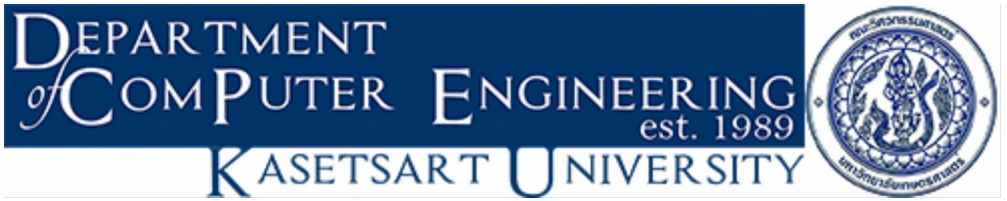


Success factor Software Testing

Uwe Gühl

Winter 2015 / 2016

Contents

- Introduction
- Software testing: What are success factors?
    1) Risk Based Testing
    2) Requirements
    3) Prioritization
    4) Communication
    5) Early Testing
    6) Reviews
    7) Test automation
    8) Test Know-how

Introduction

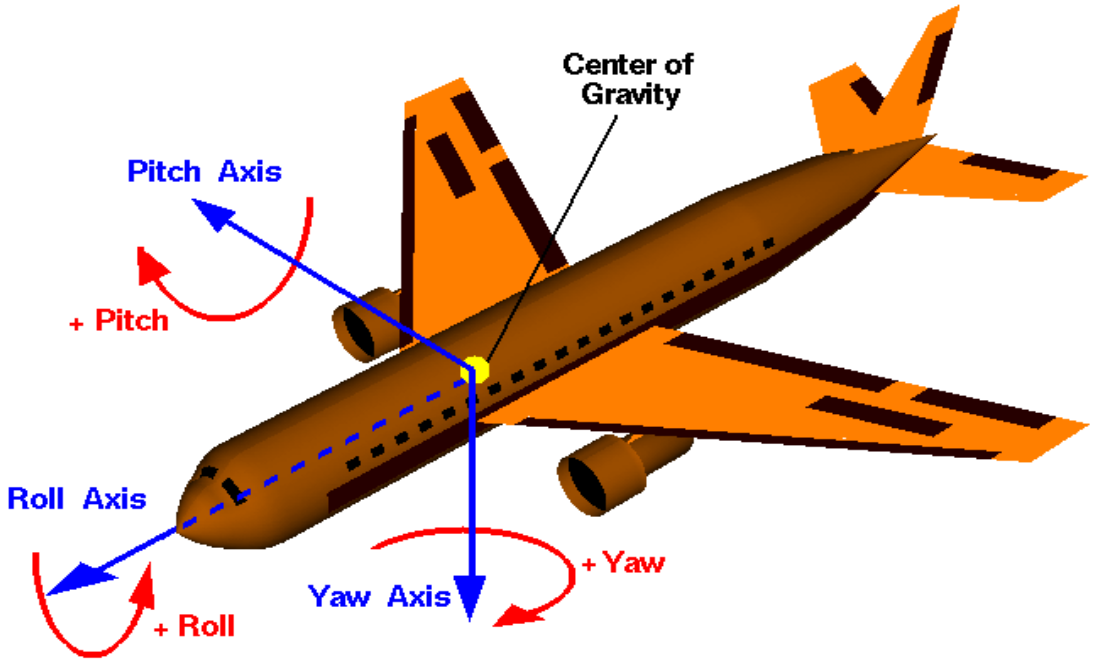
(Fatal) software defects

- 1996 a prototype of the Ariane 5 rocket of the European Space Agency was destroyed one minute after the start.
- Reason:
  The code of the Ariane 4 was used.

Introduction

(Fatal) software defects

- In 1982 there was a crash of a Lockheed F-117A Night Hawk during take off
- Reason:
  The fly-by-wire system had been hooked up incorrectly ("yaw rudder" was used instead of "pitch elevator" and visa versa)



(Image source: NASA, http://en.wikipedia.org/wiki/File:Rollpitchyawplain.png Public domain)

Introduction

(Fatal) software defects

- 2012 Knight Capital lost about $440 million in 45 minutes
- Reason:
  Because of a defect in untested released software the program rapidly bought and sold millions of shares accidentally – resulting in a big loss [Wik16]

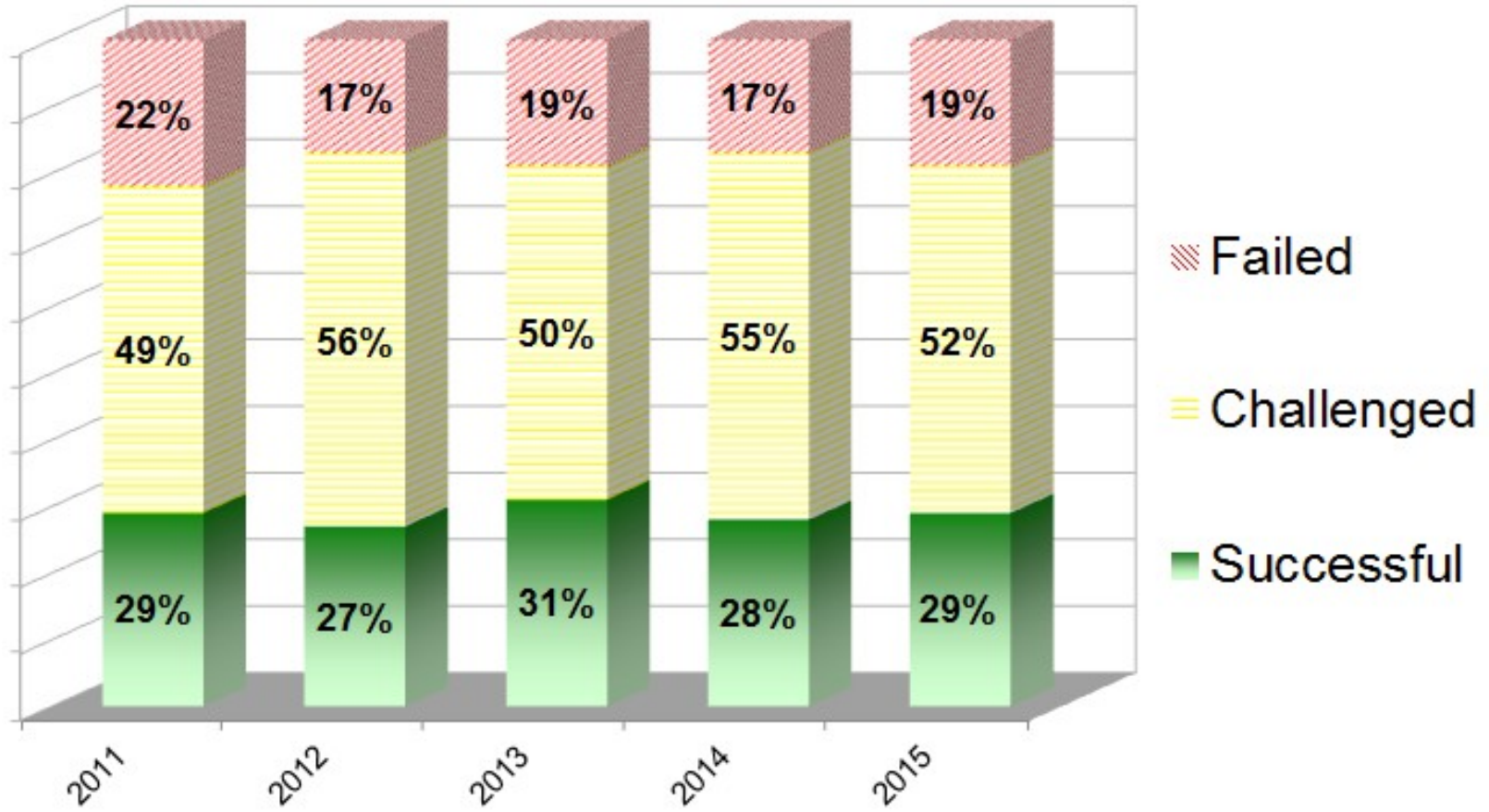
Introduction

Result of an analysis of the Standish Group, Chaos Report 2015 [HW15]

- Failed
  The project is cancelled at some point during the development cycle.
- Challenged
  Cost or time overruns or didn’t fully meet the user’s needs
- Successful
  On time, on budget with a satisfactory result

