INF3121 : Software Testing
26. 02. 2015
Lecture 6

Tool support for testing

UNIVERSITETET I OSLO

Lecturer: Raluca Florea

Overview

1. Types of test tools
2. Effective use of tools: benefits / risks
3. Introducing a tool into an organization

- 1.1 Test tool classification
- 1.2 Tool support for management of testing & tests
- 1.3 Tool support for static testing
- 1.4 Tool support for test specification
- 1.5 Tool support for test execution and logging
- 1.6 Tool support for performance and monitoring
- 1.7 Tool support for specific application areas
- 1.8 Tool support using other tools

1. Types of test tools

1.1 Test tool classification

- Tools are classified in this course according to the testing activities that they support.
  - one activity;
  - more than one activity, but classification falls under the main activity
- Testing tools can improve:

Efficiency of testing

by automating repetitive tasks.

Reliability of testing

by automating large data comparisons or simulating behavior.

Notes:

1.Some types of test tool can be intrusive - the tool itself can affect the outcome of the test. (i.e. timing measurements may be different depending on how you measure it with different performance tools).

The consequence of intrusive tools is called the probe effect.

2. Some tools offer support more appropriate for developers. Such tools are marked with “(D)” in this chapter.

1.2 Tools support for management of testing & tests

Characteristics :

- Support for the management of tests and the testing activities.
- Interfaces to:
  - test execution tools
  - defect tracking tools
  - requirement management tools.
- Support for traceability of tests, test results and incidents to source documents, such as requirements specifications.
- Generation of progress reports.
- Logging test results.
- Offer info on metrics related to the tests.

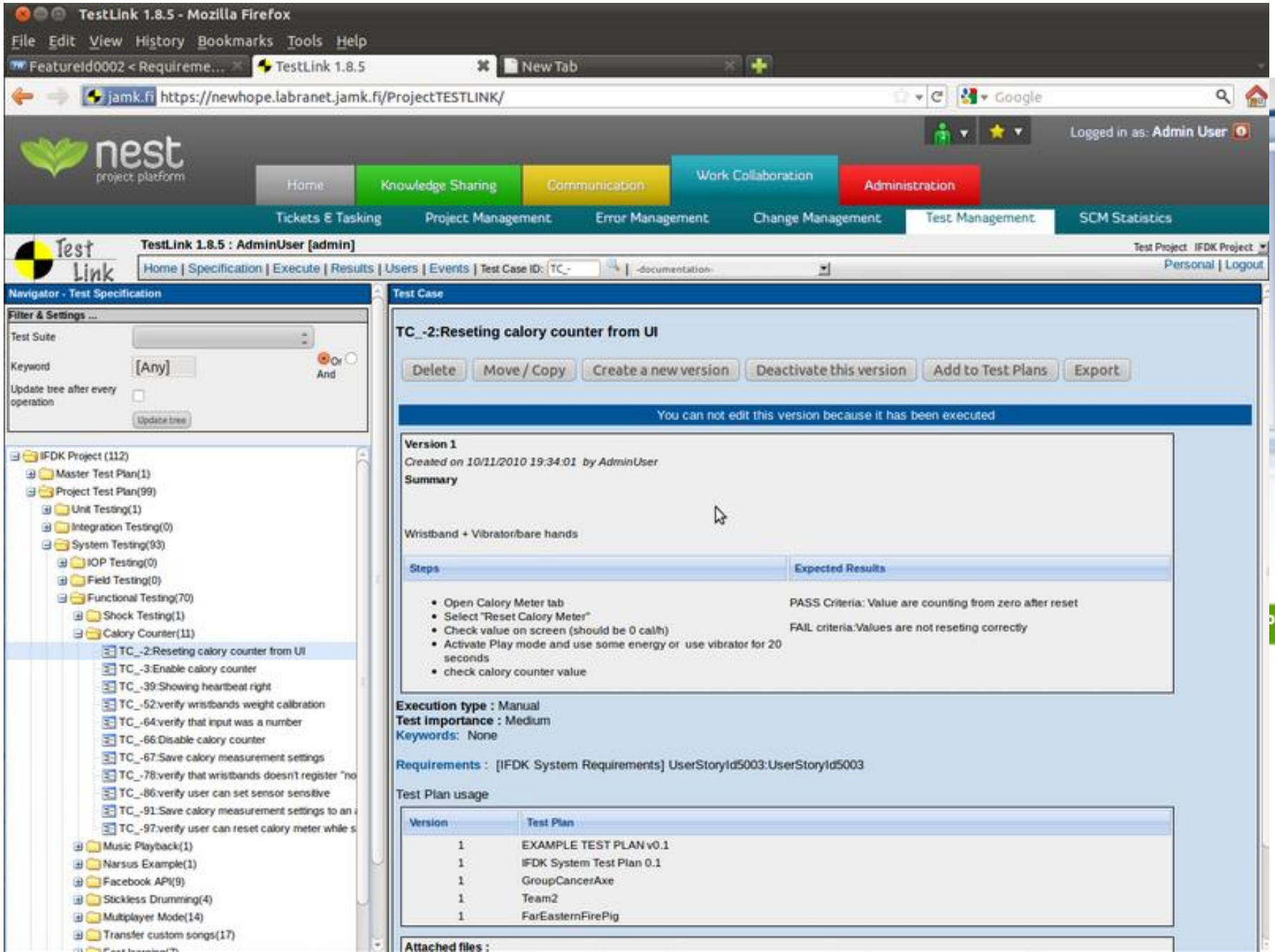
1.2 Tools support for management of testing & tests

Tools:

