6. Test Tools

(CAST - Computer Aided Software Testing)

Contents

Warning

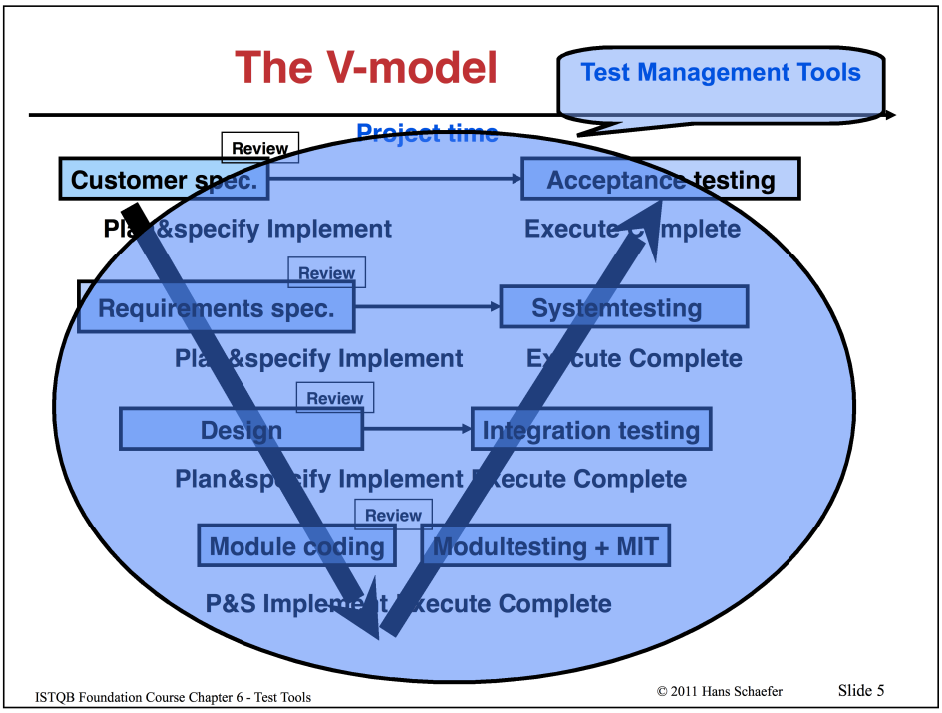
The most important testing tool is your brain!

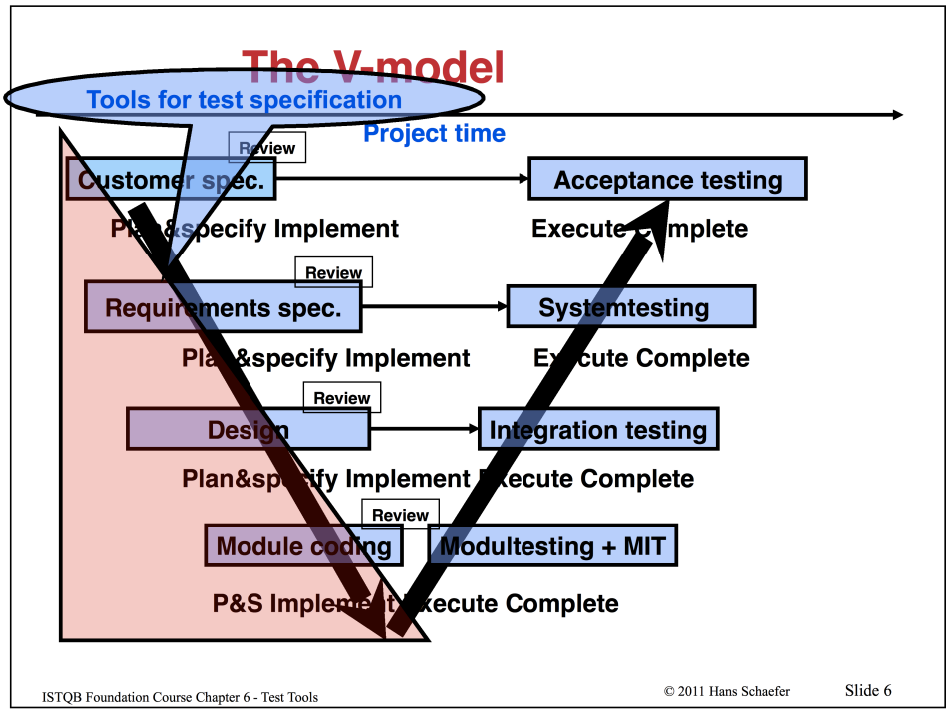
Software tools do not replace it!