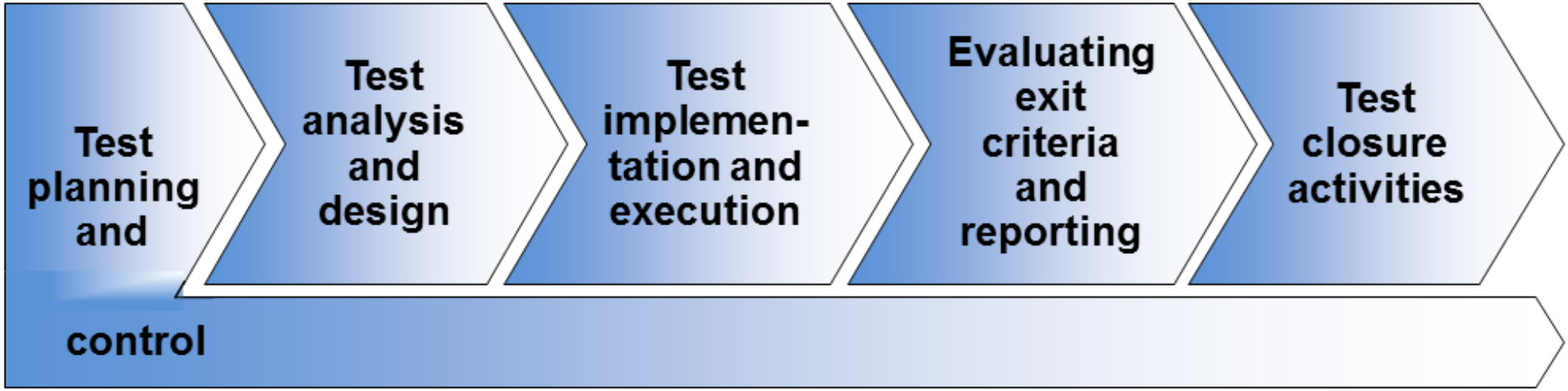
Introduction

Result of an analysis of more than 9000 IT projects
(Standish Group, Chaos Report 2015) [HW15]



Introduction

Fundamental test process [IST16]



Introduction

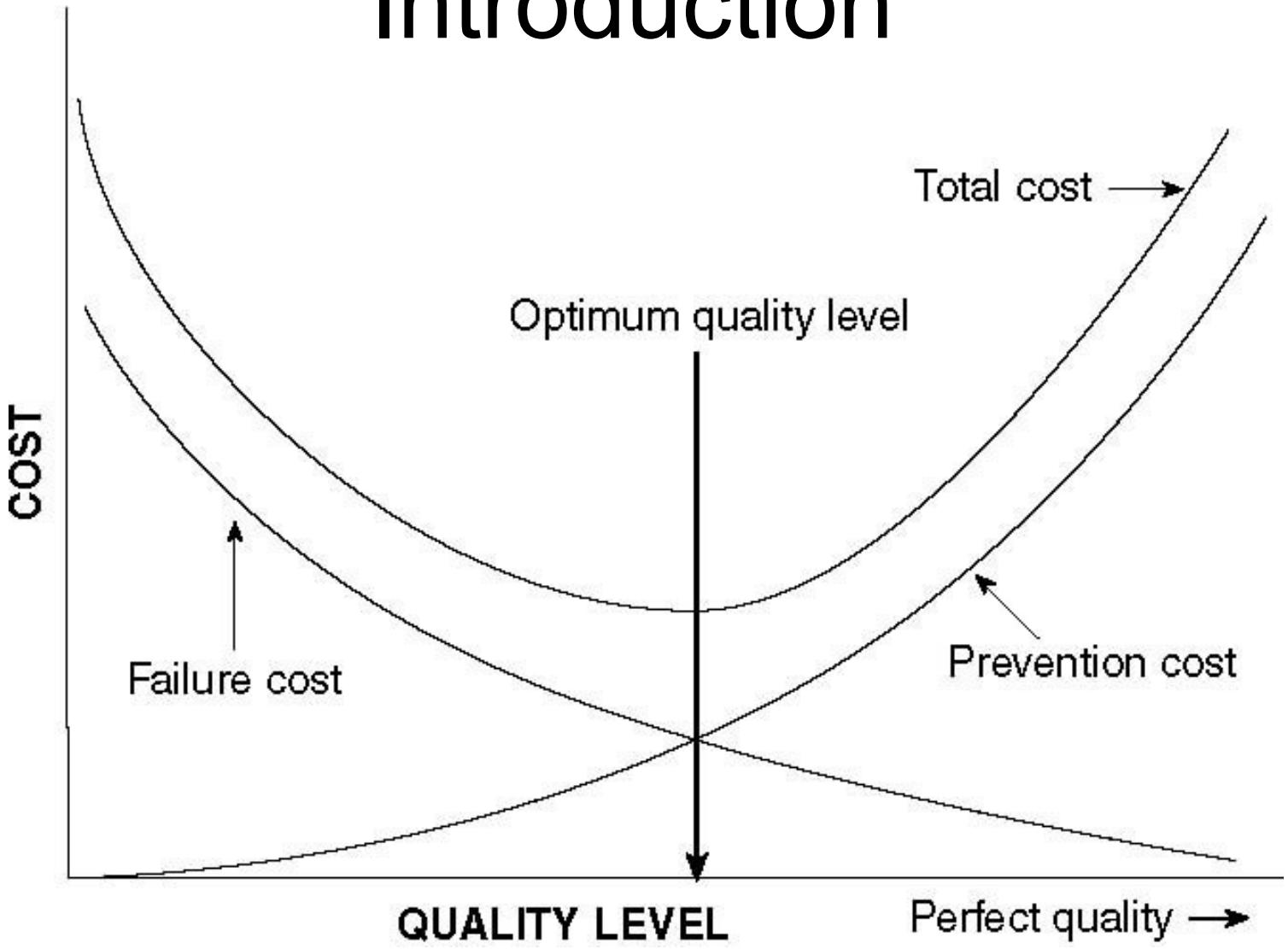


Image source: http://qualityamerica.com/images/figure_vi39_qualitycosts.jpg

Figure 8.3. Classical model of optimum quality costs.
From Jurans Quality Control Handbook, 4th edition. J.M. Juran, editor.
Copyright © 1988, McGraw-Hill

Introduction

Effort of software testing as part of software development

- The cost of testing is up 40 to 60 percent of the total project costs, depending on the required level of quality [SJ06]
- 1979, and in 2012, in a typical programming project approximately 50 percent of the elapsed time and more than 50 percent of the total cost were expended in testing the program or system being developed [MSB12]

Introduction

- General: Preventing defects is more efficient than fixing
  Prevention, ... not cure
- The earlier a defect is detected, the cheaper is the correction
- Cheapest are defects, that don't occur at all
- To be considered:
  You can't test quality into the product; it must be built in
- Idea: Increasing quality „from scratch“ with corresponding measures:
  E. g. early reviews of requirements, code, ...

Costs of defect fixing

Phase	Relative Cost to Correct
Definition	1 $
High-Level Design	2 $
Low-Level Design	5 $
Code	10 $
Unit Test	15 $
Integration Test	22 $
System Test	50 $
Post-Delivery	100 $

Based on [Dus03]