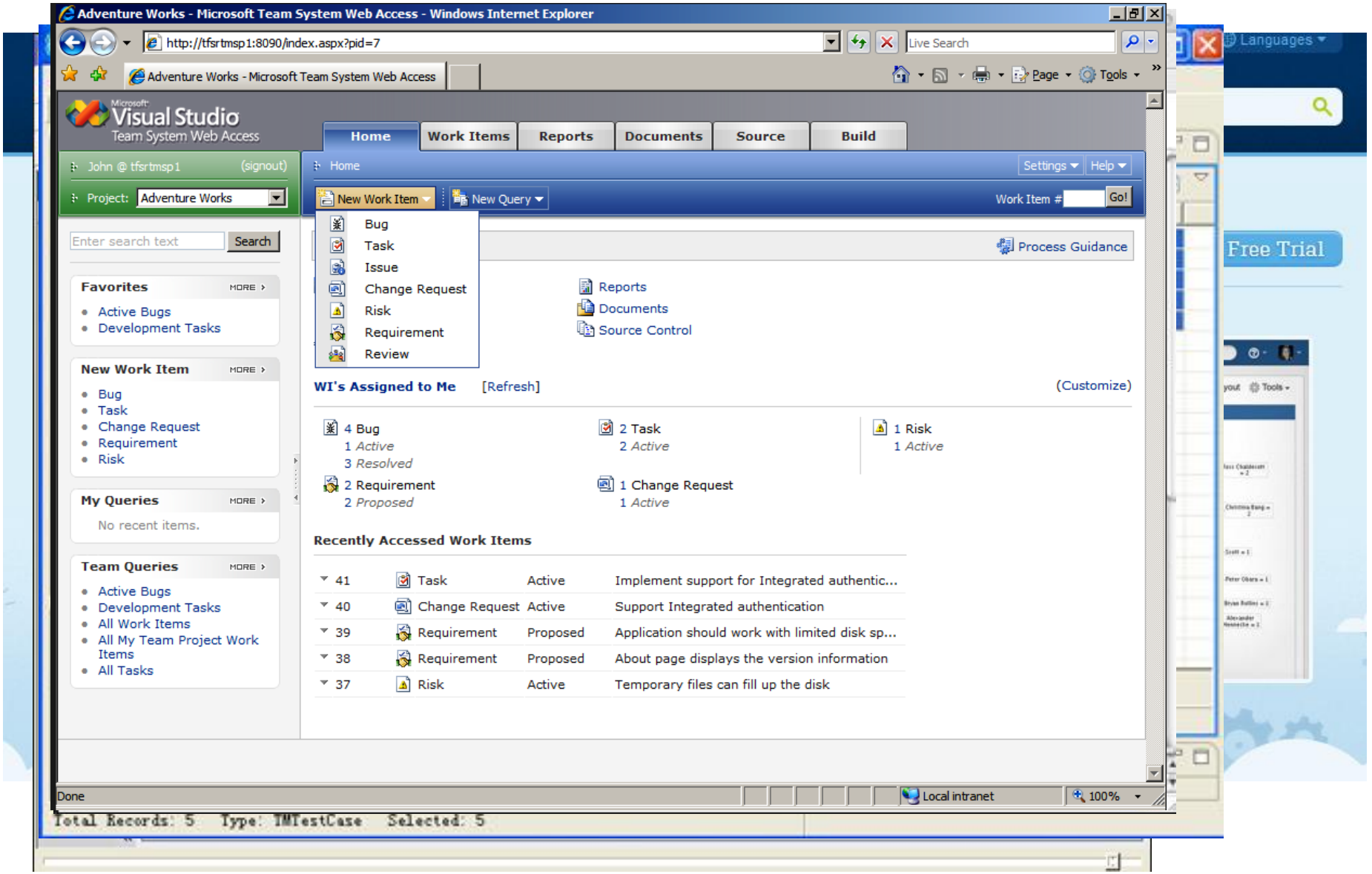
Requirements management tools	• store requirements • check for consistency and undefined (missing) requirements • allow prioritization • enable individual tests to be traceable to requirements
Incident management tools	• store and manage incident reports • support management of incident reports
Configuration management tools	• are necessary to keep track of different versions and builds of the SW and tests • are particularly useful when developing on more than one configuration of the HW/SW environment

1. Types of test tools

•Requirements management tools



•Incident management tools



•Configuration management tools

1.3 Tools support for static testing

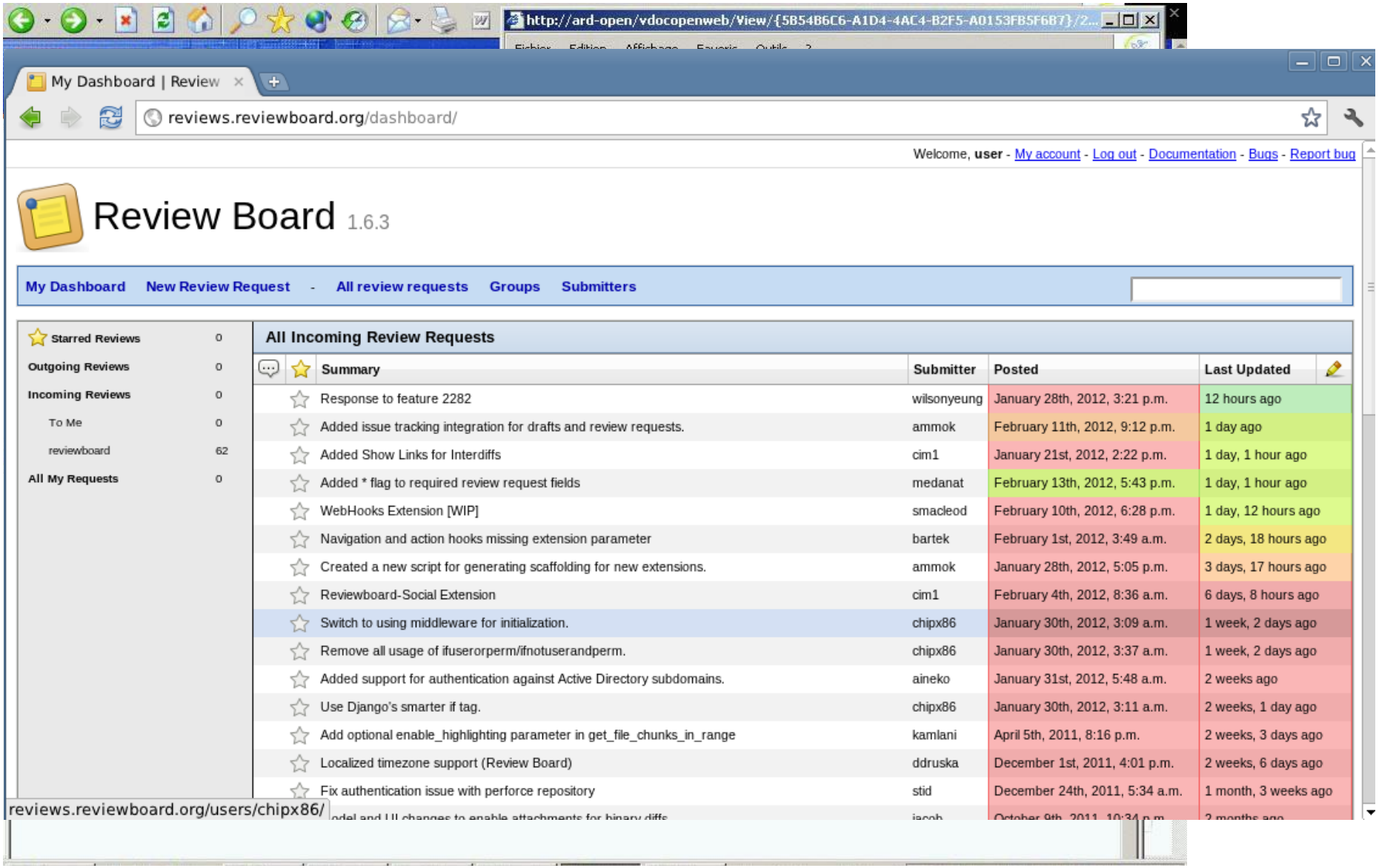
Review tools	• store information about review processes, • store and communicate review comments, report on defects and effort, • They can provide aid for online reviews, which is useful if the team is geographically dispersed.

Static analysis tools (D)	• support developers, testers and quality assurers in finding defects before dynamic testing.
	• Major purposes :
	  • The enforcement of coding standards.
	  • The analysis of structures and dependencies (e.g. linked web pages).
	  • Aiding in understanding the code.
	• Static analysis tools can calculate metrics from the code (e.g. complexity), which can give valuable information for planning or risk analysis.

