

# 5. Test Organization and Management

Get an understanding of a test manager's most important tasks

Hans Schaefer  
[hans.schaefer@ieee.org](mailto:hans.schaefer@ieee.org)  
<http://www.softwaretesting.no>

## Contents

- 1 Test Organization**
  - 1.1 Organization and Independence
  - 1.2. Tasks for Test Personnel
- 2 Test Planning and Estimating**
  - 2.1 Foundations
  - 2.2 Planning Activities
  - 2.3 Exit Criteria
  - 2.4 Estimation
  - 2.5 Test Approaches
- 3 Follow-up and Progress Management**
  - 3.1 Progress and Follow-up
  - 3.2 Test Reporting
  - 3.3 Test Control
- 4 Configuration Management**
- 5 Risks and Testing**
  - 5.1 Project Risks
  - 5.2 Product Risks
- 6 Problem / Incident Handling**

# 1 Test Organization

## ->1 Test Organization

1.1 Organization and Independence
1.2 Tasks for Test Personnel
2 Test Planning and Estimating
2.1 Foundations
2.2 Planning Activities
2.3 Exit Criteria
2.4 Estimation
2.5 Test Approaches
3 Follow-up and Progress Management
3.1 Progress and Follow-up
3.2 Test Reporting
3.3 Test Control
4 Configuration Management
5 Risks and Testing
5.1 Project Risks
5.2 Product Risks
6 Problem / Incident Handling

## 1.1 Independence

## 1.2 Tasks for different test personnel

## 1.1 Independence (repeated from earlier)

Testing requires an independent view

Levels of independence:

1. Developer self
2. Colleague of developer (buddy system)
3. Special tester(s) in the development team
4. Independent test group in the organization (report to project management or line management)
5. Independent testers from user organization or business
6. Independent test specialists (for special test areas or certification)
7. Outsourced or independent organization outside the own organization (third party test service)

Three dimensions:

Technical (fresh view)  
Management  
Financial

**Independent preparation most important!**

## Why Independence? Different Focus:

### Developer

- Wants to get ready
- Thinks everything is right
- Blind for own errors
- Constructive
- Uses (possibly wrong) assumptions

### Tester

- Has time to test
- Is a pessimist
- Specialist in finding faults
- Destructive
- Can check assumptions

It takes extra time for another one to learn the test object.  
**-> Varying degree of independence!**

## What Requires Increasing Independence?

- Safety-critical projects
- Large projects
- Higher test levels

### Problems with independent test

- Time to learn
- Test can be seen as a bottleneck
- Developers can lose responsibility ("Just ship it!")

## Exercise: Define a typical distribution of responsibilities

	Web service (train information)	Control of ABS brake for cars
Component test		
Integration test		
System test		
System integration test		
Acceptance test		

## 1.2 Tasks for Different Test Personnel

Many roles

The two most important:

- Test manager
- Tester

# Test Manager

## Test manager, test coordinator = Test project manager

- Develops the test policy for the organization
- Develops the test strategy and test plan
- Coordinates test plans and approach with project manager and others
- Acquires test resources
- Runs the test project through its phases, follows up and controls it
- Implements support processes like problem / Incident handling and configuration management
- Controls the project depending on results and progress
- Selects and implements tools (testers may check them out first)
- Decides about automation
- Organizes training
- Organizes test environment
- Plans time schedule
- Plans and uses measurements for test progress and evaluation of quality of the product and the testing
- Reports from testing
- Initiates archiving and learning processes

# Tester

- Analyzes and checks specifications etc.
- Gives feedback about testability
- Develops or acquires test cases and test data
- Implements the (details of) tests
- Helps others with review of test material
- Sets up the test environment
- Runs the tests and evaluates the results
- Reports problems and deviations from expected results
- Uses tools, automates test work
- Measures performance and other attributes of test objects
- Collects and reports measurement data

## Roles where specialists are often used:

- Automating test
- Data base set up
- Support for test tools
- Get test data from production
- Performance test

# “Soft Skills” for a Tester

(not for exam)

## General human skills for testing people

[http://www.stickyminds.com/r.asp?F=DART\\_6752](http://www.stickyminds.com/r.asp?F=DART_6752)

- Discipline and Perseverance
- Reading Skills
- Critical thinking
- Communication and interpersonal skills
- Time management and effort prioritization
- The right attitude
  
- Meeting moderator skills (source: Martine Herpers, Knowledge Department)

# 2 Test Planning and Estimating

1 Test Organization
1.1 Organization and Independence
1.2 Tasks for Test Personnel
-> 2 Test Planning and Estimating
2.1 Foundation
2.2 Planning Activities
2.3 Exit Criteria
2.4 Estimation
2.5 Test Approaches
3 Follow-up and Progress Management
3.1 Progress and Follow-up
3.2 Test Reporting
3.3 Test Control
4 Configuration Management
5 Risks and Testing
5.1 Project Risks
5.2 Product Risks
6 Problem / Incident Handling

## Content:

- Start and Exit Criteria
- Explorative testing
- Test approach
- Test level
- Test plan
- Test procedure / Test instruction

## 2.1 Foundations

**Test is a lot of, and complicated work. It requires a plan.**

**A template... in IEEE Standard 829 (Software Test Documentation).**

## What the Plan Depends on

### Information about

- **Criticality**
- **Other quality assurance (quality plan)**
- **Test objectives**
- **How much test (level, special test, all test levels and types?)**
- **Goal of project and product**
- **Risk of project and product**
- **Time for test planning and test running with respect to the project**
- **Testability**
- **Resources (personnel, test tools and equipment)**