Risk Based Testing

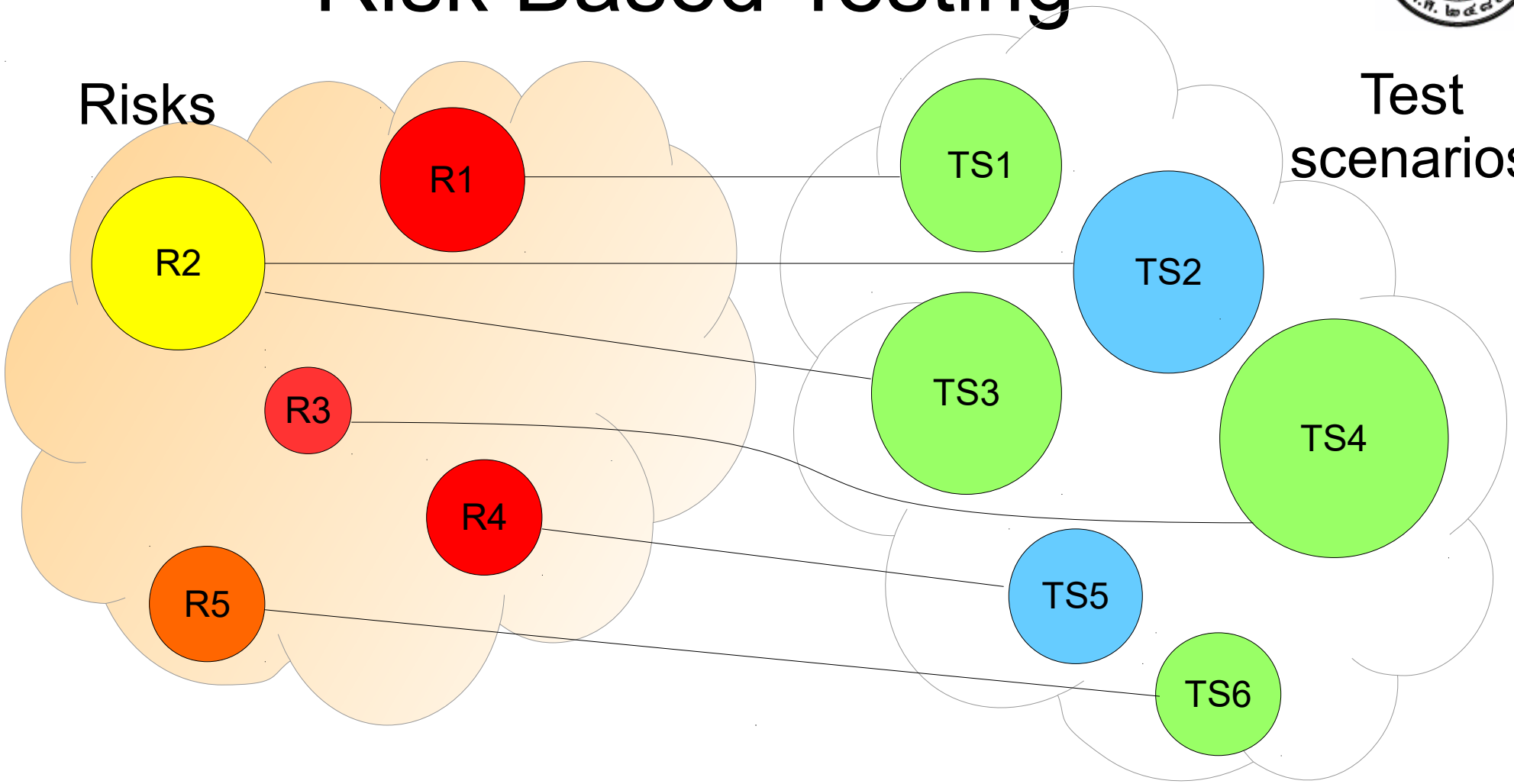
Definitions by [IST15]

- Project risk
  - A risk related to management and control of the (test) project, e.g. lack of staffing, strict deadlines, changing requirements, etc.
- Product Risks
  - A risk directly related to the test object.
- Risk based testing
  - An approach to testing to reduce the level of product risks and inform stakeholders of their status, starting in the initial stages of a project. It involves the identification of product risks and the use of risk levels to guide the test process.

Risk Based Testing

- Approach based on product risks
  What is the worst that can happen?
  - Collection and regular update of all possible product risks
  - The higher the risk the more corresponding tests to be prepared and executed
- Purpose of testing: Focus on high risk areas, especially on corresponding requirements
- Focus on concrete examples
- Simple summary: No risk, no test

Risk Based Testing



Risks (R1, R2, .. , R5) to be covered by corresponding test scenarios (TS1, TS2, … , TS6)

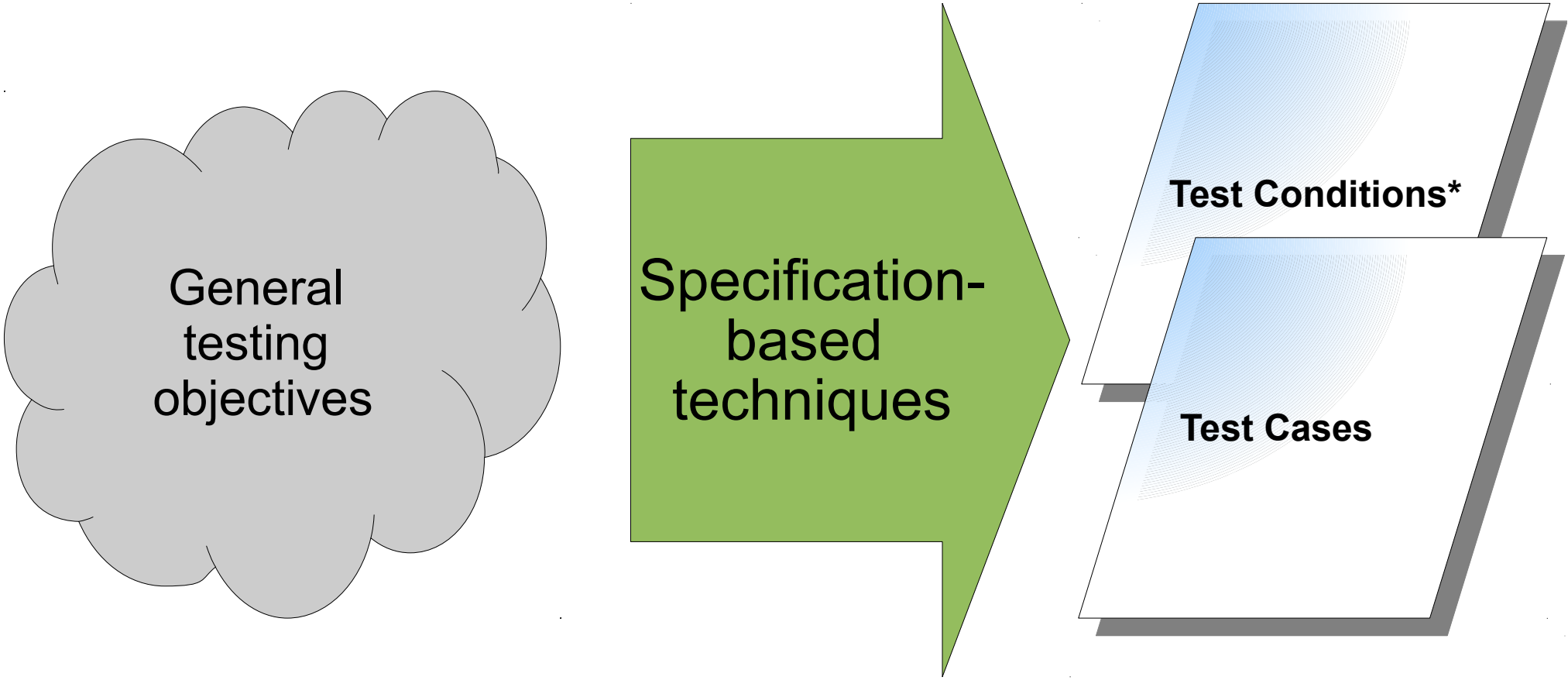
Requirements

- Take joint responsibility on requirements
- Why?
  - Requirements and testing work together
  - Typically requirements are the most important test basis
  - Out of a global survey about 48% of developers cited changing or poorly documented project requirements as the reason for failure [ADA15]
  - Often root cause of defects in IT projects: Requirements [Ric05]

