



Web Engineering

Test

Prof. Dr. Alexander Knapp

Dr. Nora Koch

SS 07 (7) – 25.06.07

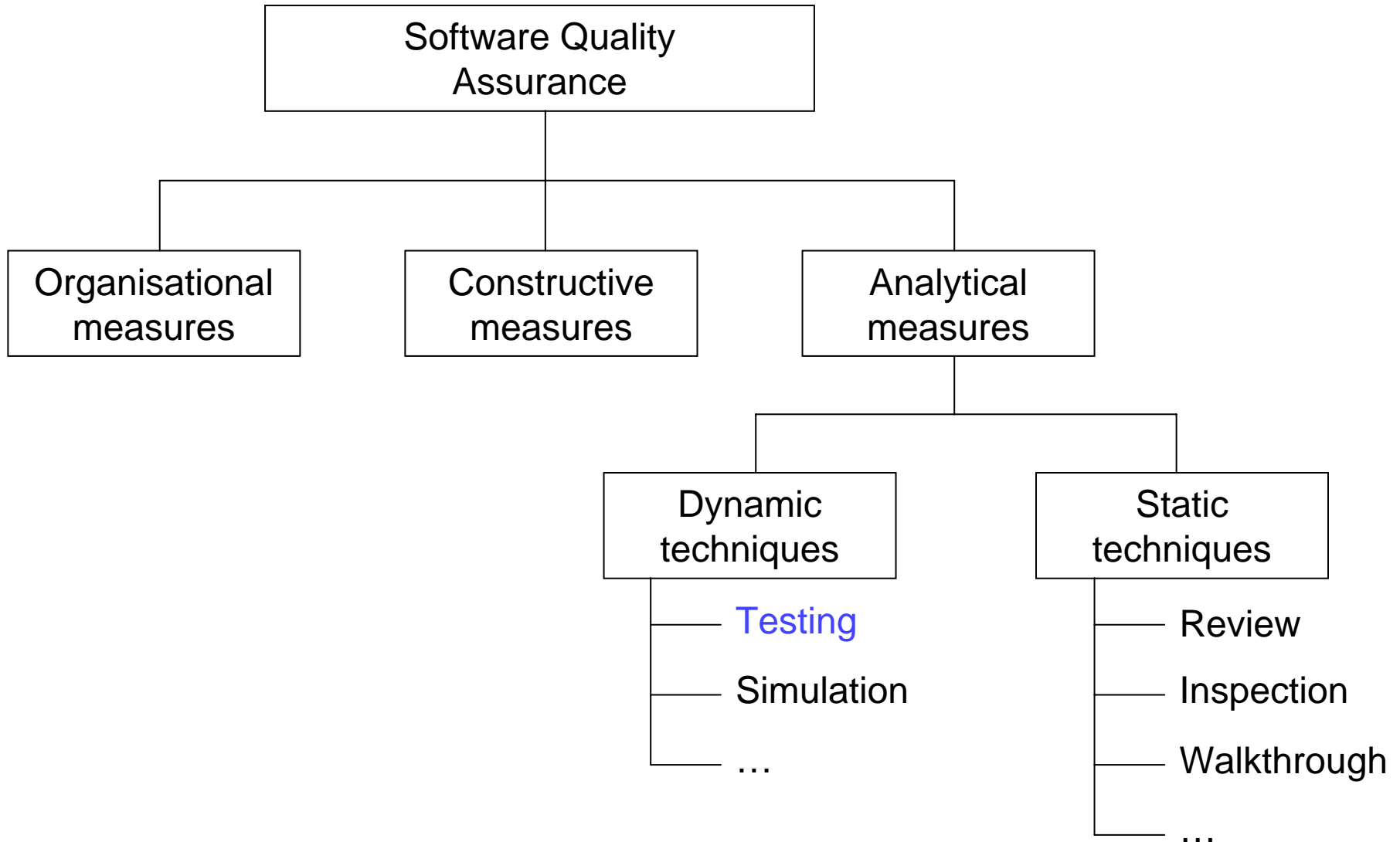
Ludwig-Maximilians-Universität München

- **Testing**
 - Web Engineering specifics

- **Test approaches**

- **Test dimensions**

- **Test methods and techniques**



- **Test case**

- a set of inputs, execution conditions, and expected results

- **Test**

- a set of test cases for a specific **object under test**
 - Web application, components of a Web application, ...