1 Types of Tools

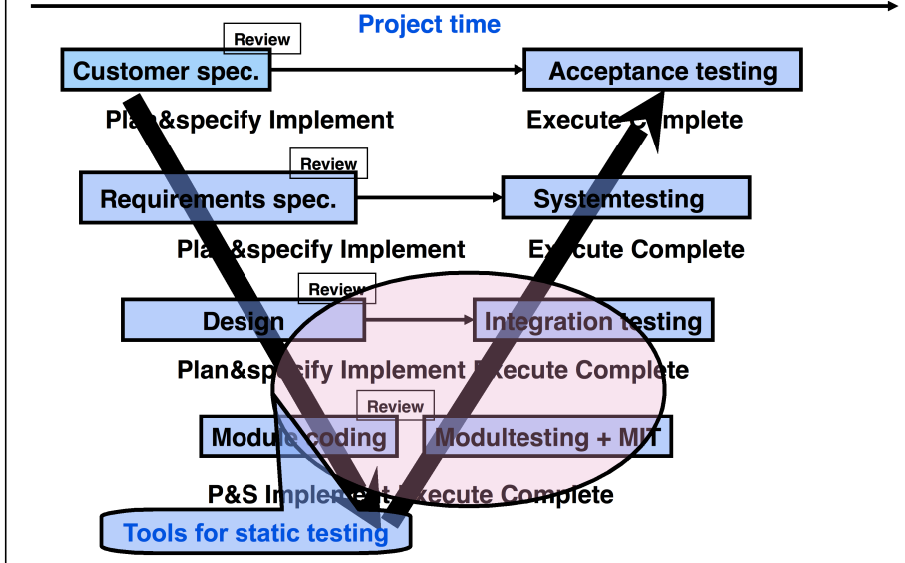
1 Tools for test management and control
2 Tools for test specification
3 Tools for static testing
4 Tools for test execution and logging
5 Tools for performance and monitoring (non-functional testing)
6 Tools for Specific Testing Needs

In addition many tools covering several areas or at the boundary between areas.

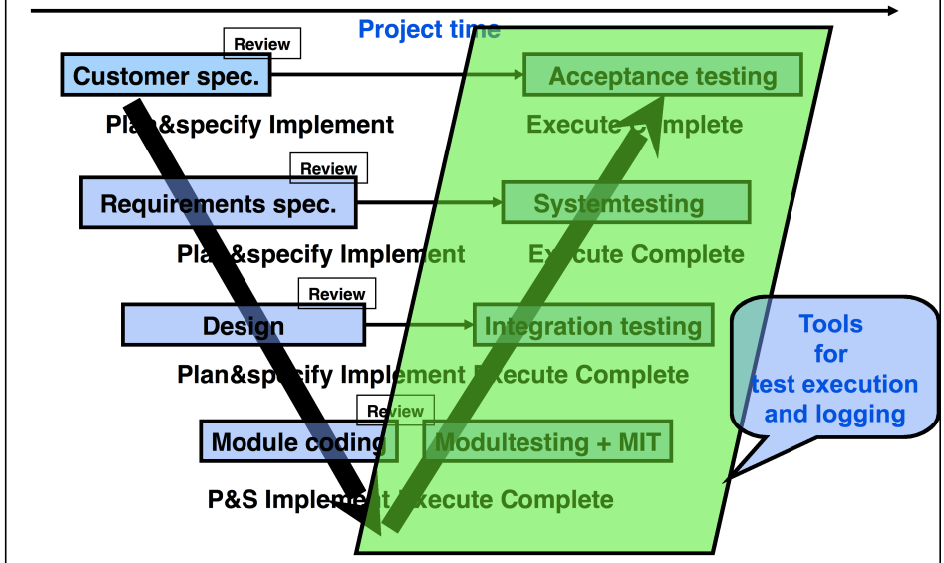

The V-model
Test Management Tools
Project time
Review
Customer spec.
Acceptance testing
Plan&specify Implement
Execute Complete
Review
Requirements spec.
Systemtesting
Plan&specify Implement
Execute Complete
Review
Design
Integration testing
Plan&specify Implement
Execute Complete
Review
Module coding
Modultesting + MIT
P&S Implement
Execute Complete
ISTQB Foundation Course Chapter 6 - Test Tools
© 2011 Hans Schaefer
Slide 5