Requirements

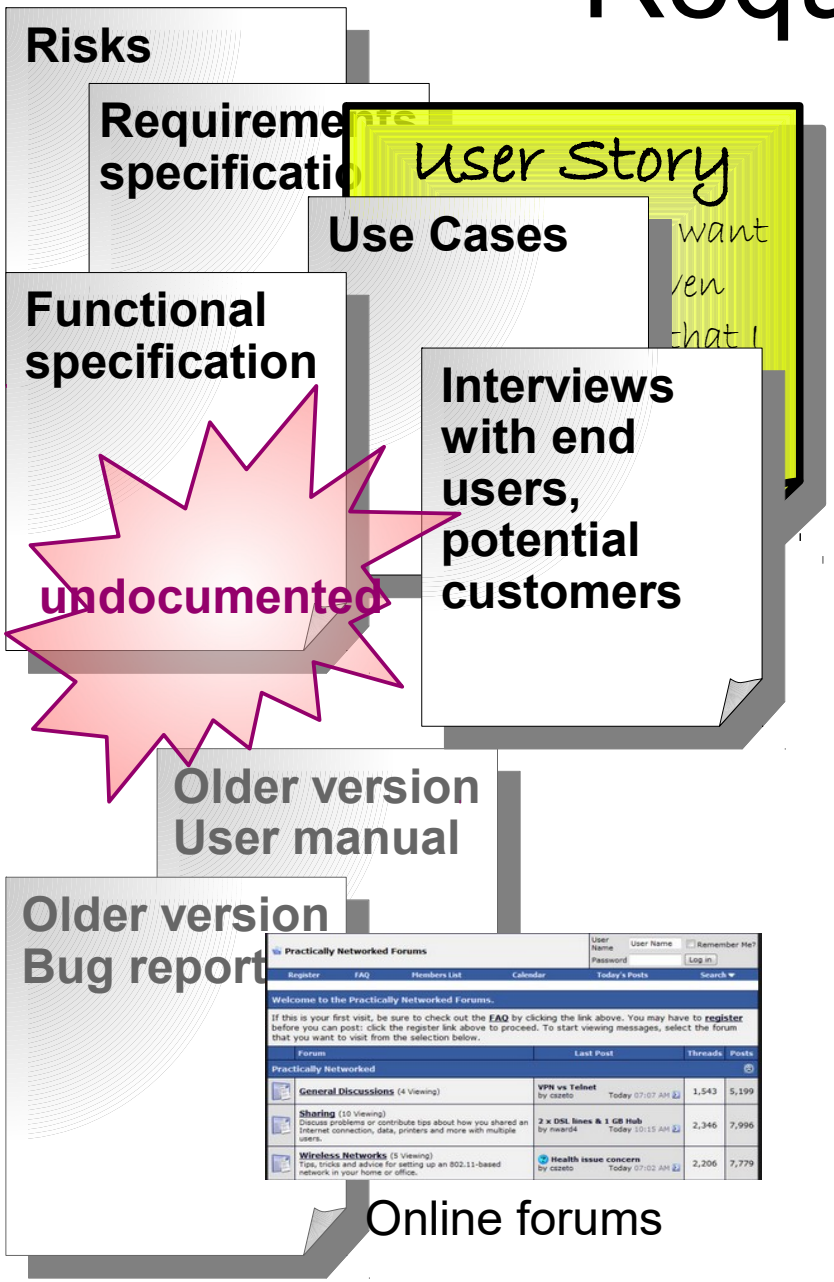
- What to do?
  - (In-official) order to the test team: Help to
    - clarify requirements, find gaps, inconsistencies
    - determine acceptance criteria for requirements
  - Testers have to identify the most crucial and most risky requirements => to be tested first
  - Activities to be done, if requirements are missing or not clear, especially non-functional requirements
    - Clarification of business scenarios
      => Basis for test scenarios
    - Identification of functional / non functional requirements, e.g. using quality model defined by ISO 9126 [Wik16a]
      => Determination, which test type to use

Requirements



* Test condition = An item or event of a component or system that could be verified by one or more test cases, e. g. a function, transaction, feature, quality attribute, or structural element [IST15].

Requirements



Specification-based techniques

Test Conditions*

Test Cases

* Test condition = An item or event of a component or system that could be verified by one or more test cases, e. g. a function, transaction, feature, quality attribute, or structural element [IST15].

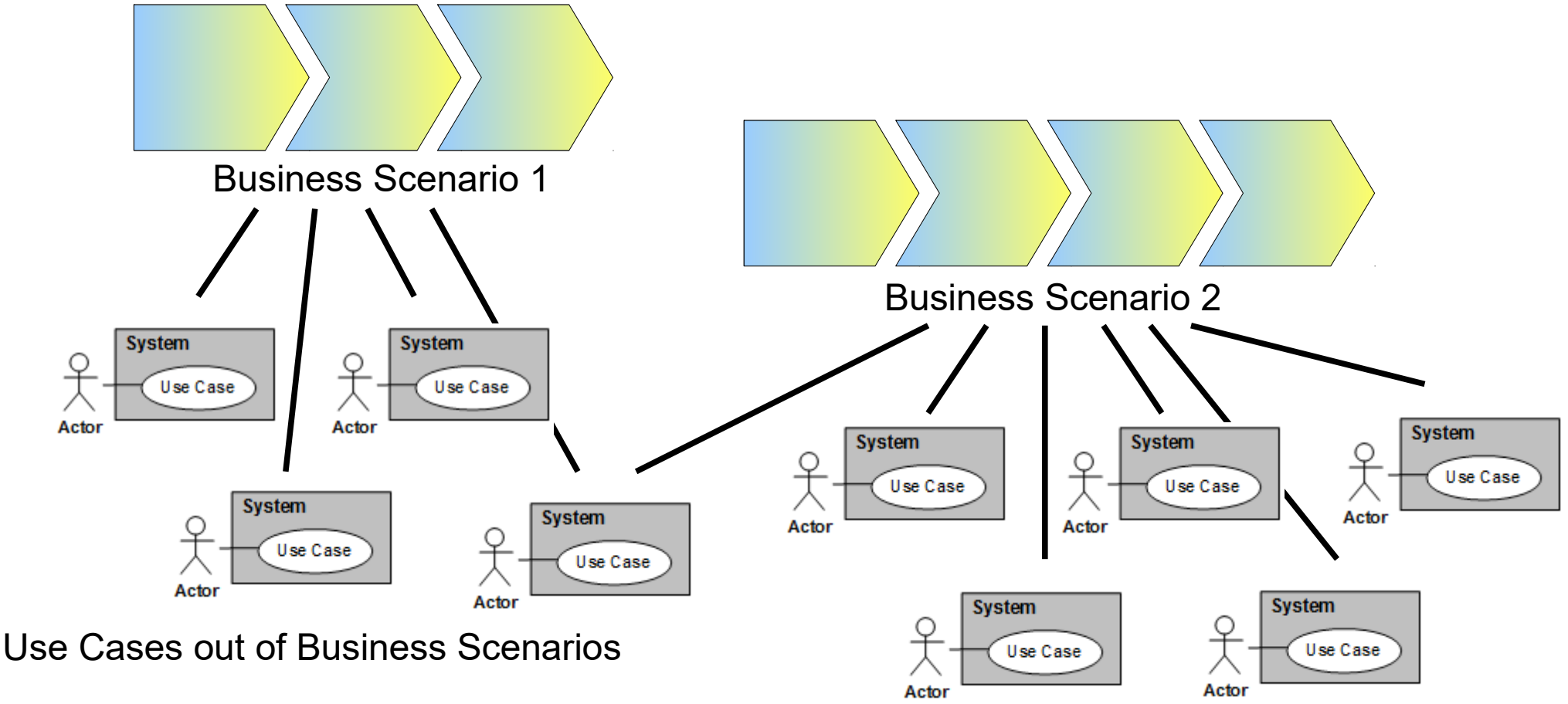
Requirements

- How to identify requirements / risks?
  - Interviews with stakeholders
    E.g. Sales, end user, project manager, ...
  - Definition of Business Scenarios
    … to identify business needs
    … to define use cases (Top down approach)
    … to prioritize testing activities

    e.g. in a corresponding workshop

Requirements

Top-Down Approach: Identifying requirements (here: Use Cases) out of Business Scenarios



Use Cases out of Business Scenarios

Requirements

- Definition of acceptance criteria
  - Helpful: Concrete examples.
  - Out of it: Define test cases to be passed.
- Excerpt (out of agile software development):
  "Definition of done" is an agreement to decide, when a realization of a requirement could be accepted by the customer.
  E.g. presentation successful, automated test cases passed.

Requirements

- Prioritization of requirements
  - High priority: **Must** – to be realized in the next iteration, e.g. product release.
  - Medium priority: **Should** – necessary.
  - Low priority: **Could** – Nice to have if there is enough time.
- High risk areas and high prioritized requirements result in corresponding high prioritized test cases.

Requirements

- Unknown Non-functional Requirements are a big risk in IT projects, if so called “self evident requirements” are not fulfilled (security, performance, load).
- Specification documents often leave the area “Non-Functional Requirements” empty or imprecise (“fast”, “easy to use”, “secure”)
  → IT Architecture cannot follow conditions.
  → No proper test planning.
- Proposal: Early identification of non-functional requirements!

Requirements

- ISO/IEC 9126 Software engineering – Product quality [Wik16a]
  - was an international standard for the evaluation of software quality – focusing on the product
  - tries to develop a common understanding of the project's objectives and goals
  - applies to characteristics to evaluate in a specific degree, how much of the agreements got fulfilled

