1 Overview	2 Life cycle components	3 Infrastructure components	4 Management components	5 Standards and Organizing
6 Static tesing	7 Dynamic testing	8 Test management	9 Tools	

Tool support for testing (CAST)

Learning objectives

- Classify different types of test tools according to the test process activities
- Recognize tools that may help developers in their testing
- Summarize the potential benefits and risks of test automation and tool support for testing
- State the main principles of introducing a tool into an organization

References

- Dorothy Grahamet, Erik van Veenendaal, Isabel Evans, Rex Black. *Foundations of software testing: ISTQB Certification*

Contents

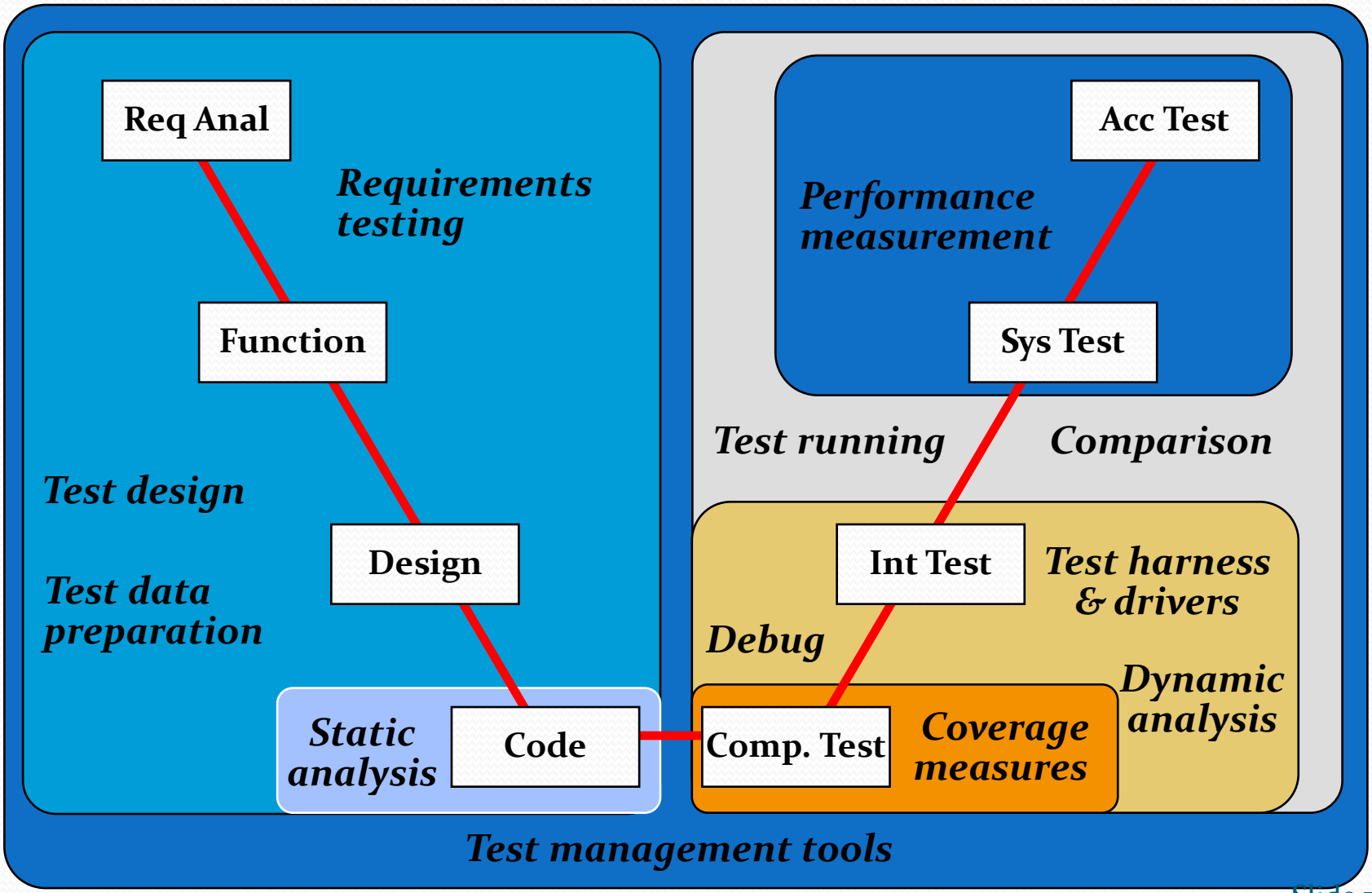
Testing tool classification

- Tool support for **management** of testing and tests
  - requirements management tools
  - incident management tools
  - configuration management tools
  - test management tools
- Tool support for **static** testing
  - review process support tools
  - static analysis tools
  - modeling tools
- Tool support for test **specification**