**The plan should be a living document! Must be updated and tailored all the time.**

## 2.2 Planning Activities

The general approach (test levels, start criteria, comprehensiveness)  
Coordination of test with development  
What shall be tested, prioritization, when, how  
What shall not be tested and why  
Exit Criteria  
Resources  
Assign roles  
Schedule  
Plan the test documentation (templates etc.)  
Choose and prepare collection of measurement data and their evaluation  
Determine how results will be evaluated

## Test Plan after IEEE 829-1998

1. Test Plan Identifier
2. Introduction
3. Test items
4. Features to be tested ([product risks](#))
5. Features not to be tested
6. Approach
7. Item pass/fail criteria (test exit criteria)
8. Suspension criteria and resumption requirements
9. Test deliverables
10. Testing tasks
11. Environmental needs
12. Responsibilities / Authority
13. Staffing and training needs
14. Schedule
15. Risks and contingencies ([project risks](#))
16. Approvals

**Too bureaucratic?  
-> Make your own template.**

**The standard has been updated  
and (829-2008) contains now both a  
Master and a Level Test Plan.**



## Test Execution Schedule

### Dependencies / Order:

- When is the software ready to be tested? Preliminary versions?
- 1. Set up/verify the test environment(s)<sup>(1)</sup>
- 2. **Installation**
- 3. Create basis data (settings, database content)
- 4. **Smoke test**
- 5. Remaining test
  1. Create, then search, update, results from several actions
  2. Delete and cleanup at end
  3. Uninstallation, upgrading etc.
  4. **Technical dependencies** (test cases that need correct results from other test cases)
- 6. **Re-test and regression test**, typically at least twice

## Test Schedule Exercise

Your task is to draft a schedule for system testing the web-based train information system.

Make a list of major tasks and time dependencies between them.

The list should contain groups of typical test cases (like easy positive tests, user error tests, difficult information search, ...)

## Start Criteria

### To execute the testing

- Static analysis and review are done, and faults are explained or fixed.
- Clean compile, link, installation.
- Test environment, test data and test tools are in place and tested.
- Product passes a smoke test.
- Product shall be “stable enough” in the test before. Can be expressed statistically with MTBF.
- Not too many failures in the tests before. Max. twice the number expected in this test level.
- The test level before is finished with most or all exit criteria met.

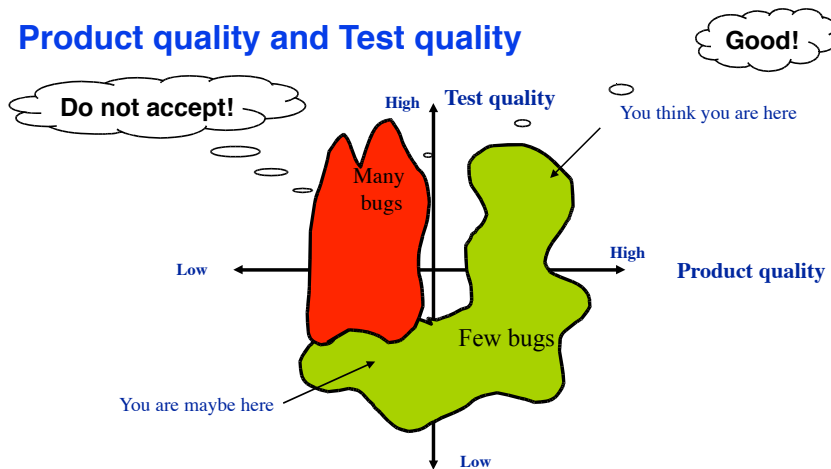
## 2.3 Exit Criteria

### When to stop testing?

- For a test level
- For a set of test cases with a certain goal

## What Do We Want to Control?

### Product quality and Test quality



## Reasonable Exit Criteria

The next pages contain a lot of details on exit criteria  
(Not part of ISTQB curriculum)

### Test quality related

- (1) Certain test methods have been used
- (2) Test coverage
- (3) A typical number of problems has been found
- (4) Percentage of unplanned defect detection (through exploratory testing)
- (5) Test progress versus plan

### Product quality related

- (1) The product is stable at the end of testing
- (2) Reliability model is used
- (3) State of defect repair

You always need a combination!

## (1) Require the use of test design methods

For example:

Test cases have been designed using state transition testing at switch level.

Test cases have been designed using boundary value analysis.



No required test design method



Every data element correct and false



Basic test design methods



Combination test methods

## (2) Test Coverage

For example:

Module test: 100% statement and 85% branches have been executed.

Integration test: All call pairs and all other interfaces have been executed.

System test: 70 - 85% branches have been executed.

All requirements have been tested.

Typical percentage:  $\leq 85\%$  (6 of 7).  
Above this little benefit and a lot  
Of effort.



No required test coverage



Every function tested



Basic black and white box coverage



Higher level coverage criteria

### (3) A typical number of problems has been found

For example: Test until you have found X problems.

What is a reasonable number?

You need data to estimate this!

Count your own defect rate!



No idea about typical numbers



Problem data collected and reviewed



Number based on experience, about same number found



Number based on experience, less defects found

### (4) Percentage of unplanned error detection

Measure how many percent of the defects are detected by improvised extra tests.

If this percentage is high, the test plan was bad.

The percentage must not be over the one from earlier successful projects.

(Industry data indicate max. 25 percent.)