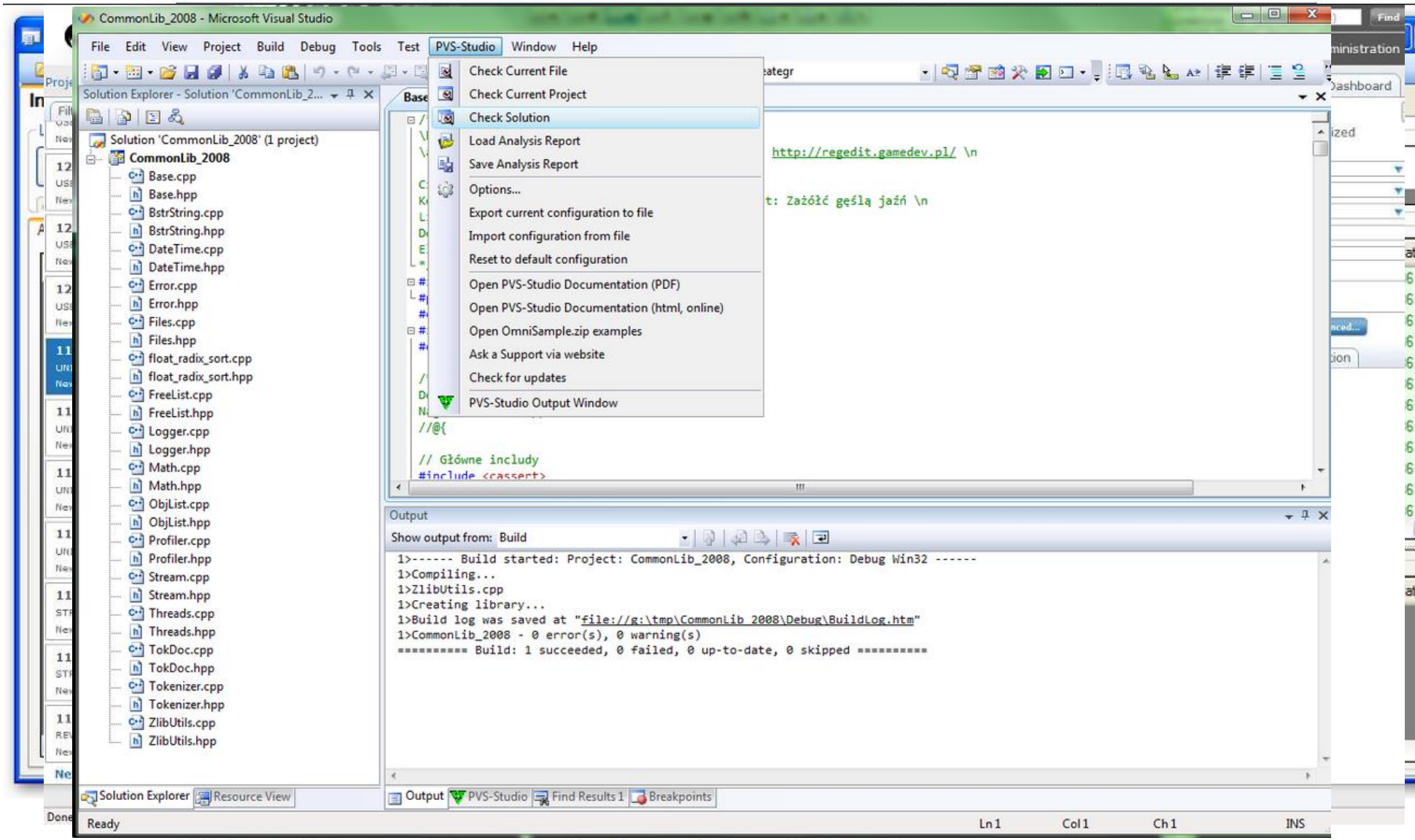
Modeling tools (D)	• Validate models of the software.

The major benefit of static analysis tools and modeling tools is the cost effectiveness of finding more defects at an earlier time in the development process. As a result, the development process may accelerate and improve by having less rework

•Review tools

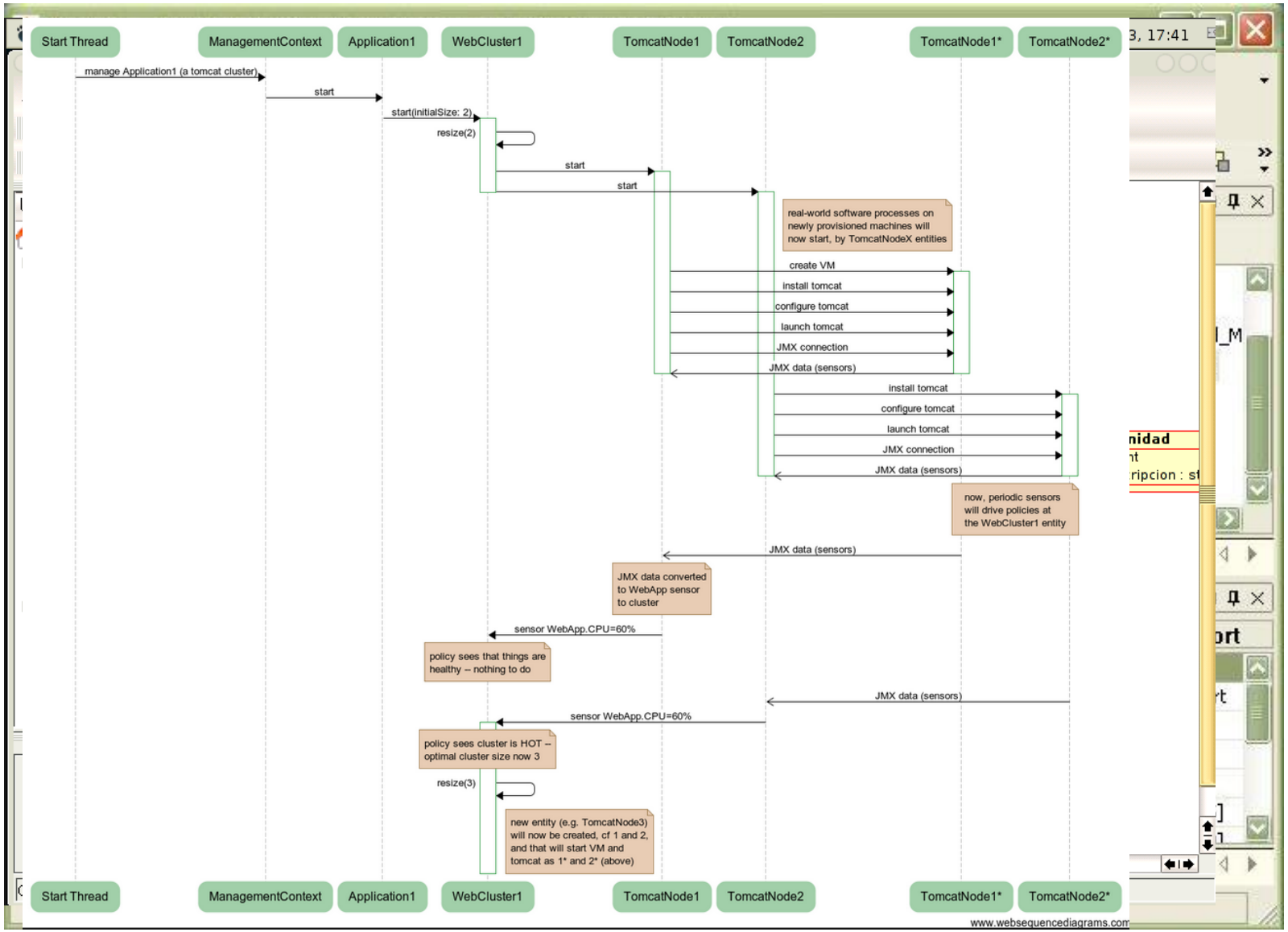


•Static analysis tools (D)



CodeSonar chart summarizing warning locations and classes for a project.

•Modeling tools (D)



1.4 Tools support for test specification

Test design tools

- Generate test inputs or executable tests
  - from requirements,
  - from a graphical user interface,
  - from design models (state, data or object)
  - or from code.
- This type of tool may generate expected outcomes as well (i.e. may use a test oracle)
  - They can save valuable time and provide increased thoroughness of testing because of the completeness of the tests that the tool can generate.