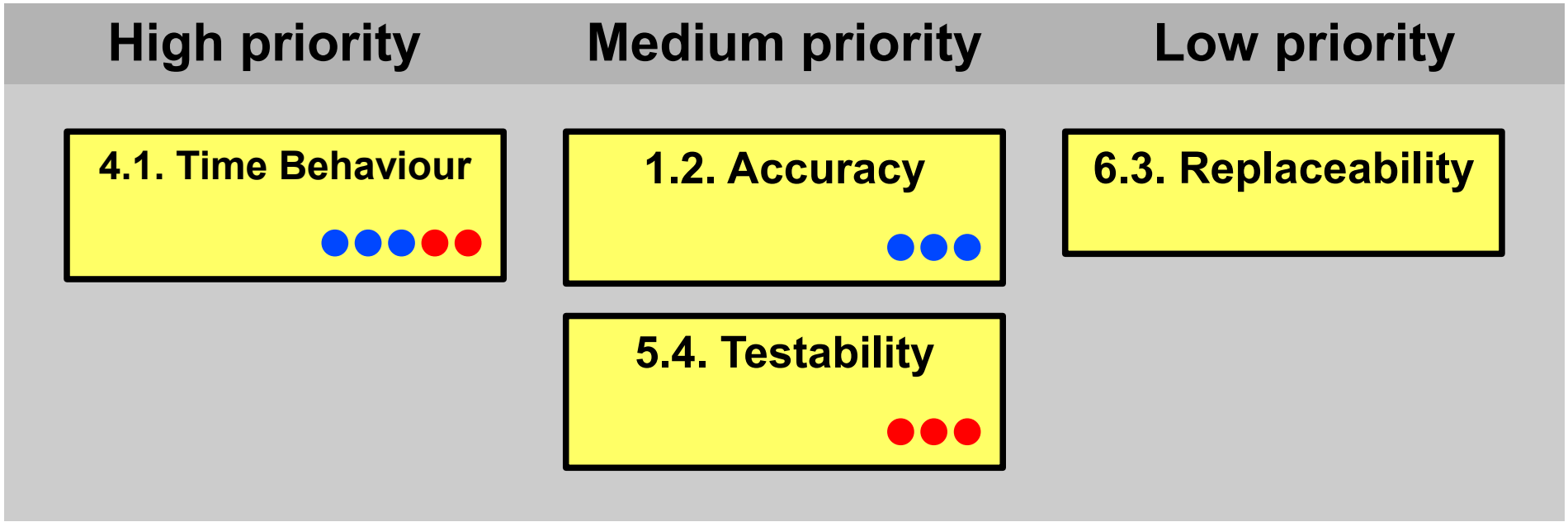
- Hint: Since 2011 there is a successor available:
  ISO 25010:2011 has eight product quality characteristics (in contrast to ISO 9126's six), and 39 sub-characteristics

Requirements

ISO/IEC 9126 Quality Model

Requirements

- Proposal: Performing a work shop following ISO/IEC 9126
- Result: * Prioritization of quality criteria



* List of corresponding requirements including acceptance criteria

Prioritization

Task:

- Testing of a simple program with three integers, up to 16 Bit
- Every combination should be tested
- Duration with assumption 100.000 tests / second

Solution:

- $2^{16} * 2^{16} * 2^{16} = 2^{48}$ combinations
  = 281.474.976.710.656 combinations
- Duration: About 90 years

Prioritization

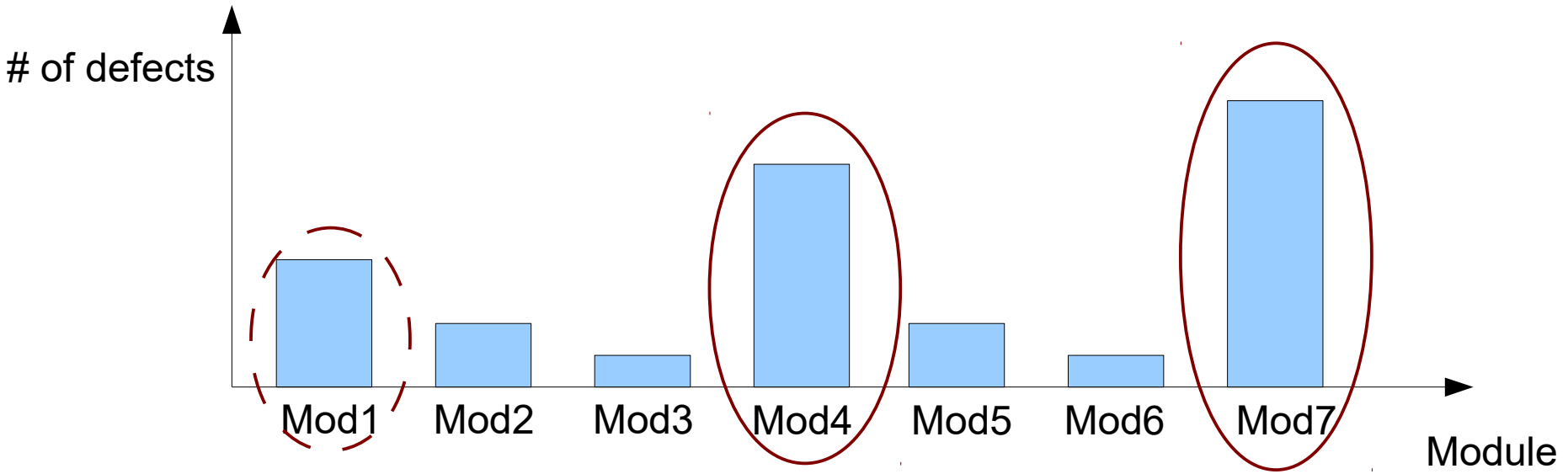
- So: You can't test everything
- What to do?
  - Risk based testing
  - Smart test design techniques (equivalence partitioning, boundary value analysis)
  - Prioritization
    “Prioritise tests so that, when ever you stop testing, you have done the best testing in the time available” (ISEB testing foundation course material 2003)
  - Always focus on the most important and most risky requirements

Prioritization

- Which tests to plan, prepare and execute first?
  - The higher the risk, the earlier
  - The higher the value for business, the earlier
  - Positive tests (instead of alternative tests / negative tests)
- Rule of thumb: Work with three priorities
  - About 10% of all test cases should get priority 1
  - About 50% to 70% of all test cases should get priority 2
  - About 20% to 40% of all test cases should get priority 3
- Regular check and update of priorities
- Plan exploratory testing as well
  … could be source for additional test cases

Prioritization

- A small number of modules usually contain most of the defects.
- Defect clustering is based on the
  Pareto principle – the 80-20 rule.
  Approximately 80% of the problems are caused by 20% of the modules [Jaw13].
- Update of prioritization if required



Prioritization

- Defects
  - Priority (test execution related): High prioritized defects to fix first
  - Severity (business impact): Most severe defects to fix first
- Ensuring that high prioritized tests get executed first, the probability increases that severe defects get detected early
  ==> Enough time to fix and retest

Communication

- Communication is a (if not the) key to project success

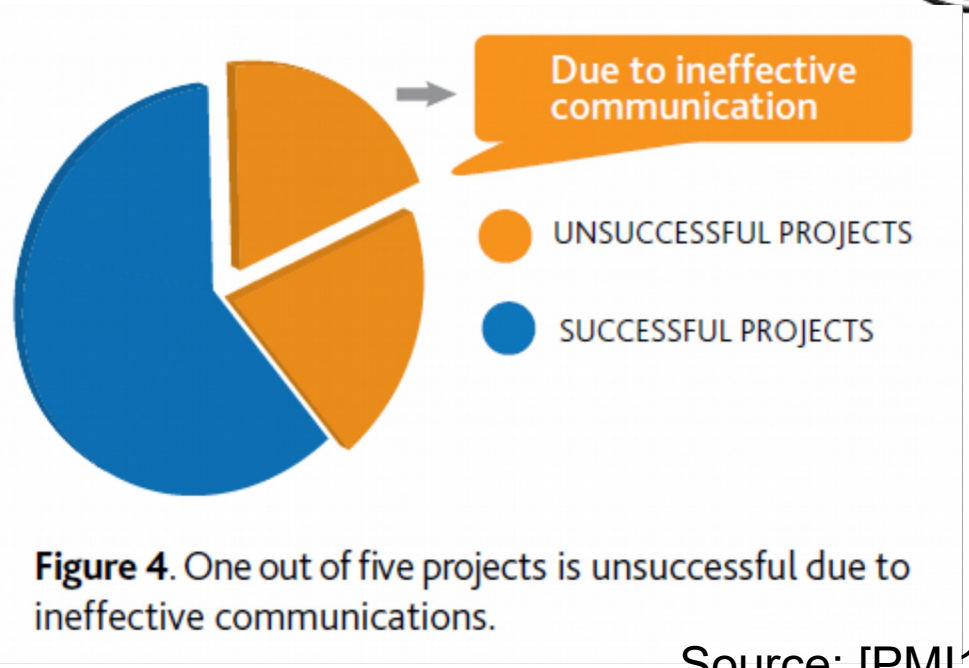


Figure 4. One out of five projects is unsuccessful due to ineffective communications.