More than 50%



25-50%







25%



<25%

## (5) Test Progress vs. Plan

**Most test cases should have run without failures.  
Especially high priority test cases (for high risk areas).**

-  Some high priority cases not run or unsuccessful
-  All high priority test cases success, lower levels not run or failed
-  All test cases run, some failures at lower priority test cases
-  All test cases run and successful





## (6) Stable Product: Failure Rate / Test Productivity

**For example:**

**Less than 1 serious failures during last minute / hour / day / week of testing**

**This criterion implies that the product is stable.**

**Finding the next fault by testing is more expensive than letting it live on. Its marginal cost exceeds its damage.**

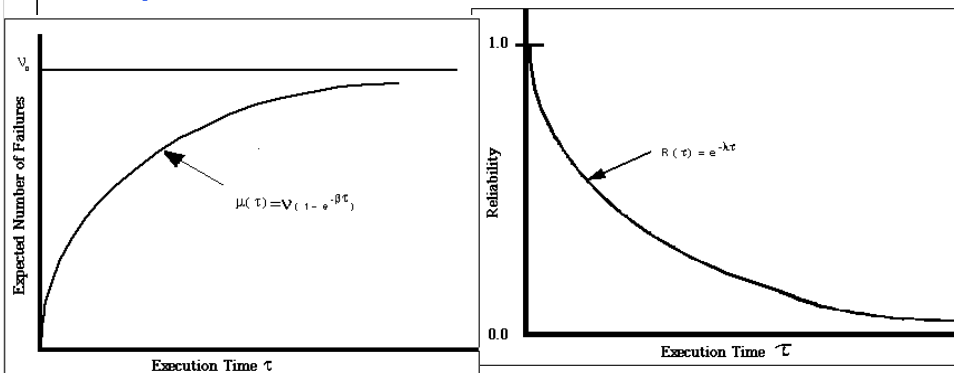
-  Failure rate > Required in use \*10
-  As required \* (2 to 10)
-  As required to twice
-  Less than required

## (7) Use of a reliability model

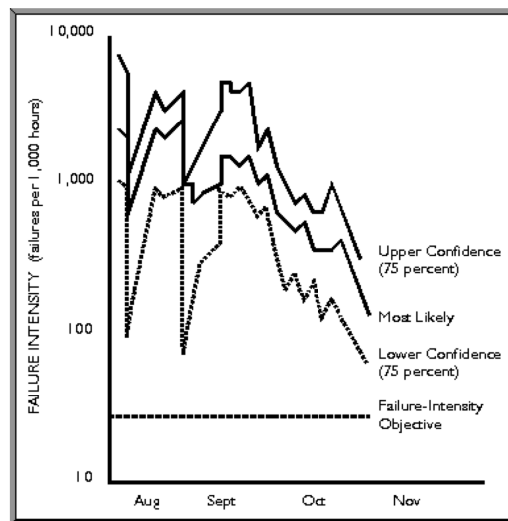
- Failure intensity falls exponentially.
- Predict the remaining faults.
- Predict when the product will be reliable enough and with which confidence interval.
- Need for calibration /experience data).
- Need for statistical usage test.

## Example: Ideal failure detection curves

Faults are counted only once, or immediately repaired.



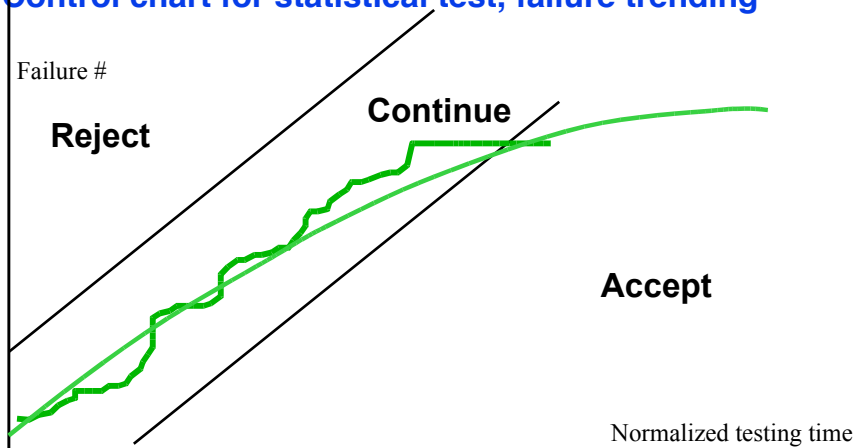
## Example for REAL failure data



Note:  
Exponential scale!

## Reliability Model for Acceptance Test

Control chart for statistical test, failure trending





## References to Reliability Modeling

Internet search for:

John Musa

Okumoto

Bev Littlewood

“Software Reliability Engineering”





## (8) The state of repair

Are the found defects repaired?

Re-test done?

Which known defects are not repaired, why, and when will they be fixed?

Not too many defects should be open.

	Catastrophic defects not fixed
	Serious defects not fixed
	Many minor defects not fixed
	Few minor defects open

## (9) Cost and Time

Maximal cost for testing

Maximal risk in the application

Release time (can be determined before)

## Criteria in Practice

Always more than one criterion.

Most of them should be "in the right direction".

Time to market must be included.

For example:

Low and falling failure frequency

High test coverage

About as many defects as expected

Time



Most criteria in red



Average at this level



Most criteria in green area



All criteria in green area

## Example of criteria in practice

Bjarne Månsson, lecture to Danish Special Interest Group in SW Testing, 2001:

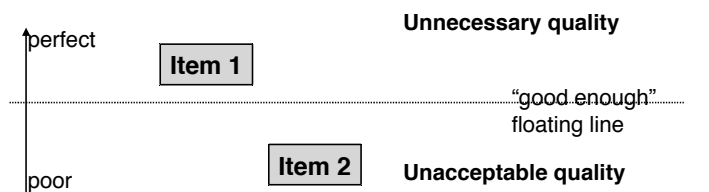
### Completeness:

- All requirements within scope implemented
- Manuals and on-line help available
- Demo files, sample data and education material available
- Production material (parts lists, installation files ) available

### Quality

- System tests completed
- Performance tests completed "satisfactory"
- Report on closed and deferred defects
- Training on sales staff, consultants and support performed
- No defects of severity 1
- Max 10 defects of severity 2
- Defect trend falling (per test hour)

## "Good Enough" Testing



The following four points must apply:

1. System has enough benefits
2. System has no critical problems
3. Benefits of system outweigh the problems
4. In the present situation, *all things considered*, improving the system would cause more harm than good.