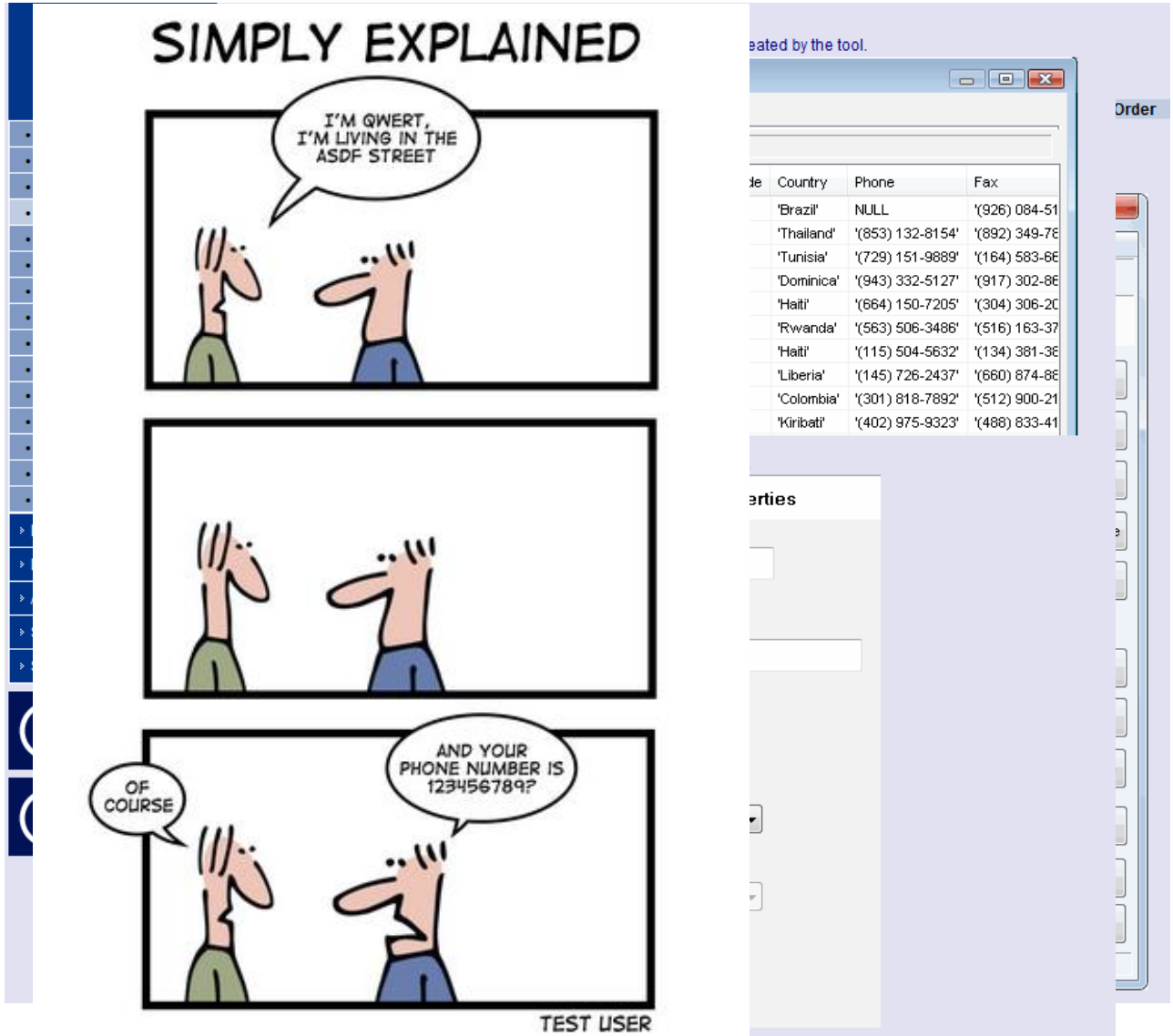
Test data preparation tools

- manipulate databases or files to set up test data to be used during the execution of tests
- benefit: they ensure that live data in a test environment is made anonymous, for data protection.

•Test design tools

#	Operating System	Language	Browser
1	Windows 8 64bit	French	Chrome latest
2	Windows 7 Ultimate	Italian	Internet Explorer 8
3	Windows XP home	French	Internet Explorer 8
4	Ubuntu 12.10	French	Chrome latest
5	Windows 8 64bit	German	Internet Explorer 10
6	Windows 8 64bit	Spanish	Safari latest
7	OS X 10.8	Chinese Simpl.	Chrome latest
8	Windows 8 64bit	Chinese Trad.	Firefox latest
9	Windows 8 64bit	Japanese	Opera latest
10	Windows 8 64bit	Korean	Internet Explorer 10
11	Windows 8 64bit	Italian	Internet Explorer 10
12	Windows 8 64bit	Chinese Simpl.	Internet Explorer 10
13	Windows 7 Ultimate	French	Internet Explorer 10
14	Windows 7 Ultimate	German	Chrome latest
15	Windows 7 Ultimate	Spanish	Safari latest
16	Windows 7 Ultimate	Chinese Simpl.	Firefox latest
17	Windows 7 Ultimate	Chinese Trad.	Opera latest
18	Windows 7 Ultimate	Japanese	Internet Explorer 8
19	Windows 7 Ultimate	Korean	Internet Explorer 8
20	Windows XP home	Italian	Chrome latest
21	Windows XP home	German	Safari latest
22	Windows XP home	Spanish	Firefox latest
23	Windows XP home	Chinese Simpl.	Opera latest
24	Windows XP home	Chinese Trad.	Internet Explorer 8
25	Windows XP home	Japanese	Chrome latest
26	Windows XP home	Korean	Chrome latest
27	Ubuntu 12.10	Italian	Firefox latest
28	Ubuntu 12.10	German	Opera latest
29	Ubuntu 12.10	Spanish	Chrome latest
30	Ubuntu 12.10	Chinese Simpl.	Opera latest
31	Ubuntu 12.10	Chinese Trad.	Chrome latest
32	Ubuntu 12.10	Japanese	Firefox latest
33	Ubuntu 12.10	Korean	Firefox latest
34	OS X 10.8	French	Safari latest
35	OS X 10.8	Italian	Firefox latest
36	OS X 10.8	German	Opera latest
37	OS X 10.8	Spanish	Opera latest
38	OS X 10.8	Chinese Trad.	Safari latest
39	OS X 10.8	Japanese	Safari latest
40	OS X 10.8	Korean	Safari latest
41	Windows XP home	French	Firefox latest
42	Ubuntu 12.10	French	Opera latest
43	Windows 7 Ultimate	Italian	Safari latest
44	OS X 10.8	Italian	Opera latest
45	Windows 7 Ultimate	German	Internet Explorer 8
46	Windows 8 64bit	German	Firefox latest
47	Windows 7 Ultimate	Spanish	Internet Explorer 8
48	Windows 8 64bit	Spanish	Internet Explorer 10
49	Windows XP home	Chinese Simpl.	Internet Explorer 8
50	OS X 10.8	Chinese Simpl.	Safari latest
51	Windows 7 Ultimate	Chinese Trad.	Internet Explorer 10
52	Windows 8 64bit	Japanese	Internet Explorer 10
53	Windows 8 64bit	Korean	Opera latest