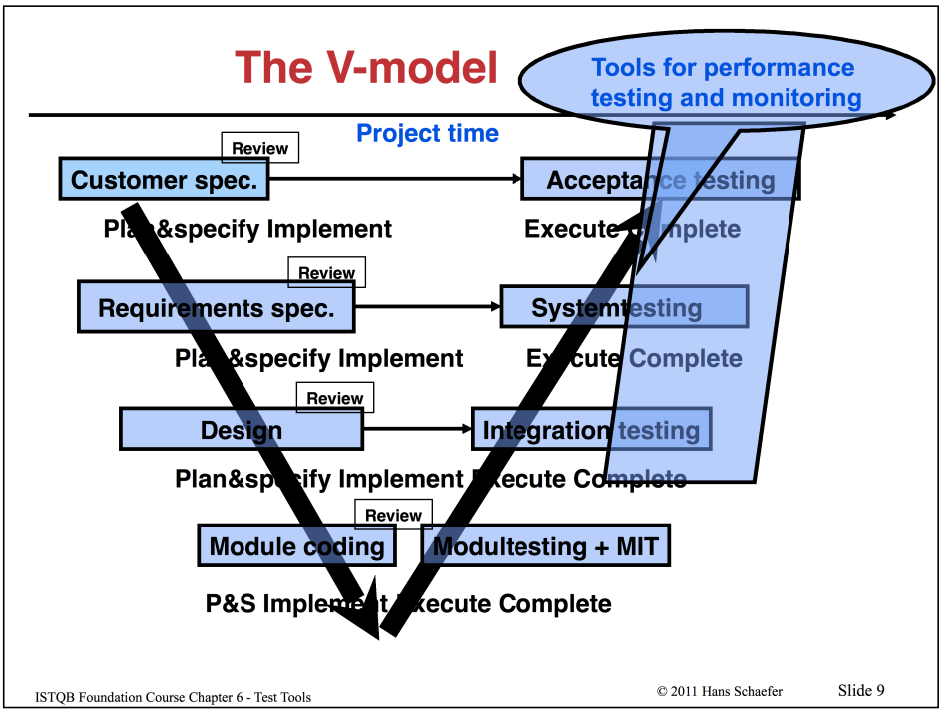

The V-model
Tools for test specification
Project time
Review
Customer spec.
Acceptance testing
Plan&specify Implement
Execute Complete
Review
Requirements spec.
Systemtesting
Plan&specify Implement
Execute Complete
Review
Design
Integration testing
Plan&specify Implement
Execute Complete
Review
Module coding
Modultesting + MIT
P&S Implement
Execute Complete
ISTQB Foundation Course Chapter 6 - Test Tools
© 2011 Hans Schaefer
Slide 6

The V-model



The V-model





Why Use Tools

- Some activities cannot be done without tools (f.ex. performance test)
- Improves test effectiveness by automating error-prone activities (repeated test execution)
- Improves reliability of the test (handling of large or complex data collections)
- Eases overview and control
- Can make testing cheaper
- Can make testing more systematic

Terminology: Test framework

- A tool to make execution of a test object possible
- A developer tool to isolate a component from its surroundings (or that replaces or generates drivers and stubs)
- Otherwise used in industry for:
  - Reusable and extensible testing libraries to build testing tools (such tools are also called ”test harness”)
  - A type of design of test automation (like data-driven, keyword-driven)
  - The process for executing tests

1 Tools For Test Management And Control

Test Management
Requirements Management
Incident Management
Configuration Management