Source: [PMI13]

- Enforce communication
  Requirements Engineer ⇔ Developer ⇔ Tester
- Role of testers:
  - Helping to deliver software successful
  - Not gate-keepers, but continually improve

Communication

Main goal of communication

- … during test planning/ test analysis and design
  - To identify scope:
    Customer, principal, project manager
  - To identify risks:
    Test team, developer, sales, architect, end user, people related to similar projects, investigation
- … during test implementation and execution
  - early detection of issues
  - update of test activities based on current test situation
  - controlling testing

Communication

- Learning attitude
  - Feedback / Retrospective / Lessons learned
    - Use your and your fellows experience:
      People know already – ask and transfer
    - Use experience out of project team:
      Regular lessons learned (workshops) with measures
  - Establishing a failure culture:
    Better we detect the defect than the customer
  - Involve people, end-user, operation

Communication

- Meetings
  - Try to install regular meetings concerning test topics
    - Test analysis and design: Once or twice a week
    - Test implementation and execution: Daily
  - Better often regular short meetings instead of seldom irregular long meetings
  - Follow a simple approach in the meetings;
    every participant should be involved in testing and report
    - What did I achieve?
    - Next steps
    - Current issues

Communication

- Documentation:
  - **Test plan** to communicate with principal / project team
    - Consider: A (test) plan is always wrong,
    - Worst than a wrong test plan: A dead test plan
    - Goal of test planning is not the test plan but doing test planning – Understand what to test how intense.
    - Basic estimations
      - Scope: How many test cases and defects expected?
      - Schedule: Based on expected time for test design and test execution, and expected defects

Communication

- Documentation:
  - **Test report** to communicate to all project stakeholders
    - Alignment concerning needs and contents with principal in advance
    - Consider: The test report is the working result and business card of a tester
    - Contents is based on test plan: Plan/actual comparison
      - Test coverage
      - Defect situation
      - Quality statements
      - Issues and risks

Early Testing

Consider and organize testing from the project start on, as testing is much more than test execution:

- Test planning
  Test order, test plan (cooperation with customer / development, standards to follow), schedules, organization, set-up of test team, test tools, ...
- Test analysis and design
  Test environment, configuration management, review of specifications and requirements, test scenario / test case design, test data design, …
- Test execution with static test techniques
  Review, code coverage, quality and complexity of code

Early Testing

Test team set-up

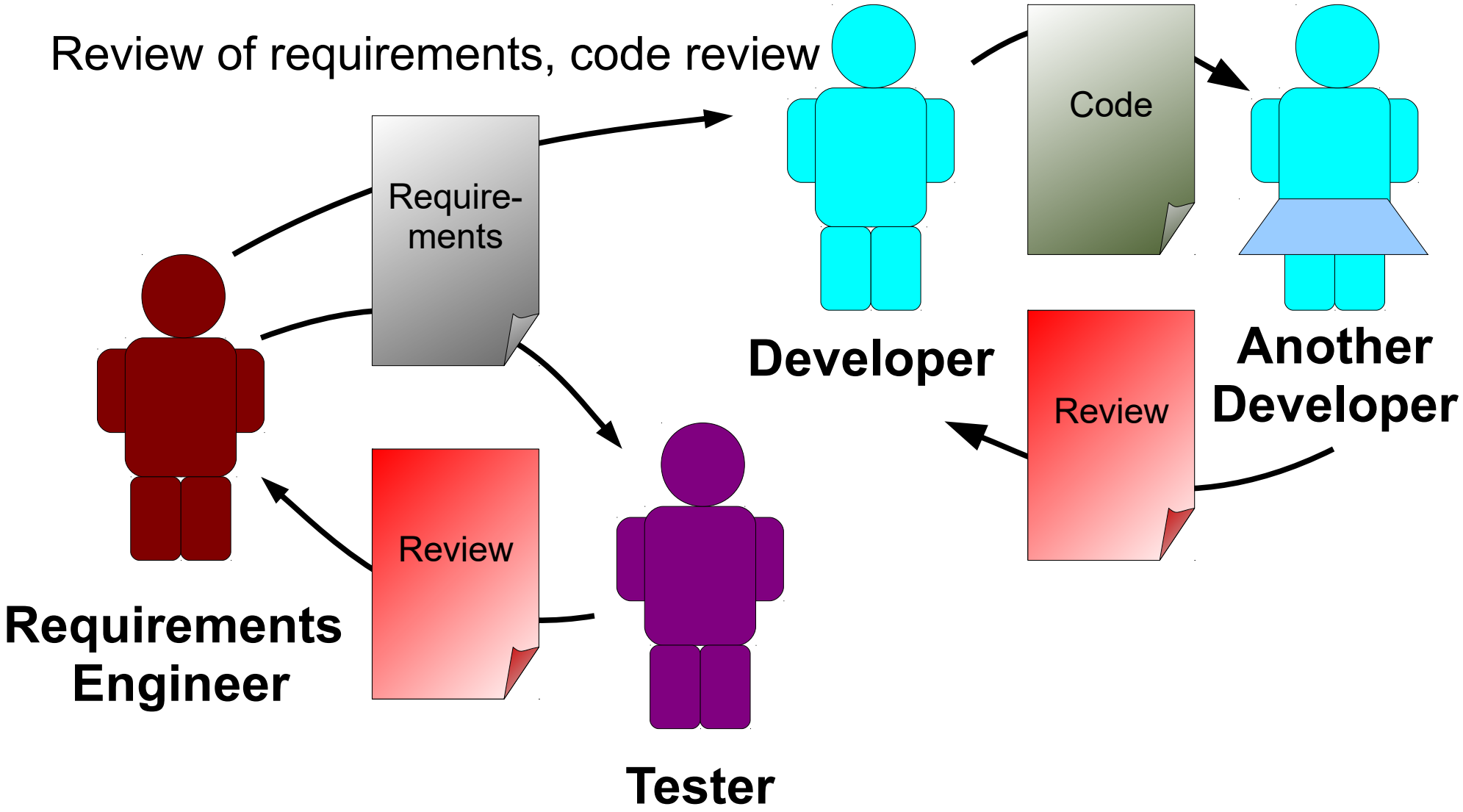
- Identification of roles and skills required for testing; team mix
  - Young / old
  - Male / female
  - Different background (technical, business, user / operational representative, testing experience {functional, security, operational, performance})
  - International
- (Business) Knowledge Transfer Sessions
- Training
- Ensure cooperation with customer, management, developer, operation

Reviews

- Reviews help to
  - clarify requirements,
  - reduce project costs in detecting defects early,
  - gain understanding,
  - educate testers and new team members.
- Different types of reviews possible like
  - Informal Review
  - Walkthrough
  - Technical Review
  - Inspection

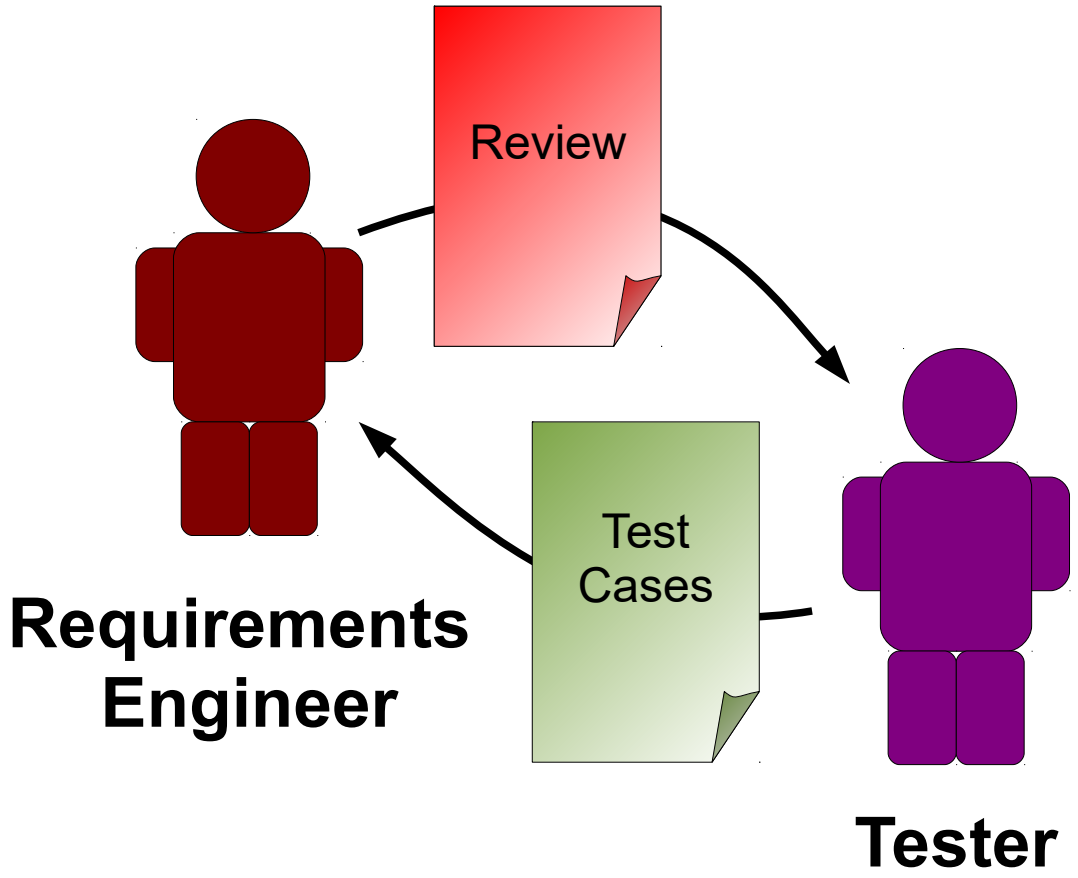
Could be performed as a "Peer Review" by colleagues of the producer of the product

Reviews


Review of requirements, code review
Require-
ments
Code
Developer
Another
Developer
Review
Review
Requirements
Engineer
Tester

Reviews

Review of test cases based on requirements



Reviews

Cost-value ratio

- Reviews cost about 10 to 15 % of development budget.
- Reviews save costs [Bus90] [FLS00] [GG96]:
  - About 14% up to 25% savings in IT projects possible with additional costs of reviews already considered.
  - It's possible to find up to 70% of defects in a document.
  - Reduction of defect costs up to 75%.