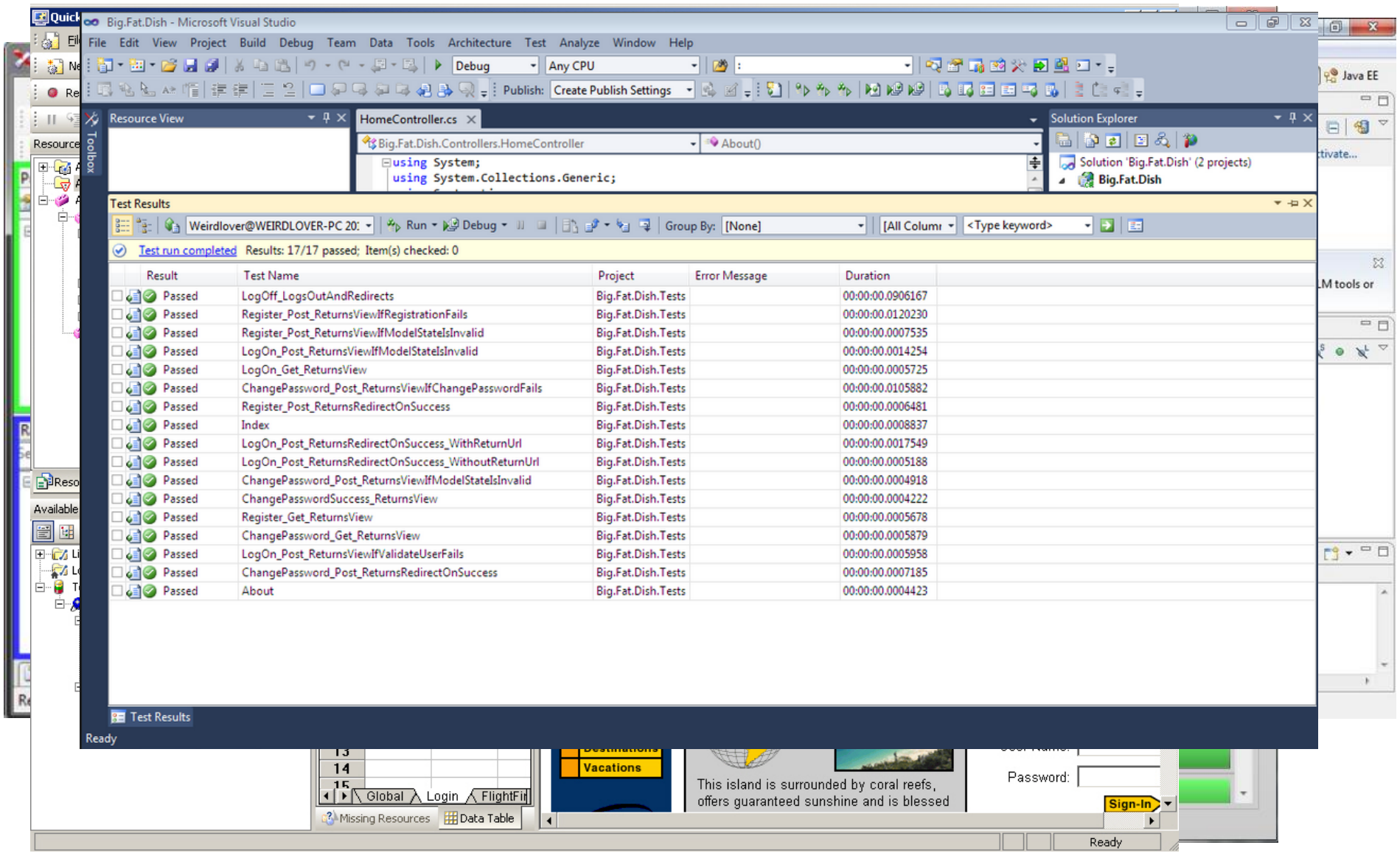
•Test data preparation tools



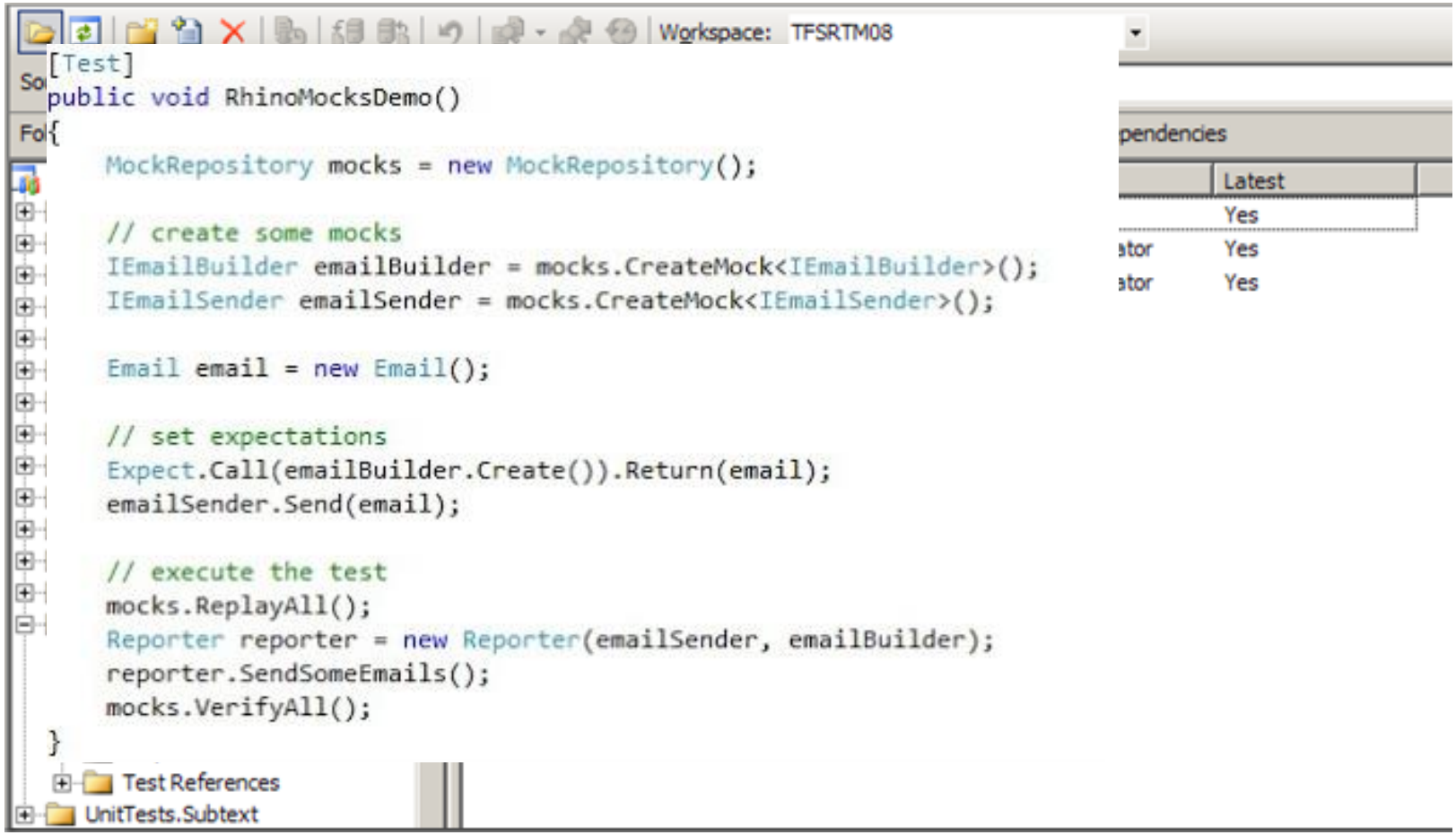
1.5 Tools support for test execution & logging