  - test design tools
  - test data preparation tools

Testing tool classification (con't)

- Tool support for test **execution** and **logging**
  - Test execution tools
  - Test harness/unit test framework tools
  - Test comparators
  - Coverage measurement tools
  - Security tools
- Tool support for **performance** and **monitoring**
  - Dynamic analysis tools
  - Monitoring tools
- Tool support for **specific** application areas
- Tool support using other tools

Where tools fit?



Tool support for management
Requirements management tools

- Help with analysing requirements by **verification**
- Support for
  - **storing** requirement statements;
  - **identifying** undefined, missing or 'to be defined later' requirements;
  - **checking consistency** of requirements;
  - **prioritizing** requirements for testing purposes;
  - **traceability** of requirements to tests and tests to requirements, functions or features;
  - calculating **requirements coverage** by a set of tests

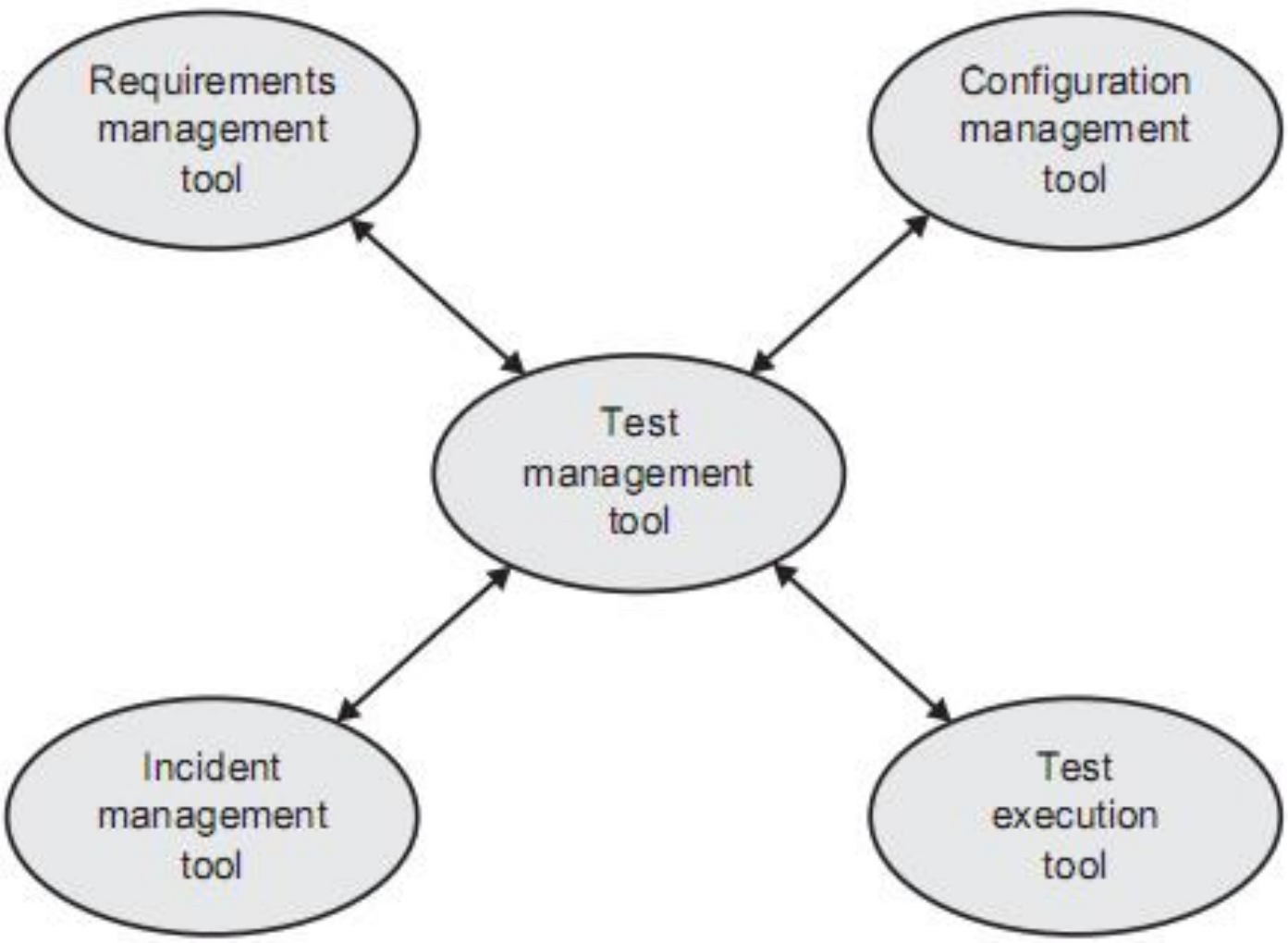
Tool support for management
Incident management tools