Tools For Test Management

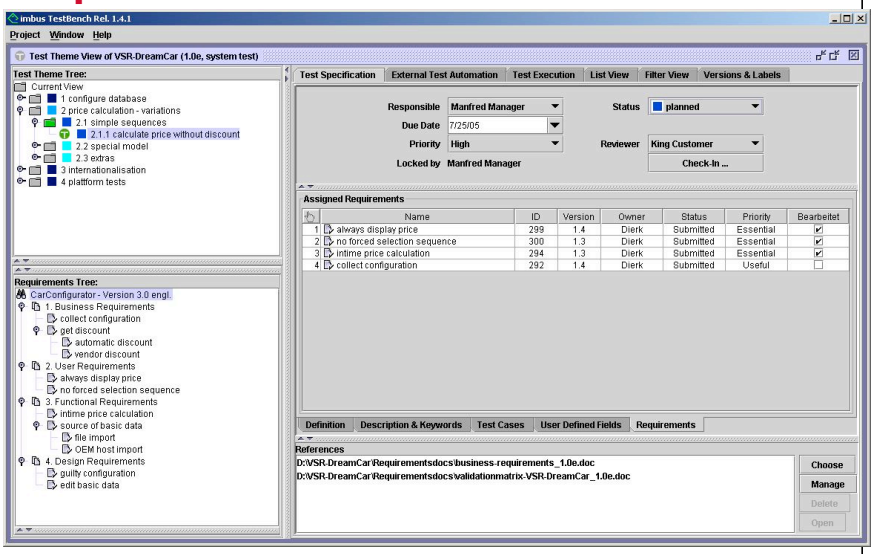
Can be used for all test activities throughout the life cycle

Overview of all test activities through the whole life cycle

- **Overview over test cases and priorities**
- **Interface for executing tests, tracking defects and managing requirements**
- **Status of test cases**
- **Resources and time for test work**
- **Import of requirements - link to test cases (traceability)**
- **Logging of test results**
- **Generating progress reports**
- **Possibilities for analysis -> control, improvement of the test process**

Example: HP Quality Center, imbus Testbench, SQS Test, IBM/Rational Testmanager, Borland Silktest, Microsoft Visual Studio Team Test, testuff

Example imbus Testbench



Requirements Management - Control Of Requirements

Database of requirements
Prioritization
Tracing and checking of requirements test coverage
Help identifying inconsistent or missing requirements
Maintenance of requirements

These are not test tools, but very helpful for testers

Example: Telelogic DOORS, IBM/Rational RequisitePro, RTM, Seapine TestTrack RM

Problem/Incident Management

Problems / failures / faults etc:
- Register
- Administrate
- Prioritize
- Assign responsible person for research and fixing
- Follow up (history, status)
- List (problem reports)

Statistical analysis
Problem handling model
Integration with test management tools
Integration with communication tools

Example: Bugzilla, Jira, FogBugz

Configuration Management

Not strictly a test tool

Very important basis for testing

Versions and builds of product under test, documentation and test material

Build / collection of test objects, test environments and test results

Traceability between test material, test execution and product(variants)

Especially useful when development is for more than one environment (platform)

Example: CVS, Subversion

2 Tools For Test Specification

Tools for test design and test data generators

- From database
  - Generates databases and files from schema
  - Filters contents from existing bases
  - Translates data between different formats
- From code
  - In order to cover all code
  - Problem: Missing code not found
  - Problem: Code itself is test basis :-(
- From interfaces
  - Finds which data are used in interfaces
  - Can make a GUI-map
  - Can generate erroneous input
- From (requirements) specification or models
  - Formal notation necessary
  - UML, MSCs, state charts
  - "Model based testing"
- Generators for combinations
- Random generation

Creativity of a test designer cannot be replaced by tools. But "standard tests" can be generated.

Expected results/ test oracle: Very difficult to generate.

Tools For Test Specification And Design

Examples: allpairs (www.satisfice.com), CASEMAKER, Microsoft PICT, Classification Tree Editor

AppLabs.com DFT (to fill databases), Compuware FileAid

CASEMAKER

3 Tools for Static Testing

Tools for review support (review process support tools (Crucible fra atlassian.com))
- Managing and organizing reviews, even online reviews

Static code analysis (for developers)
- Collection of measurement data (complexity) -> risk analysis, planning
- Control- and data flow bugs
- Code standard violations
- Link check (web pages)
- Compatibility check (platforms)

Warning-level in compiler!

Open source tool: FindBugs (Java)

Modeling tools - check models against formal specifications or check consistency and open ends in models (not many available, not much used!).
- Such tools can alos be used to generate test cases.