James Bach, "Good enough Quality: Beyond the Buzzword". IEEE Computer, August 1997.

James Bach, "A framework for good enough testing", IEEE Computer Magazine, October 1998

## The Final Exit Criterion

You finish testing when you know enough about the product that management can make a decision about releasing the product with necessary confidence.

People need to know the risk of releasing.

## References About Exit Criteria

- Li Guo Huang and Barry Boehm, How Much Software Quality Investment Is Enough: A Value-Based Approach, IEEE Software, Sep/Oct 2006
- B. Steece, S. Chulani, and B. Boehm, "Determining Software Quality Using COQUALMO," Case Studies in Reliability and Maintenance, W. Blischke and D. Murthy, eds., Wiley, 2002; documentation available at [http://sunset.usc.edu/research/coqualmo/coqualmo\\_main.html](http://sunset.usc.edu/research/coqualmo/coqualmo_main.html)
- B. Boehm et al., "The ROI of Software Dependability: The iDAVE Model," IEEE Software, vol. 21, no. 3, 2004, pp. 54–61; initial spreadsheet tool available at <http://sunset.usc.edu/cse/pub/research/iDAVE/iDAVE.zip>.
- J. Bullock, "Calculating the Value of Testing," Software Testing and Quality Eng., vol. 2, no. 3, 2000, pp. 56–62.

## 2.4 Estimation

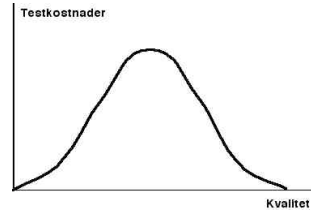
Test work must be estimated.

Two main methods:

- Based on analogy (experience from not too different projects or from experts)
- Based on algorithms

Syllabus: Metrics-based and expert-based

Problem: The more failures during test, the more testing must be done.



## Estimation can be based on

### Product attributes

- Criticality
- Size and complexity
- Requirements for documentation
- Regulation (requirements by authorities)
- Safety and security requirements

### Process attributes

- Stability of the organization
- Tools
- Test process
- Time pressure
- People qualifications

### Product quality

- Quality of test basis (specification)
- Number and weight of faults
- Still missing (correction-) work

## References to Model-based Estimation (not for exam)

<http://www.surety.nl/images/P001 - Testpointanalysis, a method for test estimation tcm6-2506.pdf>

Software Cost Estimation with Cocomo II (with CD-ROM), Barry W. Boehm, Ellis Horowitz, Ray Madachy, Donald Reifer, Bradford C. Clark, Bert Steece, A. Winsor Brown, Sunita Chulani, Chris Abts, Prentice Hall 2000.

<http://sunset.usc.edu/research/COCOMOII/>

## 2.5 Test approaches

**Determine: When to prepare the test?**

- **Preventative** - as early as possible
- **Reactive** - after the product is developed

**(Chapter "Test approach" in a test plan following IEEE Std. 829-1998)**

## More approaches

- Analytical methods (for example risk-based testing - see later)
- Statistical testing (random-generated, with or without usage profile)
- Model-based testing (where one develops the test from a model of the program. For example state machines)
- Systematic, methodical ways, based on typical errors and faults, based on checklists (for example failure-based)
- Standard-based methods (industry standards)
- Heuristic or experience-based and dynamic methods, for example explorative testing, agile techniques
- Use of specialists (experts) or tools to guide the test
- Fighting regressions (side-effects) with large scale test automation and standard-test suites

**The approaches can be combined!**

**A testing strategy's most important property is the number of faults it uncovers as a function of work time expended.** Bertrand Meyer, *Seven Principles of Software Testing*, IEEE Computer Magazine, Aug. 2008

## How to Choose a Test Approach

**It depends on:**

- **Risks**
  - with the project
  - with the product (against users, others, environment)
- **Knowledge and qualification of testers about tools, methods**
- **Test project goals and mission**
- **Project attributes**
- **Type of application area and system**
- **External requirements / authorities / standards / rules**

**Quote from syllabus: "The selected approach depends on the context and may consider risks, hazards and safety, available resources and skills, the technology, the nature of the system (like custom build vs. COTS), test objectives, and regulations."**

## Exercise: Test Approach

You shall test in a "crisis situation".

The system, a bus information system with web interface, is "nearly" ready.

It shall be in production as soon as possible.

Which strategies do you choose?

Think, then discuss!