Test execution tools	• Enable tests to be executed automatically using stored inputs & expected outcomes • The scripting language allows to manipulate the tests with little effort(i.e. repeat the test with other data) • Can also be used to record tests(capture &playback tools)
Test harness/unit test framework tools (D)	• Facilitate the test of components of a system - simulating the environment in which that test object will run. • They may be called unit test tools when they have a particular focus on the component test level.
Test comparators	• Determine differences between files, databases or test results • Test execution tools include dynamic comparators, but post-execution comparison may be done by a separate comparison tool. • A test comparator may use a test oracle, especially if it is automated.
Coverage measurement tools (D)	• Can be intrusive or non intrusive (depends on the measurement technique used) • Measure the percentage of specific types of code structure that have been exercised
Security tools	• Search for specific vulnerabilities of the system

•Test execution tools



•Test harness/unit test framework tools (D)



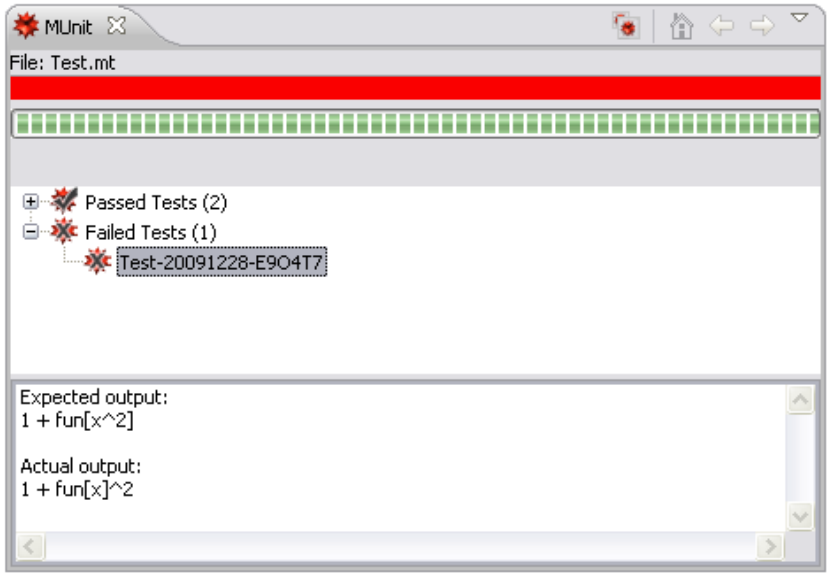
•Test comparators

Mathematica Development User Guide > Tasks > Wolfram MUnit Tester

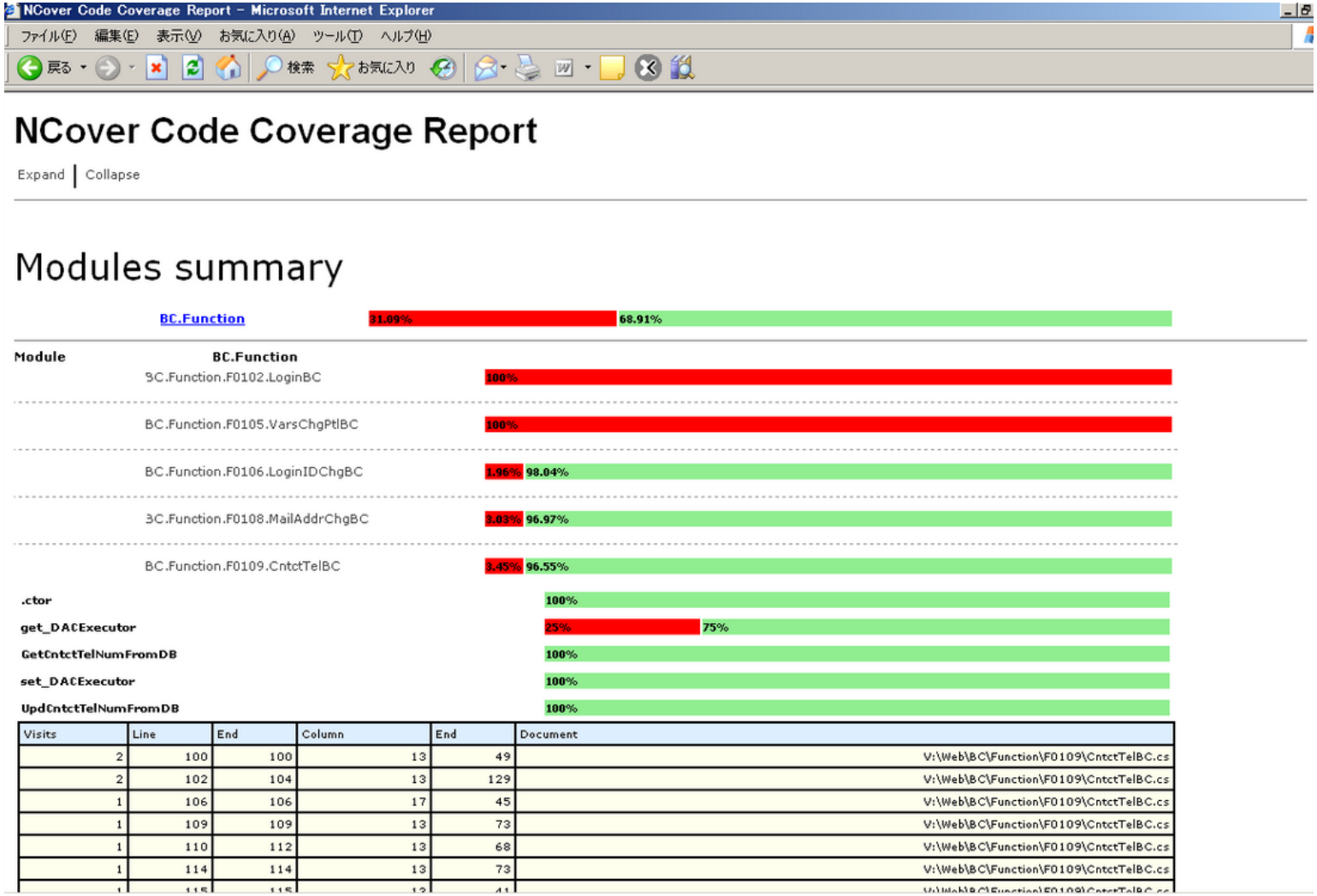
The Test Comparator

The test comparator is a useful feature for seeing exactly what the error is in a test failure.

First, you should run a test file or test suite. Then select one of the test failures: this should show the expected and actual results in the test report. A sample is shown in the following.



•Coverage measurement tools (D)



NCover Code Coverage Report – Microsoft Internet Explorer

ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)