List of check points in a static analysis tool (Not for exam)

- Control flow analysis to find errors such as unreachable code, endless loops, violations of recursion conventions, and loops with multiple entry and exit points;
- Warning for dangerous coding constructs like typical errors (C: if (A=B)...);
- Data use analysis to find errors such as data used before initialization, variables declared but not used, and redundant writes;
- Range bound analysis to find errors such as array indices outside the boundaries of the array;
- Interface analysis to find errors such as mismatches in argument lists between called modules and calling modules;
- Information flow analysis to check the dependency relations between input and output;
- Verification of conformance to language standards (e.g. ANSI C);
- Verification of conformance to project coding standards, such as departures from naming conventions;
- Metrics collection such as code volume analysis, such as counts of the numbers of modules and the number of lines of code in each module;
- Complexity analysis, such as measurements of cyclomatic complexity and integration complexity.

Static analysis tools must be configured!
They should not give too many false warnings

4 Tools for Test Execution and Logging

Comparator
Test environment generator (test harness generator)
Test execution tool
Tool for measuring test coverage (for developers)
Some security tools

Tools for Test Execution and Logging

Comparator
- During test execution and afterwards
- Test result against expected result (test oracle)
- Any comparison (files, databases)
- Often integrated in a test execution automation tool
- May be configured (ignore, intelligent comparison)

-> tkdiff, diff, windiff, spiff, meld, FileMerge ...

Test environment generator / test harness generator

Generates test driver and stubs, i.e. test harness, (from understanding the code)

Even generic test frameworks (xUnit)

Standardizes test environment (for component- and integration test)

Typical developer tool

Examples:
www.ipl.com
www.ldra.com
www.junit.org

Test Execution Tools

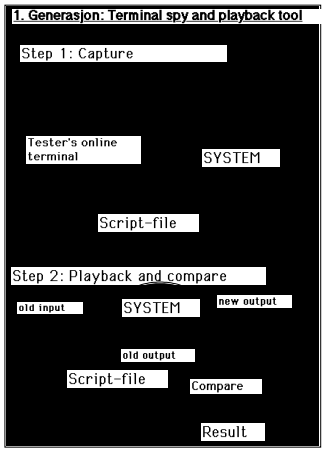
Also called test driver, test robot

Very popular
Automates test execution
    replaces staff which is expensive and unreliable
Automatic test log
Can record tests
Problem to maintain automatic tests

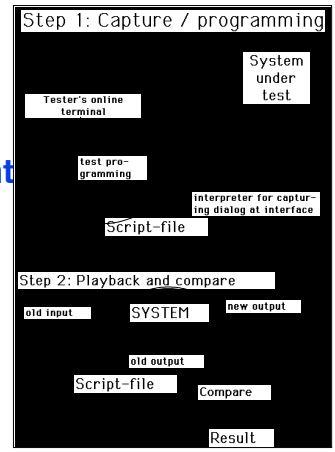
Requires programming (abilities)

Test Execution Automation Tools

Capture-playback
Does not work well
Scripts not maintainable
Can be used for logging



Test Execution Automation (modern)



Test is programmed
Requires development environment for the test scripts
New programming language!
Keep data and test script separate!
Flexibility!
Data driven test
Keyword-driven test
Possible with ”intelligent” capture + editing

Coverage Measurement Tools

Instrument the code with “counters”
Count during test execution
Show execution statistics after execution

Problems:
- The program may be much slower (“intrusive tool”).
- Some code can be difficult to execute.
- Too much reliance on thinking that full coverage means correct code.

Example for Coverage Report

http://www.semdesigns.com/Products/TestCoverage/JavaTestCoverageReport.

```
SUMMARY:
      Total Probes: 136832
      Total Files: 3824
      3.0% covered (4195 out of 136832).
      97.0% uncovered (132637 out of 136832).

COVERAGE REPORT BY FILE:
      [1] D:\SDProductTests\DMS\Domains\Java\TestSource\bandera\Bandera.java
    96.0% uncovered (48 out of 50).
      [2]
    D:\SDProductTests\DMS\Domains\Java\TestSource\bandera\ca\mcgill\sable\laleh\java
    \astfix\ASTFixer.java    100.0% uncovered (651 out of 651).
      [3]
    D:\SDProductTests\DMS\Domains\Java\TestSource\bandera\ca\mcgill\sable\laleh\java
    \astfix\JJCParser.java    100.0% uncovered (2 out of 2).
```