## 3 Follow-up and Progress Monitoring

### About

- Fault density
- Failure frequency
- Test control
- Test coverage
- Follow-up
- Test Reporting

1 Test Organization
1.1 Organization and Independence
1.2 Tasks for Test Personnel
2 Test Planning and Estimating
2.1 Foundations
2.2 Planning Activities
2.3 Exit Criteria
2.4 Estimation
2.5 Test Approaches
->3 Follow-up and Progress Monitoring
3.1 Progress and Follow-up
3.2 Test Reporting
3.3 Test Control
4 Configuration Management
5 Risks and Testing
5.1 Project Risks
5.2 Product Risks
6 Problem / Incident Handling



## 3.1 Progress and Follow-up

**Test must be visible!**

**Test shall give feedback to development!**

**Measurement data must be collected manually and with tools.**

**Used for**

- measuring exit criteria
- measuring progress against the plan (time, budget)

## Popular Measurement Data From Testing

**How much work is done / is to be done (f.ex. test cases, test environment)**

**Test cases passed / failed / blocked**

**Data about faults and failures (density, failure rate, repaired or not, results of re-test)**

**Test coverage (requirements, risk, code)**

**Subjective trust by the testers (bad!)**

**Milestones**

**Cost / benefit of testing**

## 3.2 Test Reporting

### End result from testing

For product owner, to make decisions and see what has happened in testing.

### Recommendation about the product / further test

What happened?

Was testing good enough?

How is product state?

What can we do better next time?

Template: IEEE Standard 829,  
See [www.wikipedia.org](http://www.wikipedia.org), search for IEEE 829

## Details

### Reporting about the testing

- Were the test objectives OK?
- Was the test good enough?
- Effectiveness of the test with respect to the objectives?
- When were the exit criteria fulfilled?

### Reporting about the product

- Open faults
- Open risks
- Would it pay off to test further?
- Trust in the product from the information collected during testing.

Information and measurement data shall support recommendations and decisions about future actions.

## Test Summary Report (IEEE 829-1998)

- 1 Test summary report identifier (title)
- 2 Summary
- 3 Variances (of test items and testing, with reasons)
- 4 Comprehensiveness assessment (test coverage)
- 5 Summary of results (incidents and resolution)
- 6 Evaluation (exit criteria, failure risk)
- 7 Summary of activities
- 8 Approvals

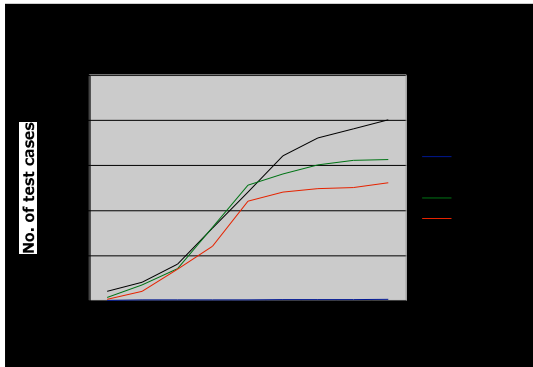
## 3.3 Test Control

How is testing changed along the way, based on information and measurement data? Consequences for testing and/or other activities with the product/system.

### Examples:

- Re-prioritization of tests
- Change of the schedule because test environment or other resources were not available
- More test where you found many faults
- Tougher start criteria if quality too low
- More test where test coverage was too low
- Step wise testing if time pressure
- Outsourcing of special tasks if time pressure
- Use of existing test material if time pressure
- Cutting of test if time pressure
- Follow-up re-test after changes and repairs

## Example (not for exam)



How would you have reacted after week 5?

Web links:

<http://home.c2i.net/schaefer/testing/teststatusreport.xls.hqx>

<http://home.c2i.net/schaefer/testing/testtrack.xls.hqx>

## Test effectiveness: Defect Detection Percentage (DDP)

$$\text{DDP} = 100 \times \frac{\text{\# of faults found in this test}}{\text{\# of faults found in this test} + \text{\# of faults found later}} \%$$

“This” test =

- One test level (f.ex. Component test )
- All testing
- One review
- Regression test

DDP = Defect Detection Percentage

The DDP is best measured over longest possible time, i.e. until about 6 months after delivery.

Useful to predict testing effectiveness in the next project and to control optimism.

# Fault Distribution

**Where in the system are the faults?**

**Why? (compared to other data)**

**Measure by counting number of faults per component divided by size (f.ex. complexity, length).**

Reference for predicting fault prone areas in general software systems:

Andersson, C., Runeson, P., "A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems", IEEE Transactions on Software Engineering, Vol 33, No. 5, May 2007. (Supports the hypothesis that the Pareto-principle of fault distribution holds, i.e. That a small number of modules contains most faults, and that this is not size-related. The study also supports that fault distribution in earlier test levels predicts fault distribution in later test levels.

Reference for predicting fault prone classes in object-oriented software:

Olague, H.M., Etzkorn, L.H., Gholston, S., Quattlebaum, S., "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Development Processes", IEEE Transactions on Software Engineering, June 2007.

The article showed good prediction performance on the following two metrics:

Metric CK-WMC – Weighted Methods per Class: Sum of complexities of the local methods of a class. For simple methods, the number of methods in the class.

Metric CK-RFC – Response for a Class: Count of all local methods of a class plus all methods of other classes directly called by any of the methods of the class.

The metrics were defined in Chidamber, S. and Kemerer, C., "A Metrics Suite for Object-Oriented Design", IEEE Transactions on Software Engineering, June 1994.