- Known as ‘defect-tracking tool’, ‘defect-management tool’
- Support for
  - **storing** information and attachments;
  - **prioritizing** incidents;
  - **assigning** actions to people;
  - status (e.g. open, rejected, duplicate, deferred, ready for confirmation test, closed);
  - **reporting** of statistics/metrics (e.g. number of incidents with each status, total number raised, open or closed)

Tool support for management
Configuration management tools

- Managing: the versions of different software (and hardware) components, testware
- Support for
  - **storing** information about versions and builds of the software and testware
  - **traceability** between software and testware and different versions
  - **keeping track** of which versions belong with which configurations

Tool support for management
Test management tools



Tool support for management
Test management tools

- Help to gather, organize and communicate information about the testing on a project
- Support for
  - **management** of tests, testing activities
  - **scheduling** of tests to be executed
  - **interfaces** to other tools (e.g. requirements, incident, configuration, test execution)
  - **traceability** of tests (to requirements, designs)
  - **logging** test results

Tool support for static testing
Review process support tools

- Keep track of all the information for a review process
- Support for
  - **storing** and **sorting** review comments
  - **communicating** comments to relevant people
  - **keeping track** of comments and providing statistical information
  - providing **traceability** between comments, documents reviewed and related documents
  - a **repository for rules**, **procedures** and **checklists** to be used in reviews, as well as **entry** and **exit criteria**
  - **monitoring** the review status
  - **collecting** metrics and **reporting** on key factors

Tool support for static testing
Static analysis tools (D)

- Used by developers
- Analyse source code (not executing)
- To produce code that will be easier to maintain in the future
- Support for
  - calculate **metrics** (cyclomatic complexity, nesting levels)
  - enforce **coding standards**
  - analyze **structures** and **dependencies**
  - identify **anomalies** in the code

Tool support for static testing
Modelling tools (D)

- Help to validate models of the system or software
- Used by developers
- Generate test cases
- Support for:
  - identifying inconsistencies and defects within the model
  - prioritize areas of the model for testing
  - helping to identify test conditions using a modeling language such as UML

Tool support for test specification
Test design tools

- Help to derive test inputs, or test cases
- Support for
  - **generating test input values** from
    - requirements
    - design models (state, data or object)
    - code
    - graphical user interfaces
    - test conditions
  - **generating expected results**, if an oracle is available to the tool

Tool support for test specification
Test data preparation tools

- Test data preparation tools enable data to be **selected** from an existing database or **created**, **generated**, **manipulated** and **edited** for use in tests
- Support to
  - **extract** selected data records from files or databases
  - **generate** new records populated with pseudo-random data, or data set up according to some instructions or rules
  - **construct** a large number of similar records from a template

Tool support for execution and logging
Test execution tools

- Enable tests to be executed automatically
- Known as 'capture/playback' tools, 'capture/replay' tools
- Use a scripting language
  - linear scripts
  - structured scripts
  - shared scripts
  - data-driven scripts
  - keyword-driven scripts
- Best used for regression testing