Reviews

- Use checklists for reviews
  - Cheap and efficient
  - Challenge: “Right” checklist
    Idea: Common preparation
  - Good to use for milestones / quality gates
  - Tailoring to adapt checklists to project needs

- General hint concerning review findings:
  Do not address issues only, but do propose better statements as well

Reviews

Example for good review finding [Wie99]:
"The HTML Parser shall produce an HTML markup error report which allows quick resolution of errors when used by HTML novices"

- Incomplete
  What goes into the error report?
- Proposal
  "The HTML Parser shall produce an error report that contains the line number and text of any HTML errors found in the parsed file and a description of each error found. If no errors are found, the error report shall not be produced."

Test automation

- Do smart test automation
  - It's possible to save effort
    A good test tool could support a good test process
  - But it's possible to waste money as well
    A good test tool does not improve a bad test process
- Automation of regression tests
  - Required: Stable code, test cases and test data
  - Maintenance of test scripts required
  - Probability to detect new defects is low in general

Test automation

- Areas of test automation
  - Unit tests
  - Continuous integration
    Consider automated regression test after every major integration.
  - Build procedures
  - Test data generation
  - Migration scripts
  - Retests of defect fixes for automated regression tests
- In general: Start simple, e.g. with automation of repetitive tasks

Test know-how

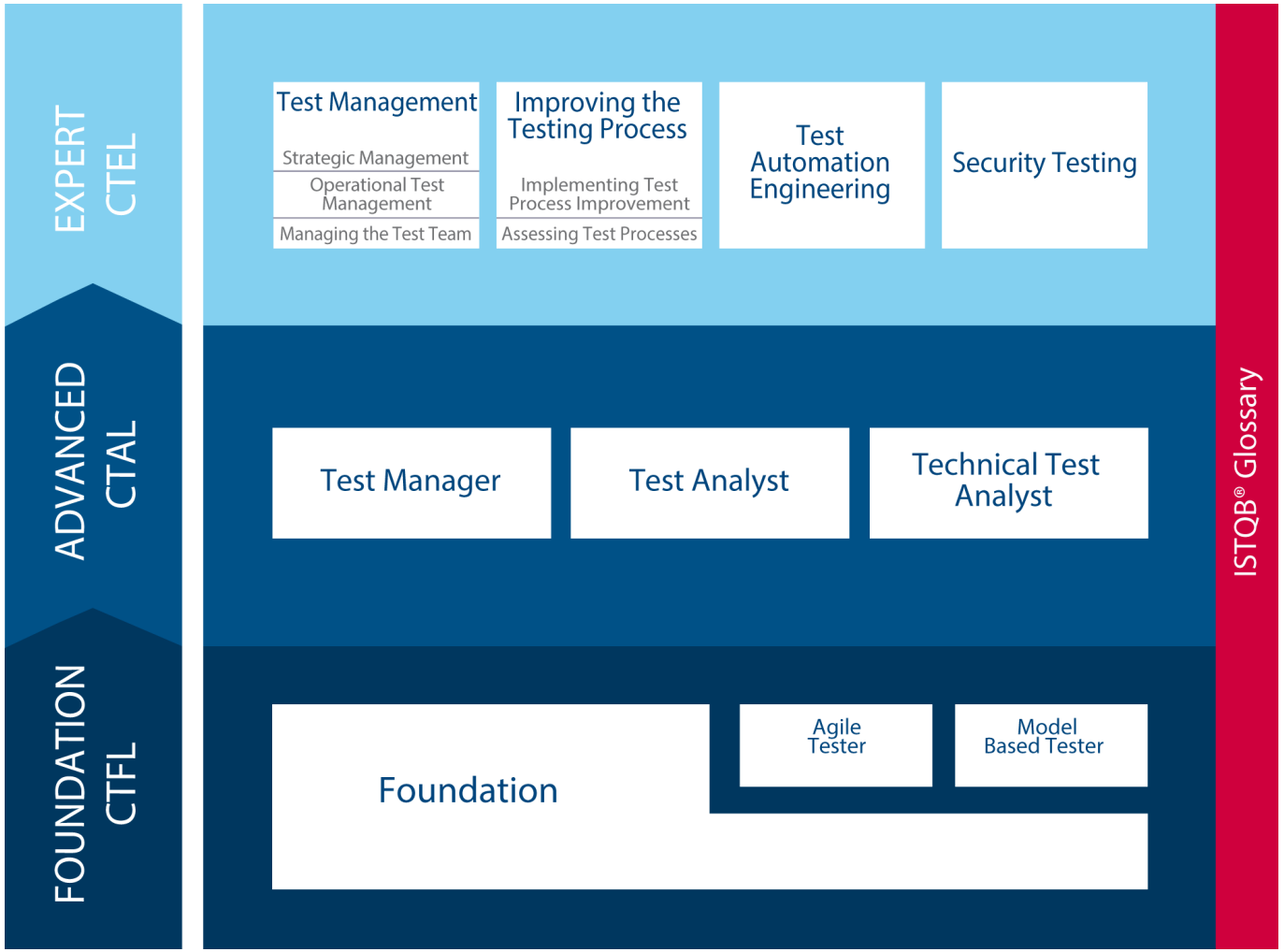
- Get educated – Build up test know-how
  Read testing books, e.g.
  - Lisa Crispin, Janet Gregory: Agile Testing: A Practical Guide for Testers and Agile Teams, Addison-Wesley Signature, 2008
  - Cem Kaner, Jack Falk, Hung Quoc Nguyen: Testing Computer Software, Wiley Computer Publising, 1999
  - Cem Kaner, James Bach, Bret Pettichord: Lessons Learned in Software Testing, Wiley Computer Publising, 2002
  - Glenford J Myers, Tom Badgett, Corey Sandler: The art of Software Testing, Third edition, John Wiley & Sons, Inc., Hoboken, New Jersey, 2012
  - Klaus Pohl, Chris Rupp: Requirements Engineering Fundamentals, 1st edition, Rocky Nook Inc., 2011
  - Andreas Spillner, Tilo Linz, Hans Schaefer: Software Testing Foundations: A Study Guide for the Certified Tester Exam, 3rd Edition, 2011
  - James A. Whittaker, Jason Arbon, Jeff Carollo: How Google Tests Software, Addison-Wesley Professional, 2012

Test know-how

- Join testing communities / follow testing blogs, e.g.
  in Germany ASQF [ASQ16]
  in Thailand “we love bug” [Zyr16]
- Get certified in requirements engineering
  - International Requirements Engineering Board, [IRE16];
    “Certified Professional for Requirements Engineering”
- Get certified in testing
  Achieve certification(s) from ISTQB [IST16]