# Example for Distribution of Faults (not for exam)

## **Practical Solution: "Test Wall"**

(not for exam)

**Use a whiteboard central in the project group!**

**List test cases or groups (progress, faults, responsible for repair, status)!**

**List open and closed faults (cumulative and this period)!**

**Use colors!**

**Large enough writing to enable passers-by to read.**

## **How precise measurement?** (not for exam)

**Measurements and their evaluation cost money.**

**Thus:**

- **Measure only what you need and as precisely as you need!**
- **More only for research!**
- **Make the measurements available to the planners of the next project!**

# 4 Configuration Management

1 Test Organization	1.1 Organization and Independence
	1.2 Tasks for Test Personnel
2 Test Planning and Estimating	2.1 Foundations
	2.2 Planning Activities
	2.3 Exit Criteria
	2.4 Estimation
	2.5 Test Approaches
3 Follow-up and Progress Management	3.1 Progress and Follow-up
	3.2 Test Reporting
	3.3 Test Control
->4 Configuration Management	
5 Risks and Testing	5.1 Project Risks
	5.2 Product Risks
6 Problem / Incident Handling	

**Very important for the test!**

**A few words about it...**

**Goal: You know what you test, with which test material, and in which test environment.**

The purpose of configuration management is to establish and maintain the integrity of the products (components, data and documentation) of the software or system through the project and product life cycle.

During **test planning**, the CM procedures and infrastructure (tools) should be chosen, documented and implemented.

**Configuration Management is a necessary service testing is dependent on. Without it, testing ends in chaos.**

Ref: "Introducing Version Control to Database Centric Applications in a Small Enterprise", Jan Ploski, Wilhelm Hasselbring, Jochen Rehwinkel and Stefan Schwierz, IEEE Software Magazine, 1/2007

# Configuration Management - IEEE - Definition of four parts

IEEE Std. 828

**Configuration management establishes and maintains the integrity of the product to be tested.**

- **Identifying** and defining the configuration items in a system and their relationships (document and file names, version numbering, library structure, baseline / release planning)
- **Controlling change** and release of these items throughout the system life (version control)
- **Recording and reporting the status** of configuration items and change requests (traceability, impact analysis, status database, queries and reports)
- **Auditing**, i.e. verifying the completeness and correctness of configuration items (control with what has been built, tested and delivered)

## Some CM Principles

- EVERYTHING gets a unique identification (test cases, testware, documentation, system).
- Version numbering
- Version control
- Traceability
- Central decision where all involved parts are represented (“Change Control Board”)
- Check-in of the official version (control, test, regression test)
- Logging of check-in
- Check-out when something shall be changed (inclusive coordination)
- Backup of old version

Ref: <http://www.troyhunt.com/2011/05/10-commandments-of-good-source-control.html>

**The tester gets precise control with this!**

**Both software, test material and test environment become repeatable.**

**The test material must be under CM from its approval until the software is retired.**

## 5 Risks and Testing

1 Test Organization	1.1 Organization and Independence
	1.2 Tasks for Test Personnel
2 Test Planning and Estimating	2.1 Foundations
	2.2 Planning Activities
	2.3 Exit Criteria
	2.4 Estimation
	2.5 Test Approaches
3 Follow-up and Progress Management	3.1 Progress and Follow-up
	3.2 Test Reporting
	3.3 Test Control
4 Configuration Management	
->5 Risks and Testing	
	5.1 Project Risks
	5.2 Product Risks
6 Problem / Incident Handling	

**We never get enough resources to do all testing we want.**

**We have to follow up risk and prepare for it -> Risk management**

**We must prioritize! -> risk-based testing!**

**Think about:**

- **Product Risks**
- **Project Risks**

- **Paper:** [http://www.testing-solutions.com/library/downloads/r/-Risk%20Based%20Assurance%20&%20Acceptance%20Test%20Management%20Suite%20v1\\_0.pdf](http://www.testing-solutions.com/library/downloads/r/-Risk%20Based%20Assurance%20&%20Acceptance%20Test%20Management%20Suite%20v1_0.pdf)



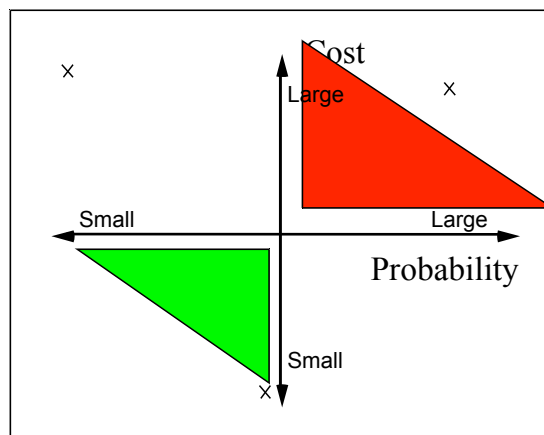
# Risks

Product of  
probability for some damage  
and  
cost of damage

Rough estimates in general good enough!

**Risk = Probability of trouble \* Cost of trouble**  
(Def from IEC 61508:2000)

# Risks, graphical



## Goal of risk-based Testing

**When you finish testing, your stakeholders should have as much information as possible about the quality of the product, the risk of using it, and you shall have done the best possible testing.**

**Risks threaten the achievement of one or more stakeholders' project objectives.**

Detail information on <http://www.softwaretesting.no/testing/risktest.doc>

## Project Risks

What can go wrong with the test project?

### Supplier

- Outsourcing, offshoring, not delivered
- Bad contract
- Organization
- Not enough validation
- Not enough people
- Time pressure (delays)
- Not enough resources
- Political problems
- Wrong attitude towards testing

### Technical

- Difficult to understand requirements
- Too advanced requirements
- Quality problems with the products the test is using (specifications, design, code, test material)

### Resources:

- Test equipment failures
- Test lab not ready on time

### Politics:

- Fighting between people
- Fighting inside the project
- Testers vs. developers
- Testers not able to tell their requirements
- No follow-up of review- and test results
- ...

