes a list of “bad smell”, i.e. indicators that there might be problems or design flaws in the application like:
  - Duplicated Code
  - Long Methods, large classes
  - Long Parameter lists
  - “Shotgun Surgery”
  - Feature Envy
  - Lazy Class
  - Speculative Generality
  - Inappropriate Intimacy
  - “Comments used as deodorant”

Ref. Example: “Encapsulate Field”

- Public field to private field plus accessors (overriding in subclasses, lazy loading, validation...)

```
class X {
  public String name;
}
```

```
class X {
  private String name;
  public void setName(String name) {
    this.name = name;
  }
  public String getName() {
    return this.name;
}
```

Ref. Example: “Extract Method”

- Long methods
- Complex Logic within one method
- Reuse of Code is not possible

```
class X {
  public void sortData
 (String infile, outfile) {

    File fin = ...(infile);
    // read data into array
    List data = ...
    ...
    // implement quicksort
    ...
    // write sorted data
    File fout = ... (outfile);
    ...
}
```

```
class X {
  public void sortData
  (String infile, outfile) {
    List data = readFile (infile);
    List sdata = quicksort(data);
    writeFile (sdata);
  private List readFile (String f) {
    ...
  }
  private List quicksort (List d) {
    ...
  }
  private void writeFile (List d) {
    ...
  }
}
```

Ref. Example: “Move Method”

- A method is using more features/functionality of another class than of the class it is implemented in
- E.g.: Customer : Order : DVDs (1:n, n:1)
  - calculateOrderPrice (...) in Customer
  - but only depending on one Order
  - hence, move method from Customer to Order

Ref. Example: “Replace Temp with Query”

- Data that is calculated or prepared for further use in a temporary variable can be replaced by a method call

```
class Bill {
  public void printBill(){
    // print header
    ...
    // print item list
    ...
    // print sum without taxes
    ...
    double tax = sum * 0.20;
    double sum_tax = sum + tax;
    // print sum with taxes
    ...
    // print footer
    ...
  }
}
```



```
class Bill {
  public static final double
  TAX = 0.20;
  public void printBill(){
    // print header
    ...
    // print item list
    ...
    // print sum without taxes
    ...
    double sum_tax =
    calcSumTax(sum);
    ... }
  double calcSumTax(double s) {
    return s*Bill.TAX;
  }
}
```

Summary

- Modern quality assurance is tool driven
- Automation with continuous integration is crucial for software construction and quality assurance
- Issue Tracking ensures collaboration between development and testing and allows to track the progress
- Refactorings increase the quality of the software without changing its functionality

Literaturreferenzen

- Biffl Stefan, Winkler Dietmar, Frast Denis: „Qualitätssicherung, Qualitätsmanagement und Testen in der Softwareentwicklung“, Skriptum zur Lehrveranstaltung, 2004. http://qse.ifs.tuwien.ac.at/courses/skriptum/script.htm
- Thomas Grechenig, Mario Bernhart, Roland Breiteneder, Karin Kappel: „Softwaretechnik: Mit Fallbeispielen aus realen Entwicklungsprojekten“, Pearson Studium, 2009
- Software Engineering Body of Knowledge, http://www.swebok.org, 2004.
- SOMMERVILLE, IAN: Software Engineering, 8th Edition, Addison Wesley, 2007.
- MARTIN FOWLER, KENT BECK, JOHN BRANT, WILLIAM OPDYKE, DON ROBERTS: Refactoring: Improving the Design of Existing Code, Addison-Wesley Object Technology Series, 1999, ISBN 0201485672
- FOWLER, MARTIN: UML Distilled Third Edition A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional, 2003.
- MESZAROS, GERARD: xUnit Test Patterns: Refactoring Test Code. Addison-Wesley, 2007.
- LIGGESMEYER, PETER: Software Qualität – Testen, Analysieren und Verifizieren von Software, Spektrum Verlag, 2009

Thank you for your attention!