- **Error**

- “the difference between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition” (IEEE standard 610.12-1990)

Test Objectives

- **Find errors**

- ... not their absence
 - and test, in general, are unsuitable for this purpose
- test run successful, if errors detected
 - unsuccessful tests are “a waste of time”

- **Find errors early**

- errors in early development phases should be corrected as soon as possible, changes in later phases may incur high costs

- **Testing alone not sufficient for quality improvement**

- errors must be detected and removed

Test Levels

- **Unit tests**
 - smallest independent testable units (e.g., classes, Web pages, ...)
 - performed by developer
- **Integration tests**
 - evaluation of interactions between tested units
 - performed by tester and/or developer
- **System tests**
 - test of complete, integrated system
 - performed by test team
- **Acceptance tests**
 - system evaluation with client, in a environment which is at least close to the production environment
- **Beta tests (informal)**
 - product evaluation
 - performed by friendly users

Test Approaches

■ Conventional

- planning: quality goals, testing strategy, test environment, ...
- preparing: test techniques, tools, test cases (with test data)
- performing: set up of test infrastructure, run test cases, document and evaluate results
- reporting: summary, produce test reports

■ Agile

- entire team responsible for quality and testing
 - developers run (automated) unit tests
- eXtreme Programming (XP) practices
 - pair programming (continuous software inspection)
 - on-site customer (continuous acceptance testing)
 - continuous integration
 - test-first development

■ Quality characteristics

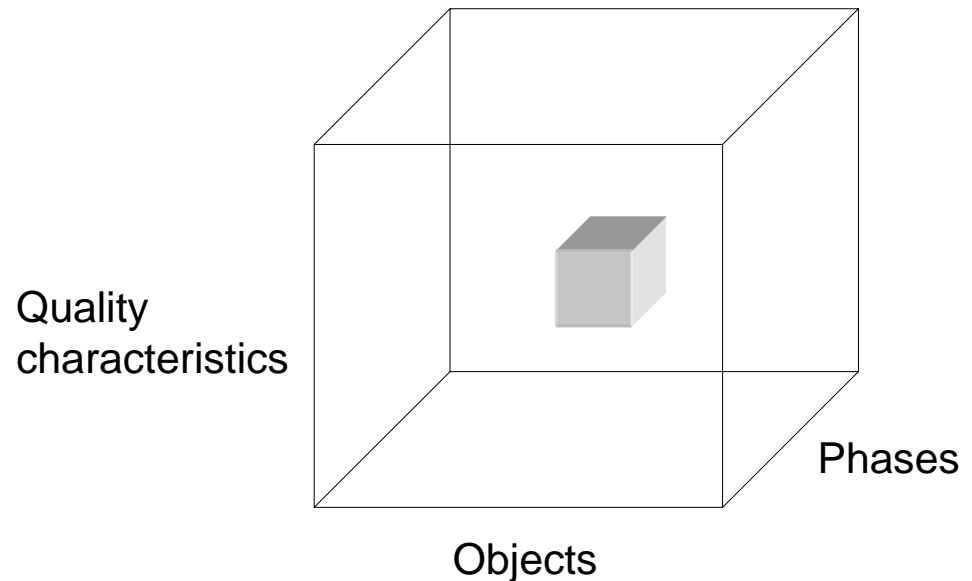
- functionality
- reliability
- efficiency
- usability

■ Objects

- content and structure
- infrastructure and environment
- functions

■ Phases

- requirements
- design and implementation
- acceptance and installation
- operation and maintenance



Test Specifics in Web Engineering

- **Content errors**
 - spell checking, proof reading
 - meta-information
- **Hypertext structure**
 - broken links
 - keeping information in “back in history”
- **Technical infrastructure**
 - different software components by different vendors (database, middleware, ...)
- **Aesthetics**
- **Multi-platform delivery**
 - different browsers
- **Global availability**
 - multi-linguality
- **Dominance of change**
 - customer expectations
 - ever-changing platforms and technologies

Testing Client Pages

■ Tests should cover

- compliance of the content displayed by the page to the one specified and expected by a user (e.g. rendering in the browser, images, ...)
- correctness of target pages pointed to by hyperlinks, i.e., right page should be returned
- existence of pending (broken) links
- correctness of the actions performed when a button is selected