## How to Deal with Project Risks?

Test plan - item "Risks and Contingencies"

**Risk Identification: Find what can go wrong**

**Risk Analysis:**  
Weight: Top 10-list  
Think about possible reactions

**Risk Mitigation: Active Follow-up**

## Product Risks

**Risk-based test = Concentration on product areas with highest risk in use.**

- > large potential to harm people or environment
- > areas too bad (attribute requirements)
- > lack of expected functionality
- > exposed to (user) errors
- > failure-prone software delivered
- > poor data quality

**A product risk occurring may lead to project risk and vice versa.**

**High risk -> Test first and test most.**

## The most Common Risk Factors (Fault generators)

- Unstable functions, interfaces, data definitions
- Too many cooks
- Many faults earlier
- Fighting between people, ...
- Personnel turnover
- Less qualified people
- Complexity (size, interactions, logic)
- New technology
- The controllability of the environment (real time program, ...)
- Time pressure
- Reorganization during development
- etc.

## Damage: What to Look for?

How often used?

How visible?

How critical is the use? (possible direct damage)

How difficult to find a way round?

## Example for Prioritization with Spreadsheet

Factors for damage      Factors for fault density

Area to test	Usage frequency	Visibility	Complexity	Geography	Turnover	SUM
Weight	3	10	3	1	3	
Function A	5	3	2	4	5	<b>1125</b>
Function A performance	5	3	5	4	5	<b>1530</b>
Function B	2	1	2	2	5	<b>368</b>
F B usability	1	1	4	2	5	<b>377</b>
Function C	4	4	3	2	0	<b>572</b>
Function D	5	0	4	1	1	<b>240</b>

Spreadsheet on <http://www.softwaretesting.no/testing/riskcalc.xls>

## How much to Test?

### Three levels:

- **Very high risk: Very comprehensive test (combinations, check of test coverage)**
  - -> Test first
- **Medium risk: Medium comprehensive test**
  - -> Test later
- **Everything else: "A little" (smoke test)**
  - -> Test last

## What Testing is Used for

- Identify product risks -> control the test process
- Reduce the probability of product risks
- Reduce the impact of product risks
- Support decisions about which risk factors should be reduced
- Improve the knowledge about risk factors (use other people's experiences and knowledge about risk factors)

**Circular, iterative, process:**

**Risk (management) -> test planning -> test execution -> risk management**

## Risk Management

**Continuously evaluate and re-evaluate the risk**

**Prioritize risks**

**Risk mitigation -> Follow-up -> Report**

**Find new risk factors or changes to them**

**ITERATIVELY!**

## What to Use Risk for

**Proactive method to reduce the level of product risks, from project start**

**Identify product risks in order to take care of them throughout the test process**

**Use supplier knowledge about risk factors**

**Prioritizing the test - find critical faults as early as possible!**

**Selection of test techniques (more or less comprehensive)**

**Feedback to the project to prevent risk**

- Other order of delivery to testing
- Training of developers
- Extra reviews
- ...

## Risk-based Test Report

**Concentration on risk areas**

**Predict remaining risks**

- 1. These risks are gone...**
- 2. The following risk is still this high...**
- 3. The total risk is ...**
- 4. Possible alternatives (more test, usage restrictions)**

## Exercise (5 min)

Find typical causes for higher frequency of faults in  
**YOUR** organization.

Project ...

Technical ...

## 6 Problem / Incident Management

1 Test Organization	1.1 Organization and Independence
	1.2 Tasks for Test Personnel
2 Test Planning and Estimating	2.1 Foundations
	2.2 Planning Activities
	2.3 Exit Criteria
	2.4 Estimation
	2.5 Test Approaches
3 Follow-up and Progress Management	3.1 Progress and Follow-up
	3.2 Test Reporting
	3.3 Test Control
4 Configuration Management	
5 Risks and Testing	5.1 Project Risks
	5.2 Product Risks
<b>-&gt;6 Problem / Incident Handling</b>	

Test shall find potential problems and faults.  
Problems/faults/incidents must be logged, prioritized,  
analyzed, followed up, corrected, tested (not necessarily in this order!).

Systematical handling requires:

**Process and rules for classification and prioritization**

Problems not only in code, also in other documents,  
**EVERYWHERE!**

From development, reviews, test, maintenance, anything!

Ref: <http://www.amibug.com/iamabug/p01.html>



## Problem / Incident Management: Goal

Trace problems (organize fault repair).  
Feedback to developers (info about problems)  
Measure quality under test and test progress.

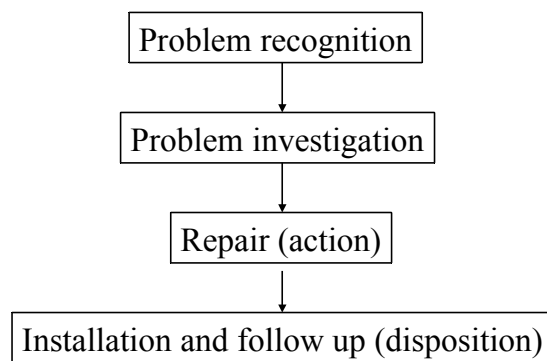
**The system shall be useful for all involved parties!**

Use data for (test)process improvement

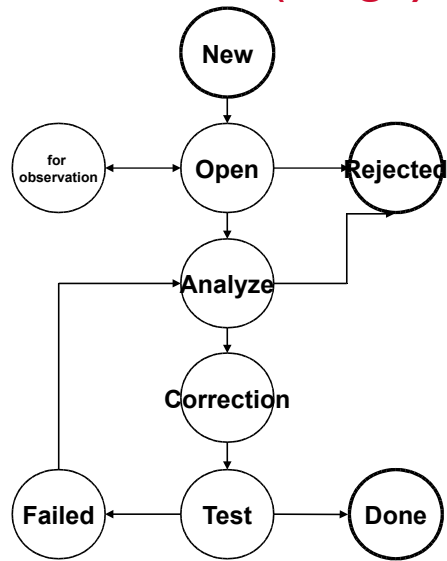
- To optimize testing in this project
- To optimize methods in the longer term
  - Which faults do we find / or not?
  - Costs?
  - Why?
  - Can we do better?