Example for Coverage Report

http://www.testwell.fi/ctcprofi.html

```
MONITORED SOURCE FILE : calc.c
INSTRUMENTATION MODE  : function-decision-multicondition-timing

    START/       END/
      TRUE      FALSE    LINE DESCRIPTION
================================================================================
         7          0       4 FUNCTION is_prime()
         2          5       8 if (val == 1 || val == 2 || val == 3)
         0            -     8   T || _ || _
         2                  8   F || T || _
         0            -     8   F || F || T
                    5       8   F || F || F
         2                  9   return 1
         2          3      10 if (val % 2 == 0)
         2                 11   return 0
         4          1      12 for (;divisor < val / 2;)
         2          2      14   if (val % divisor == 0)
         2                 15     return 0
         1                 17 return 1

***TER  88 % ( 15/ 17) of FUNCTION is_prime()
```

Security Testing Tools

Virus check
Attack prevention
Firewall (not a test tool in narrow sense)
Looking for vulnerabilities in the system
Checking security issues in general

Note: Analysis of test coverage makes it more difficult to implement unwanted back doors!
Note: Security tools can be part of many tools categories!

5 Tools for Performance and Monitoring

Especially for performance test, load test, stress test
- Load generators
- Load test drivers
- Performance measurement

Check of portability
Monitors (specific system resources, problem detection, logging)
Dynamic analysis (during test execution) (developer tool)
- "Spies"
- Time monitors
- Memory leak analysis (Example: MemoryScape from Totalview (www.totalviewtech.com/download/)

Example: www.sourceforge.net/projects/grinder, a Java-based open source-Tools for load test

6 Tools for Specific Testing Needs

Data quality assessment
... And more like...
Test of web applications
Simulation of magnetic cards
Simulation of network traffic
Test tools for mobile applications
Emulators for hardware / firmware
Static analysis for special platforms
Tools for testing usability
Tools for test of internationalization

The Tool Effect

The tool may run on the same platform as the test object.
The tool then uses resources (memory, time).
The tool then has an influence on the test object.
This is called “probe effect” (“Heisenberg effect”).

Tools running on the same platform are called “intrusive”.
Test results can then be wrong! -> new test without tools or ...

In some categories there exist ”non-intrusive” tools running on their own hardware.

Questions?

2 Selection, Introduction And Use Of Test Tools

1 Cost, risk and potential benefits
2 Special considerations for some tools
3 Selection, introduction and follow-up of tools

2.1 Cost and Potential Benefits of Tools

Potential benefits:

- Possible to run new kinds of tests
- Less repetitive and manual work (f.ex. for platform- or regression test)
- Less boring work (repetition, static analysis)
- More reliable, repeatable and complete test (generated/run by tools)
- Structured
- Cheaper (automated), more productive
- Objective (measurements)
- Ease of acces to info about the tests and the testing

Cost of Tools

- Evaluation and selection of tools
- License
- Training (!!!)
- Use, configuring, tailoring, programming
- Analyzing results from the tools
- Maintenance

Test Tools and the Test Process

Test tools should be used to support the test process.

Test tools without a process work badly.

Test tools do not automate the (disorganized) test process.

> "If you automate chaos, you get faster chaos."
>
> (Dorothy Graham)

Risk with Tools

Unrealistic expectations

Underestimating time, cost and effort to introduce (training, coaching, help, ...)

Underestimating time and cost for follow-up, change of processes, etc.

Underestimating need for maintenance of tools and the assets maintained or generated by tools

Over-reliance on the tool

Defects in the tool, and fighting against them

Neglecting version control of test assets

Neglecting relationships and interoperability between tools

Tool vendor problems (out of business, retiring the tool)

Sometimes too much trust into the tool instead of intelligent choices.

Poor support

Platform problems

2.2 Special Considerations For Some Tools

- Test execution tools
  - capture-playback not good
  - data-driven test
  - keyword-driven test
  - programming skills required
- Static analysis tools
  - configure to reduce “false-positives”
  - may enforce coding standards
- Test management tools
  - check interface to other tools and spreadsheets!