■ Evaluation criteria: (white box) test coverage

- HTML statement coverage
- Web object coverage: image, applet, ...
- script block coverage: statement/branch/path coverage
- link coverage

Testing Server Pages

■ Tests should cover

- execution of servlets
- incorrect execution of data being stored in database
- failures due to incorrect links between pages
- defects in dynamic generated pages
- state management
 - back button
 - bookmarking during a transaction: navigation to (dynamically generated) page
 - several browser windows

■ Evaluation criteria: (white box) test coverage

- coverage of servlets and other Web object
- coverage of dynamically generated pages

■ Issues

■ Confidentiality

- who may access which data?

■ Authorisation

- management of access right, data encryption

■ Authentication

- for users and servers

■ Integrity

- how is information protected from being changed during transmission?

■ Accountability

- how are accesses logged?

■ Correctness not sufficient

- correct encryption, but display of confidential data
- incomplete or emergent behaviour

Load, Stress, and Continuous Testing

■ Load tests

- are response times and throughput met?
 - load profiles: access types, number of visits, peak times, visits per session, transaction per session, transaction mix

■ Stress tests

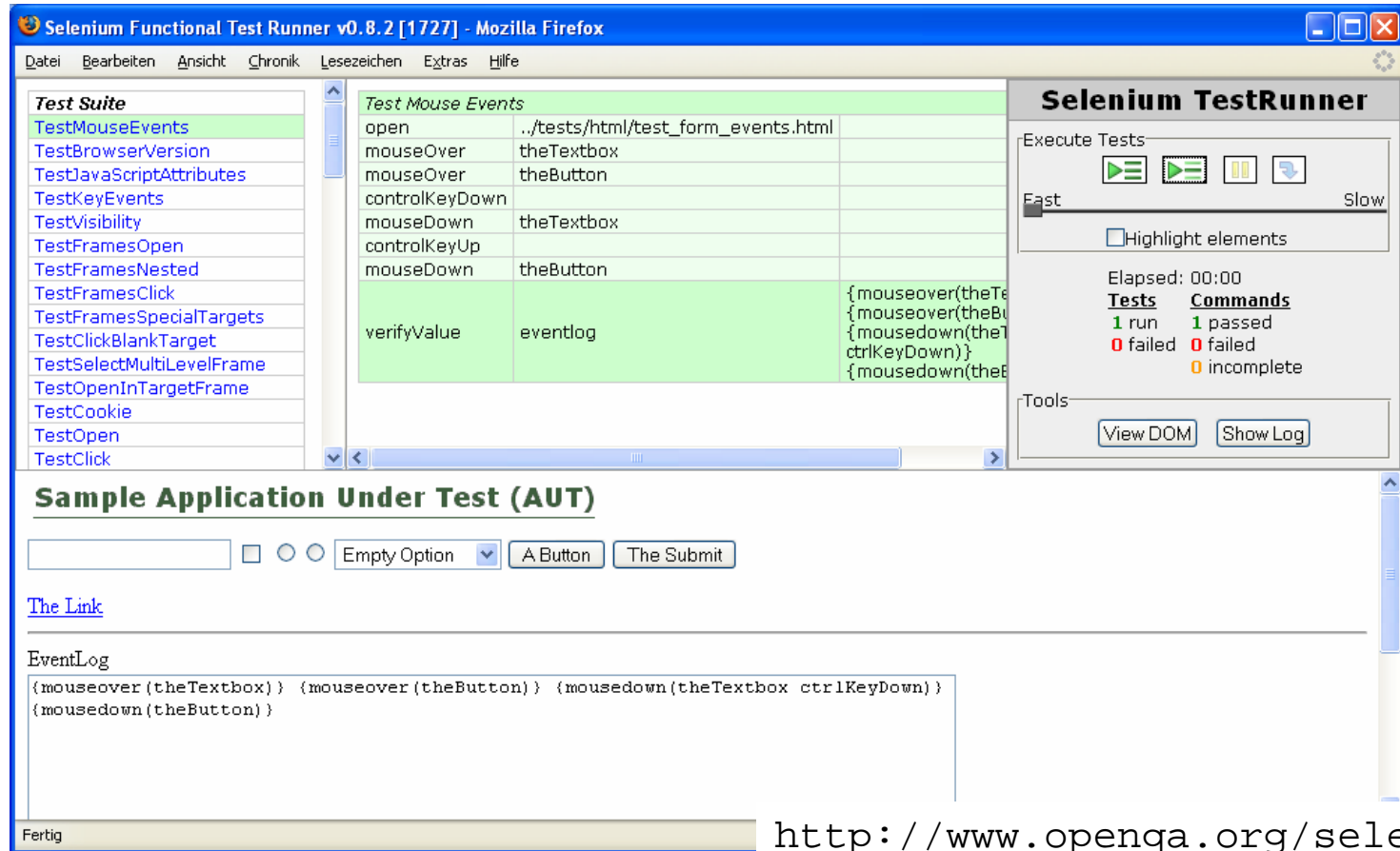
- reaction to extreme conditions
 - unrealistic overload, heavily fluctuating load
- does the system recover? does it respond with an error message when reaching a “flooding threshold”?