戻る 検索 お気に入り

NCover Code Coverage Report

Expand | Collapse

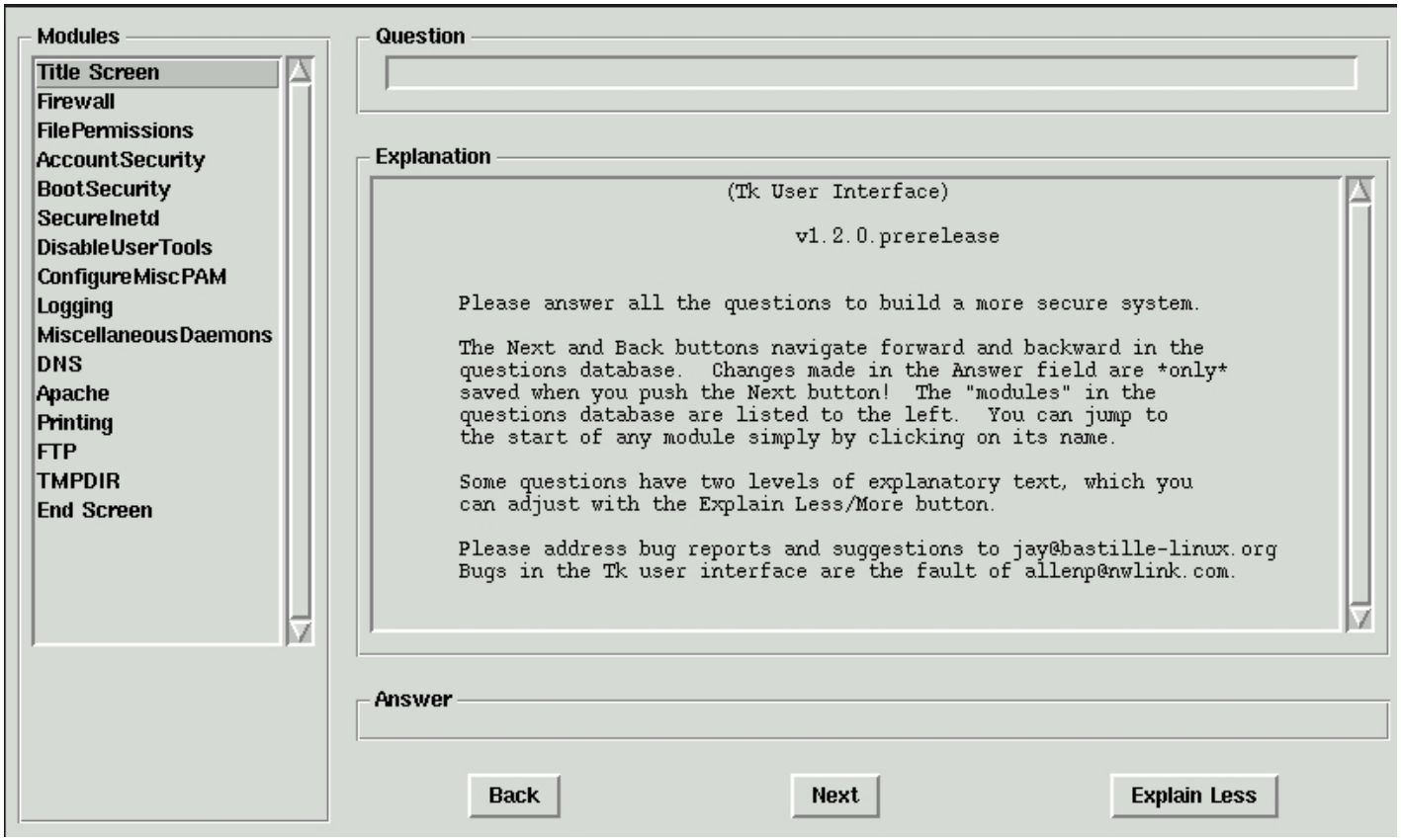
Modules summary

BC.Function 31.09% 68.91%

Module	BC.Function	
	BC.Function.F0102.LoginBC	100%
	BC.Function.F0105.VarsChgPtlBC	100%
	BC.Function.F0106.LoginIDChgBC	1.96% 98.04%
	BC.Function.F0108.MailAddrChgBC	3.03% 96.97%
	BC.Function.F0109.CntctTelBC	3.45% 96.55%
.ctor		100%
get_DACExecutor		25% 75%
GetCntctTelNumFromDB		100%
set_DACExecutor		100%
UpdCntctTelNumFromDB		100%

Visits	Line	End	Column	End	Document
2	100	100	13	49	V:\Web\BC\Function\F0109\CntctTelBC.cs
2	102	104	13	129	V:\Web\BC\Function\F0109\CntctTelBC.cs
1	106	106	17	45	V:\Web\BC\Function\F0109\CntctTelBC.cs
1	109	109	13	73	V:\Web\BC\Function\F0109\CntctTelBC.cs
1	110	112	13	68	V:\Web\BC\Function\F0109\CntctTelBC.cs
1	114	114	13	73	V:\Web\BC\Function\F0109\CntctTelBC.cs