Tool support for execution and logging Test execution tools (cont'd)

- Support for
  - **capturing** test inputs while tests are executed manually
  - **storing** an expected result
  - **executing** tests from stored scripts and data files
  - **dynamic comparison** (screens, elements, links, controls...)
  - **logging** results of tests run
  - **filtering** of subsets of actual and expected results
  - **measuring** timings for tests
  - **sending** summary results to a test management tool

Tool support for execution and logging
Test harness/unit test framework tools (D)

- Used to exercise software which does not have a user interface (yet)
- Simulators (where testing in real environment would be too costly)
- Similar to test execution tools
  - no capture/playback facility

Tool support for execution and logging Test comparators

- Detect differences between actual test results and expected results (characters, screens, bitmaps)
- Two ways
  - dynamic comparison
    - included in test execution tools
    - useful when comparing in the middle of a test (e.g error message...)
  - post-execution comparison
    - 'stand-alone' tool
    - best for comparing a large volume of data (e.g. in files or databases)

Tool support for execution and logging Coverage measurement tools (D)

- Usually used at component testing level (statement, branch, condition,...) or component integration level (a call to a function)
- Support for
  - **identifying** coverage items;
  - **calculating** the percentage of coverage items;
  - **reporting** coverage items that have not been exercised;
  - **identifying** test inputs to exercise as yet uncovered items;
  - **generating** stubs and drivers

Tool support for execution and logging Security tools

- Used to test security by trying to break into a system
  - on the network, the support software, the application code or the underlying database
- Support for
  - identifying viruses
  - detecting intrusions (e.g. Denial of Service Attacks)
  - simulating various types of external attacks
  - identifying weaknesses in password files and passwords
  - security checks during operation

Tool support for performance and monitoring

Dynamic analysis tools (D)

- Provide run-time information on software (while tests are run)
  - allocation, use and de-allocation of resources, e.g. detecting memory leaks
  - identifying pointer arithmetic errors
  - identifying time dependencies

Tool support for performance and monitoring

Monitoring tools

- Keep track of the status of the system in use
  - servers, networks, databases, security, performance, website and internet usage, and applications
- Support for
  - identifying problems and sending an alert message to the administrator
  - logging real-time and historical information
  - finding optimal settings
  - monitoring the number of users on a network
  - monitoring network traffic

Tool support for performance and monitoring

Performance testing tools

- Concerned with testing at system level to see whether or not the system will stand up to a high volume of usage
- 'Load' test checks that the system can cope with its expected number of transactions
- 'Volume' test checks that the system can cope with a large amount of data
- 'Stress' test is one that goes beyond the normal expected usage of the system (to see what would happen outside its design expectations), with respect to load or volume

Tool support for performance and monitoring

Performance testing tools (cont'd)

- Simulate loads including multiple users, heavy network traffic and database accesses
- Enable to find defects:
  - general performance problems
  - performance bottlenecks
  - record-locking problems
  - concurrency problems
  - excess usage of system resources

Other tools

- Tool support for specific application areas
  - web-based performance testing tools
  - static analysis tools for specific development platforms and programming languages
  - dynamic analysis tools for embedded systems
- Tool support using other tools
  - word processors, spreadsheets , databases for storing test designs, test scripts or test data
  - debugging tools
  - any type of tool available to support any of the testing activities

Contents

Potential benefits

- Reduction of repetitive work, time and effort
  - e.g. running regression tests; entering the same test data over and over again; checking against coding standards; creating a specific test database...
- Greater consistency and repeatability
  - e.g. checking to confirm the correctness of a fix to a defect; entering test inputs; generating tests from requirements
- Objective assessment
  - e.g. assessing the cyclomatic complexity, coverage, incident statistics
- Ease of access to information about tests
  - e.g. a chart or graph

Potential risks

- Unrealistic expectations for the tool
- Underestimating the time, cost and effort for the initial introduction of a tool
- The benefits being obtained will decrease and the tool will become redundant
- Over-reliance on the tool

Contents

Introducing a tool into an organization

- Assessment of the organization's **maturity**
- Identification of the **areas** within the organization where tool support will help to improve testing processes
- Evaluation of tools against clear requirements and objective criteria
- Proof-of-concept to see whether the product works as desired and meets the requirements and objectives defined for it
- Evaluation of the vendor
- Identifying and planning internal implementation