## Process for Problem / Incident Management

IEEE Standard 1044



## Problem / Fault Status (rough)



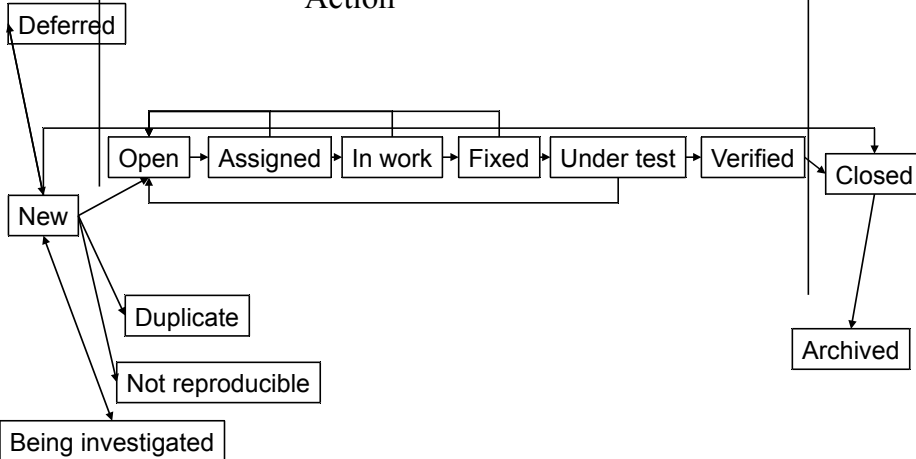
## Only for info: Details from IEEE

Std 1044

Recognition,  
Investigation

Action

Disposition



## Classification and Prioritization of Faults

- **Severity: Impact for customer / user / testing (max 5 classes)**
  1. Impact outside the system ("fatal") or blocking testing
  2. Crash
  3. Serious without "workaround" (destroying customer value)
  4. Serious with "workaround"
  5. Minor
- **Visibility of fault (How many users will see it?)**
- **Difficulty to fix it (correctly)**

## Prioritizing Fault Repair

**"Triage"**

**Not all faults must be fixed immediately!**

**Priority classes:**

1. Immediately ("Patch")
2. Next release
3. Later

# Management Information - Trends

## Faults and repair

# Contents of a Problem / Fault Report (1)

## 1. Identification

- Number
- Short description, title or name
- Date, author (+ details of author), organization
- Identification of product (version, configuration, document...)
- Identification of environment (platform etc.)
- Life cycle process where the problem was found

## 2. Description

- Details
  - Which test case(s) led to the problem?
  - Or: Input, expected and actual result, logs, database dumps etc.
  - Maybe: Can the problem be repeated, how?
- Degree to which the problem impacts the stakeholders (users, customers etc.)
- Author interpretation of severity and priority
- Reference to other / related problems / global impacts

## 3. Classification

## 4. Info from repair, acceptance

## Contents of a Problem Report (2)

### 3. Classification

- Status
- Severity
- Priority

### 4. Info from repair

- Developer
- Cause
- Change history, what was done with the problem?
- Remarks
  - Conclusions and recommendations
  - Other areas which could be impacted
  - Consequences for the owner
- Approval

-> Exercise

## What to think about when you find a problem (not for exam)

Maybe the problem is more general than you first think?

More serious consequences?

Only on a special configuration or with special settings?

Is it a problem in other contexts?

Try to find out the **WHOLE** information about the problem!

Everything you get to know!

## Tasks for your project (not for exam)

Make a model for problem status for your project!

Which priority classes for problems do you want?

How can you get experience data from your system for problem handling?

What can lead to people avoiding your system for problem handling?

## Questions?

## Information and literature

Hetzel, W. C.: The Complete Guide to Software Testing, 2<sup>nd</sup> ed., John Wiley & Sons, 1988.

Caner C., Falk, J., Nguyen, H. Q.: Testing Computer Software, 2<sup>nd</sup> ed. John Wiley & Sons, 1999.

A good book for beginners, especially in projects under time and market pressure and consumer software production. Contains some good advice for system testing. Good list of possible defects. Very practical.

Glenford Myers: "The Art of Software Testing", New York, John Wiley, 1979.

The classic book about software testing. Still, more than half of it is important knowledge. Some good advice for every test level, and the basic ideology.

Koomen, T., Pol, M.: Test Process Improvement: A Practical Step-by-Step Guide to Structured Testing, Addison-Wesley, 1999.

Pol, M., Teunissen, R., van Veenendal, E., Software Testing, A Guide to the Tmap Approach, Addison-Wesley, 2002.

This is the standard way to organize testing in the Netherlands. A good guide for managing development of application software.

Royer, T. C.: Software Testing Management, Prentice Hall, 1993.

Better Software Magazine, [www.bettersoftware.com](http://www.bettersoftware.com) . [www.stickyminds.com](http://www.stickyminds.com) Very practical!

van Veenendal (ed.), E., The Testing Practitioner, UTN 2003, ISBN 90-72194-65-9.

Tilo Linz, Andreas Spillner, Hans Schaefer, Software Testing Foundations , dpunkt Verlag 2006 (German and Dutch edition available)

Rex Black, Critical Testing Processes, Addison-Wesley, 2003.

Rex Black, Managing the Testing Process.