2.3 Selection and Introduction of Tools

Main Considerations in Selecting a Tool

Summary: To be checked before introducing a tool

Consider organization maturity, strengths and weaknesses.
Identify possibilities to use tools and to improve the test process.
Evaluate against clear requirements and objective criteria.
Pilot (proof-of-concept) to test if the tool really is good enough.
Evaluate the supplier (economy, training, support).
Identify requirements to internal support and tailoring before using the tool.
Evaluate the training needs
Make a business case and follow up cost and benefits.

Advice for Tool Selection (not for exam)

Define your goals for the test tool.
Document the requirements a tool shall meet.
Evaluate which types of tools are best for your needs.
Get evaluation copies and check them out.
Collect references from customers using the same tool, and check with them if they think the tool helps them.
Make a ”short list” with 1-3 tools – let the supplier run a ”Proof-of-concepts” on your installation, and run a pilot.

Check list: General Criteria for Evaluation (not for exam)

Product
Selling points, developer community, human hierarchies, licensing

Integration
Collaboration, modularity, standards

Supplier
Quality, economy, track record, where, support ability, ...

Use
Aupport, ease of deployment

Acceptance
User community, market penetration

Product application
Usability, Interfacing, performance, reliability, security, proven technology or new, vendor independence, platform independence, support, reporting, functionality, ...

Norbert Jansen,
Open Source Software Test Tools
EuroSTAR 2005

Introduction of Tools

Introduction process:

1. **Run a pilot**
   to learn more details about the tool and how it works in the current environment and with the current people
1. **Evaluate experiences from the pilot**
2. **Tailor the work process and design rules and standards for using the tool**
3. **Train the users**
4. **Introduction - rollout - step wise**
5. **Coaching and follow-up**

Introduction of Tools - Pilot

May be run for several tools on the short list!

Goals with a pilot project

- Proof-of-concept
- Learn the details of the tool.
- See how the tool works together with existing processes and practice, find necessary changes.
- Determine and decide upon standards (f.ex. naming conventions, layout, ...).
- Get data about cost / benefit.
- Pilot = TYPICAL project!

Do we get the benefits with acceptable cost?

Introduction of Tools (cont'd.)

Success factors (most of this is repeated)

- Step wise introduction
- Processes must be adapted and improved in order to fit the tool.
- Training and coaching of the users
- Rules for use: Support, rules, standards, templates, utilities
- Implement a way to learn from others' experiences: Follow up, measurements, tricks, FAQs, Wiki, blogg (Lessons learned sessions)
- Monitor use, cost and benefits
- Provide support for the test team
- Gather lessons learned from all teams using the tool

3 Summary

Goal: Automating shall save resources and/or increase quality levels.

Tools exist for all process phases.

Tools benefits can only be achieved with a systematic and well organized test process.

Continuous follow-up required.

Not everything can be automated!

Questions?

References and literature

- Fewster, M., Graham, D.: Software Test Automation, Effective use of test execution tools, Addison-Wesley, 1999.
- www.softwareqatest.com has lots of good tool links.
- A list of tools (software engineering and testing) is in Better Software Magazine 1/2008: www.bettersoftware.com.
- A list of open source testing tools is on www.opensourcetesting.org, www.sourceforge.net.
- www.testingfaqs.org/tools.htm has a tool supplier list.
- The site www.aptest.com has many interesting software testing links, especially to tools.
- IPL, www.ipl.com, makes tools for unit and integration test of Ada, C, C++ programs. Some good papers about unit testing are on their web site.
- Imbus AG (Germany) has a tools list on www.imbus.de
- http://www.imbus.de/download/papers/dl_whitepapers.html "How to Automate Testing of Graphical User Interfaces, Results of the ESSI PIE 24306", imbus AG, 1999.
- Autotestco, www.autotestco.com , has good links to test tools suppliers, but also to seminar companies in the USA, as well as a book about test automation.
- www.ovum.com continually evaluates the most used testing tools. You can order their (expensive) report "Ovum Evaluates: Software Testing Tools", as well as continuous updates.
- Moseley, D.J., Posey, B. A., Just Enough Software Test Automation, Prentice Hall, Yourdon Press 2002.
- Dustin, E., Rashka, J., Paul, J.: Automated Software Testing, Introduction, Management and Performance. Addison-Wesley, 1999.
- Buwalda, H., Jansson, D., Pinkster, I.: Integrated Test Design and Automation, Using the TestFrame Methods, Addison-Wesley, 2002.