Pilot project

- The project with the first thing done with a new tool
- Objectives
  - to learn more about the tool
  - to see how the tool would fit with existing **processes** or **documentation**, how those would need to change to work well with the tool and how to use the tool to streamline existing processes;
  - to decide on standard ways of using the tool

Success factors

- Providing adequate training, coaching and mentoring of new users;
- Defining and communicating guidelines for the use of the tool;
- Store information of problems encountered and the solution;
- Implementing a continuous improvement mechanism;
- Improving the process to fit with the new tool, or amending the use of the tool to fit with existing processes;
- Monitoring the use of the tool and the benefits achieved

Contents

Types of CAST tool

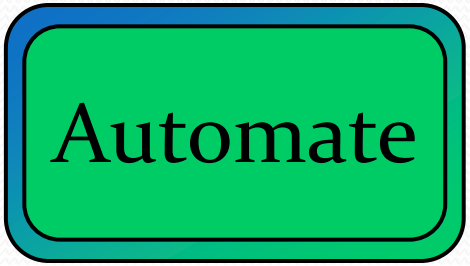
Potential benefits and risks of CAST tool

Introducing a tool into an organization

Best practice

Tests to automate

- Run many times
  - regression tests
- Expensive to perform manually
  - time consuming
  - multi-user tests, endurance/reliability tests
  - not enough resources
- Difficult to perform manually
  - performance test



Tests not to automate

- Not run often
  - if no need (rather than expensive to run manually)
- Not important
  - will not find serious problems
- Usability tests
  - do the colours look nice?
- Some aspects of multi-media applications

Sample exam questions

1. Which tools help to support static testing?

a. Static analysis tools and test execution tools.

b. Review process support tools, static analysis tools and coverage measurement tools.

c. Dynamic analysis tools and modeling tools.

d. Review process support tools, static analysis tools and modeling tools.

Sample exam questions

2. A test management tool is most likely to integrate with which of the following tools?

a. Performance testing tool

b. Test data preparation tool

c. Static analysis tool

d. Requirements management tool

Sample exam questions

3. Requirements management tool is NOT used for

a. Storing requirements

b. Identifying undefined, missing or to be defined requirements

c. Traceability of requirements

d. Generating test data

Sample exam questions

4. A Modeling tool is NOT used for

a. Identify inconsistencies or defects in models

b. Identifying undefined , missing or to be defined requirements

c. Help in prioritization of tests in accordance with the model in review

d. Predicting system response under various levels of loads

Sample exam questions

5. Which test activities are supported by test harness or unit test framework tools?

a. Test management and control.

b. Test specification and design.

c. Test execution and logging.

d. Performance and monitoring.

Sample exam questions

6. Which of the following are advanced scripting techniques for test execution tools?

a. Data-driven and keyword-driven

b. Data-driven and capture-driven

c. Capture-driven and keyhole-driven

d. Playback-driven and keyword-driven

Sample exam questions

7. A security tool is **not** used for

a. Identifying viruses

b. Identify denial of service attacks

c. Identifying weakness in passwords for files and passwords

d. Identifying performance bottlenecks

Sample exam questions

8. What are the potential benefits from using tools in general to support testing?

a. Greater quality of code, reduction in the number of testers needed, better objectives for testing.

b. Greater repeatability of tests, reduction in repetitive work, objective assessment.

c. Greater responsiveness of users, reduction of tests run, objectives not necessary.

d. Greater quality of code, reduction in paperwork, fewer objections to the tests.

Sample exam questions

9. What is a potential risk in using tools to support testing?

a. Unrealistic expectations, expecting the tool to do too much.

b. Insufficient reliance on the tool, i.e. still doing manual testing when a test execution tool has been purchased.

c. The tool may find defects that aren't there.

d. The tool will repeat exactly the same thing it did the previous time

Sample exam questions

10. Which of the following types of test tool are most likely to include **traceability** functions?

(i) Performance testing tool

(ii) Requirements management tool

(iii) Configuration management tool

(iv) Static analysis tool

a. (i) and (ii)

b. (i) and (iv)

c. (ii) and (iii)

d. (iii) and (iv)