■ Continuous tests

- running system over a lengthy period of time
 - to produce “insidious” errors, like memory leaks

User Interface Testing

- Browser compatibility
- Functional testing



The screenshot displays the Selenium Functional Test Runner v0.8.2 interface within a Mozilla Firefox browser window. The interface is divided into several sections:

- Test Suite:** A list of test cases on the left, including TestMouseEvents, TestBrowserVersion, TestJavaScriptAttributes, TestKeyEvents, TestVisibility, TestFramesOpen, TestFramesNested, TestFramesClick, TestFramesSpecialTargets, TestClickBlankTarget, TestSelectMultiLevelFrame, TestOpenInTargetFrame, TestCookie, TestOpen, and TestClick.
- Test Mouse Events:** A table showing the sequence of events for the selected test:

Event	Target	Script
open	../tests/html/test_form_events.html	
mouseover	theTextbox	
mouseover	theButton	
controlKeyDown		
mousedown	theTextbox	
controlKeyUp		
mousedown	theButton	
verifyValue	eventlog	{mouseover(theText {mouseover(theBu {mousedown(theT ctrlKeyDown)} {mousedown(theB
- Selenium TestRunner Panel:** Contains controls for executing tests (Fast/Slow), a 'Highlight elements' checkbox, and a summary of test results:

Tests	Commands
1 run	1 passed
0 failed	0 failed
0 incomplete	0 incomplete
- Sample Application Under Test (AUT):** A web form with an input field, radio buttons, a dropdown menu labeled 'Empty Option', and buttons 'A Button' and 'The Submit'. Below the form is a link 'The Link' and an 'EventLog' section displaying the following log:


```
{mouseover(theTextbox)} {mouseover(theButton)} {mousedown(theTextbox ctrlKeyDown)}  
{mousedown(theButton)}
```

<http://www.openqa.org/selenium/>

Struts2: Testing Actions (1)

- **Unit tests with mock objects**
 - JUnit 3/4 (www.junit.org)
 - JMock 2 (www.jmock.org)

```
import org.apache.struts2.StrutsTestCase;
import org.jmock.Expectations;
import org.jmock.Mockery;

public class TestViewData extends StrutsTestCase {
    private Mockery context;
    private DataStore mockDataStore;
    private ViewData action;

    protected void setUp() throws Exception {
        context = new Mockery();
        mockDataStore = mockery.mock(DataStore.class);
        action = new ViewData();
        action.setDataStore(mockDataStore);
    }
}
```

mock object based on interface DataStore

Struts2: Testing Actions (2)

```
public void testViewData() throws Exception {
    Data existingData = new Data();
    existingData.setName("data");
    existingData.setId(1);
    Data expectedData = new Data();
    expectedData.setName("data");
    expectedData.setId(1);

    context.checking(new Expectations() {
        one(mockDataStore).getData(with(1));
        will(returnValue(existingData));
    });
    action.setId(1);
    String result = action.execute();
    context.assertIsSatisfied();
    assertEquals(Action.SUCCESS, result);
    assertEquals(expectedData, existingData);
}
```

what the mock object is
supposed to do

References

- Gerti Kappel, Birgit Pröll, Siegfried Reich, Werner Retschitzegger (eds.). *Web Engineering*. John Wiley & Sons, Chichester, 2006
- Rick Hower. *Web Site Test Tools and Site Management Tools*. 2007.
<http://www.softwareqatest.com/qatweb1.html>