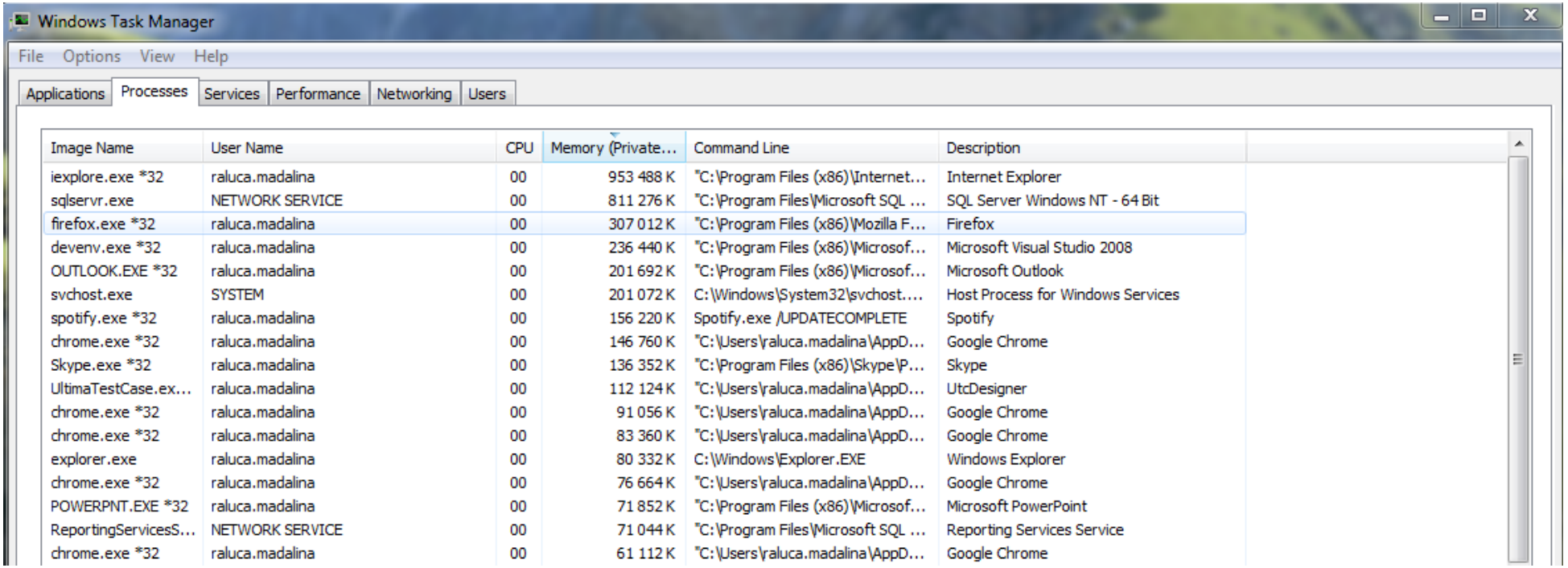
•Security tools



1.6 Tools support for performance & monitoring

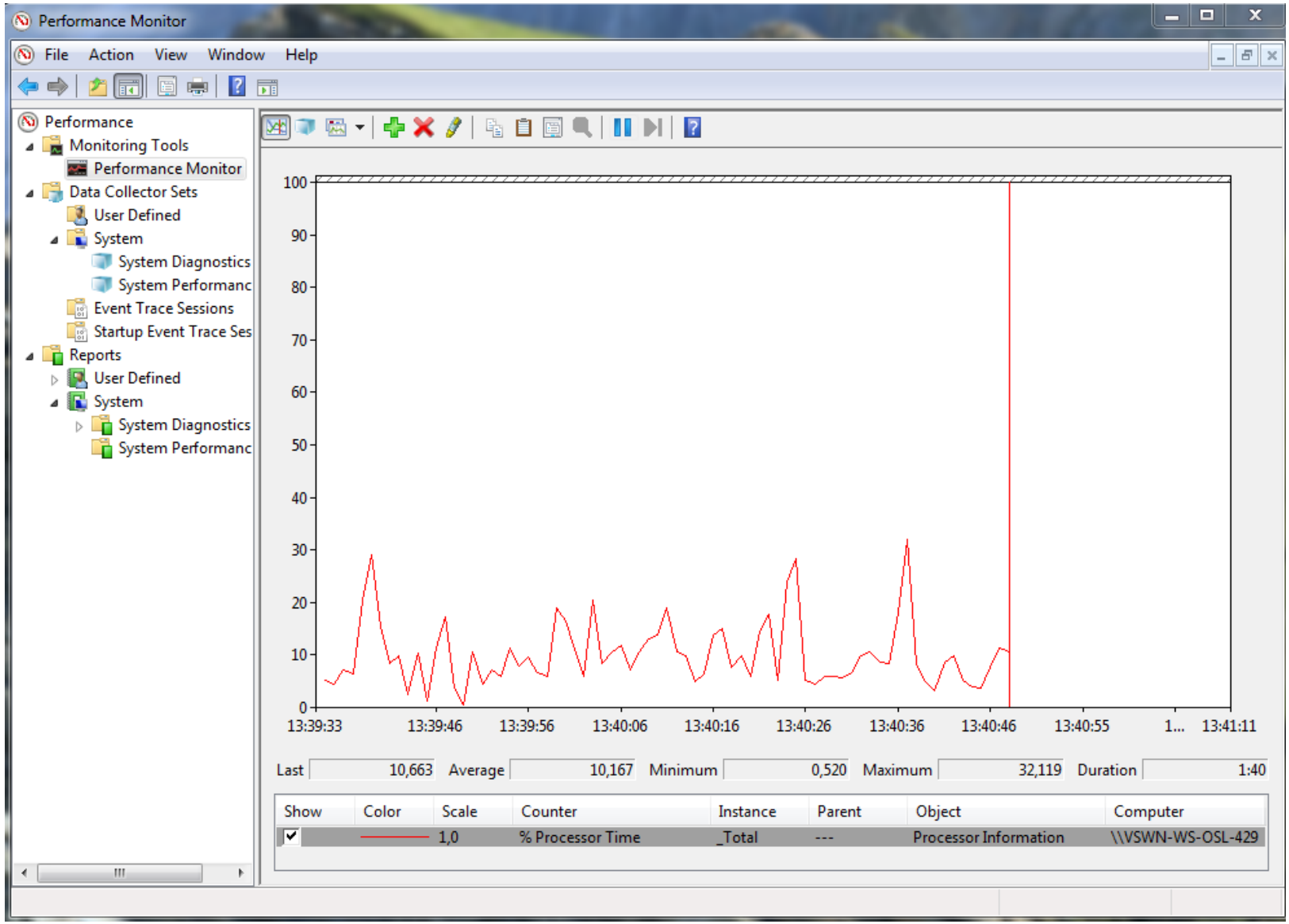
Dynamic analysis tools (D)	• find defects that appear only when software is executing (i.e. memory leaks) • They are typically used in component and component integration testing
Performance testing/load testing/stress testing tools	• Monitor and report on how a system behaves under a variety of simulated usage conditions. • They simulate a load on: • an **application**, a **database**, or a **system environment**. • They are often based on automated repetitive execution of tests, controlled by parameters.
Monitoring tools	• Are not strictly testing tools but provide information that can be used for testing purposes. • Analyze, verify and report on usage of specific system resources, and give warnings of possible service problems.

•Dynamic analysis tools (D)



- **Performance testing/load testing/stress testing tools**

•Monitoring tools



1.7 Tools support for specific application areas

- There are tools specialized for use in a particular type of application.

Example:

- performance testing tools specifically for web-based applications
- and dynamic analysis tools specifically for testing security aspects.

- Example of targeted areas: embedded systems.

1.8 Tools support using other tools

- This is not a complete list of tools of this category.
- Testers may use:
  - word processor
  - spreadsheets

  as a testing tool, but they are often used to store:
  - test designs
  - test scripts
  - test data.

- Testers may also use SQL to set up and query databases containing test data.

- Tools used by developers when **debugging,** to help localize defects and check their fixes, are also testing tools.

- It is a good idea to look at any type of tool available to you for ways it could be used to help support any of the testing activities.

- 2.1 Potential risks & benefits
- 2.2 Special considerations

2. Effective use of tools

2.1 Potential benefits and risks

- Simply purchasing or leasing a tool does not guarantee success with that tool.
- Each type of tool may require additional effort to achieve real and lasting benefits.

Potential benefits of using tools :

- Reduced repetitive work (running regression tests, re-entering the same test data. Etc)
- Greater consistency and repeatability (tests executed by a tool, tests derived from requirements).
- Objective assessment (static measures, coverage).
- Ease of access to information about tests or testing (statistics / graphs about test progress, incident rates, performance)

Risks of using tools:

- Unrealistic expectations for the tool (functionality & ease of use).
- Underestimating time, cost and effort for the introduction of a tool (training, external expertise).
- Underestimating the time and effort needed to achieve significant and continuing benefits from the tool
- Underestimating the effort required to maintain the test assets generated by the tool.
- Over-reliance on the tool (replacement where manual testing would be better).

2.2 Special considerations

1. Test execution tools

- This type of tool often requires significant effort in order to achieve significant benefits.
- Capturing tests by recording the actions of a manual tester seems attractive, but this approach does not scale to large numbers of automated tests. This type of script may be unstable when unexpected events occur.
  - Data-driven approach: separates out the test inputs (the data) and uses a more generic script that can read the test data and perform the same test with different data.
- In a keyword-driven approach: the spreadsheet contains keywords with the actions to be taken (also called action words), and test data. Testers can then define tests using the keywords.

2. Performance testing tools

- These tools need tester with expertise in performance testing to design the tests and interpret results.

3. Static analysis tools

- These tools applied to source code can enforce coding standards, but if applied to existing code may generate a lot of messages
- A gradual implementation with initial filters to exclude some messages would bean effective approach.

4. Test management tools

- They need to interface with other tools or spreadsheets in order to produce information in the best format for the current needs of the organization.
- The reports need to be designed and monitored so that they provide benefit.

3. Introducing a tool into a organization

- The main considerations in selecting a tool for an organization include:
  - Assess the organizational maturity, strengths and weaknesses
  - Evaluate against clear requirements and objective criteria.
  - A proof-of-concept to test the required functionality and determine whether the product meets its objectives.
  - Evaluation of the vendor (including training, support and commercial aspects).
  - Identification of internal requirements for coaching and mentoring in the use of the tool.

- Introducing the selected tool into an organization starts with a pilot project, with the following objectives:
  - Learn more detail about the tool.
  - Evaluate how the tool fits with existing processes and practices, and determine what would need to change.
  - Decide on standard ways of using and maintaining the tool and the test
  - Assess whether the benefits will be achieved at reasonable cost.

- Success factors for the deployment of the tool within an organization include:
  - Roll out the tool to the rest of the organization incrementally.
  - Adapt and improve processes to fit with the use of the tool.
  - Provide training and coaching/mentoring for new users.
  - Define usage guidelines.
  - Implement a way to learn lessons from tool use.
  - Monitor the tool use and benefits.