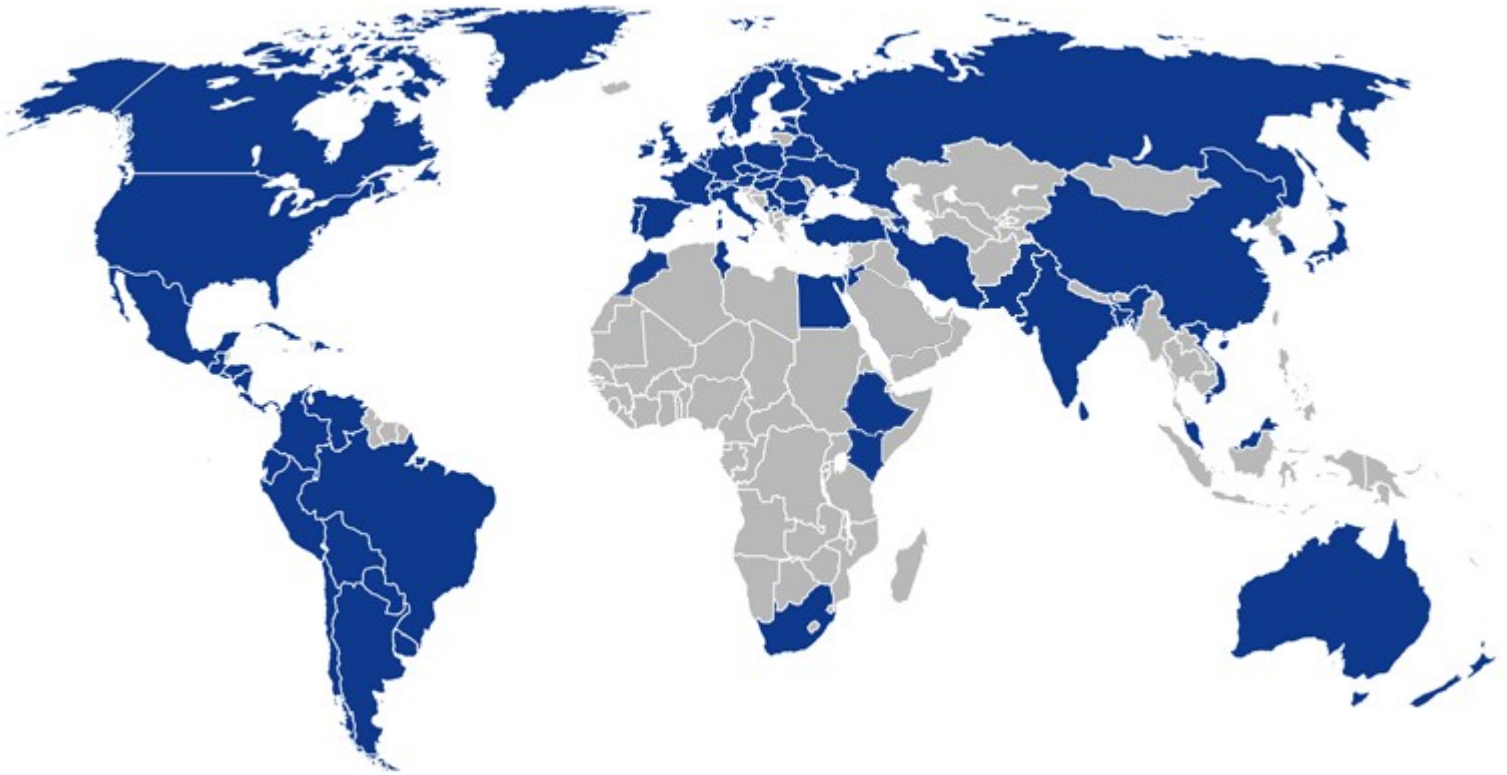
Test know-how

- Certification levels by ISTQB [IST16]



Test know-how

- ISTQB world wide [IST16]



*Countries covered by **Member Boards and Global Exam Providers***

*Countries covered by **Global Exam Providers***

Sources

- [ADA15] Application Developers Alliance, Developer Insights Report, August 2015, http://www.appdevelopersalliance.org/developer-insights-report-2015
- [ASQ16] Homepage Arbeitskreis Software-Qualität und -Fortbildung e.V, 2016, https://www.asqf.de/
- [Bus90] Bush, M.: Software Quality: The use of formal inspections at the Jet Propulsion Laboratory. In: Proc. 12th ICSE, p. 196-199, IEEE 1990
- [Dus03] Elfriede Dustin: Effective Software Testing - 50 Specific Ways to Improve Your Testing, Pearson Education, Inc. 2003
- [FLS00] Frühauf, K.; Ludewig, J,; Sandmayr, H.: Software-Prüfung: eine Fibel. vdf, Verlag der Fachvereine, Zürich, 4. Aufl. 2000
- [GG96] Gilb, T.; Graham, D.: Software Inspections. Addison-Wesley, 1996
- [HW15] Shane Hastie, Stéphane Wojewoda: Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch on Oct 04, 2015, http://www.infoq.com/articles/standish-chaos-2015

Sources

- [IRE16] International Requirements Engineering Board, 2016, https://www.ireb.org/
- [IST15] International Software Testing Qualifications Board (ISTQB): Standard Glossary of Terms Used in Software Testing, Version 3.01, March 26th, 2015, http://www.istqb.org/downloads/glossary.html
- [IST16] Homepage International Software Testing Qualifications Board (ISTQB), 2016, http://www.istqb.org/
- [Jaw13] Ranjeet Jawale: Defect clustering & Pesticide paradox, 2013, http://www.softwaretestingclub.com/profiles/blogs/defect-clustering-pesticide-paradox
- [MSB12] Glenford J Myers, Tom Badgett, Corey Sandler: The art of Software Testing, Third edition, John Wiley & Sons, Inc., Hoboken, New Jersey, 2012
- [PMI13] PMI: The high cost of low performance: The essential role of communication, May 2013, https://www.pmi.org/~/media/PDF/Business-Solutions/The-High-Cost-Low-Performance-The-Essential-Role-of-Communications.ashx

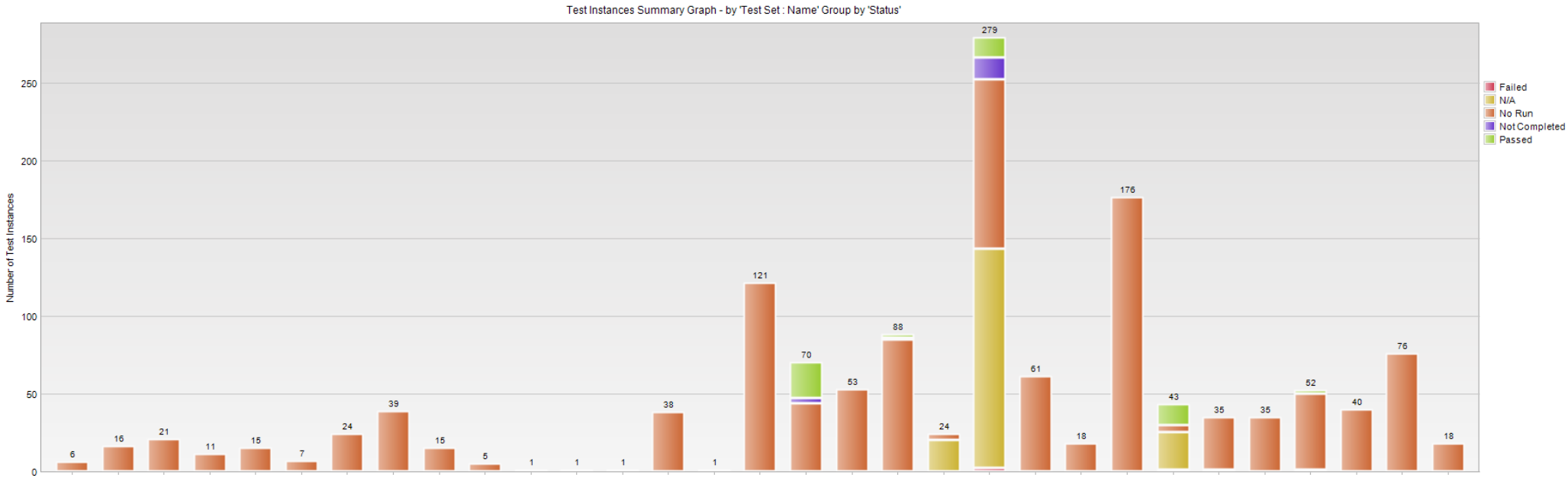
Sources

- [Ric05] Randall W. Rice: STBC The Economics of Testing, 2005, http://www.riceconsulting.com/public_pdf/STBC-WM.pdf
- [SJ06] Harry M. Sneed, Stefan Jungmayr: Produkt- und Prozessmetriken für den Softwaretest, in “Informatik-Spektrum”, Springer Verlag, vol. 29, book 1, Feb. 2006, page 23–38
- [Wie99] Karl E. Wiegers: Writing Quality Requirements, 1999, http://processimpact.com/articles/qualreqs.html
- [Wik16] Wikipedia: Knight Capital Group, 2016, https://en.wikipedia.org/wiki/Knight_Capital_Group
- [Wik16a] Wikipedia: ISO/IEC 9126, 2016, https://en.wikipedia.org/wiki/ISO/IEC_9126
- [zyr16] Zyrakuse: We love bug Thai Software Testing Blog, 2016 http://www.welovebug.com/

Backup

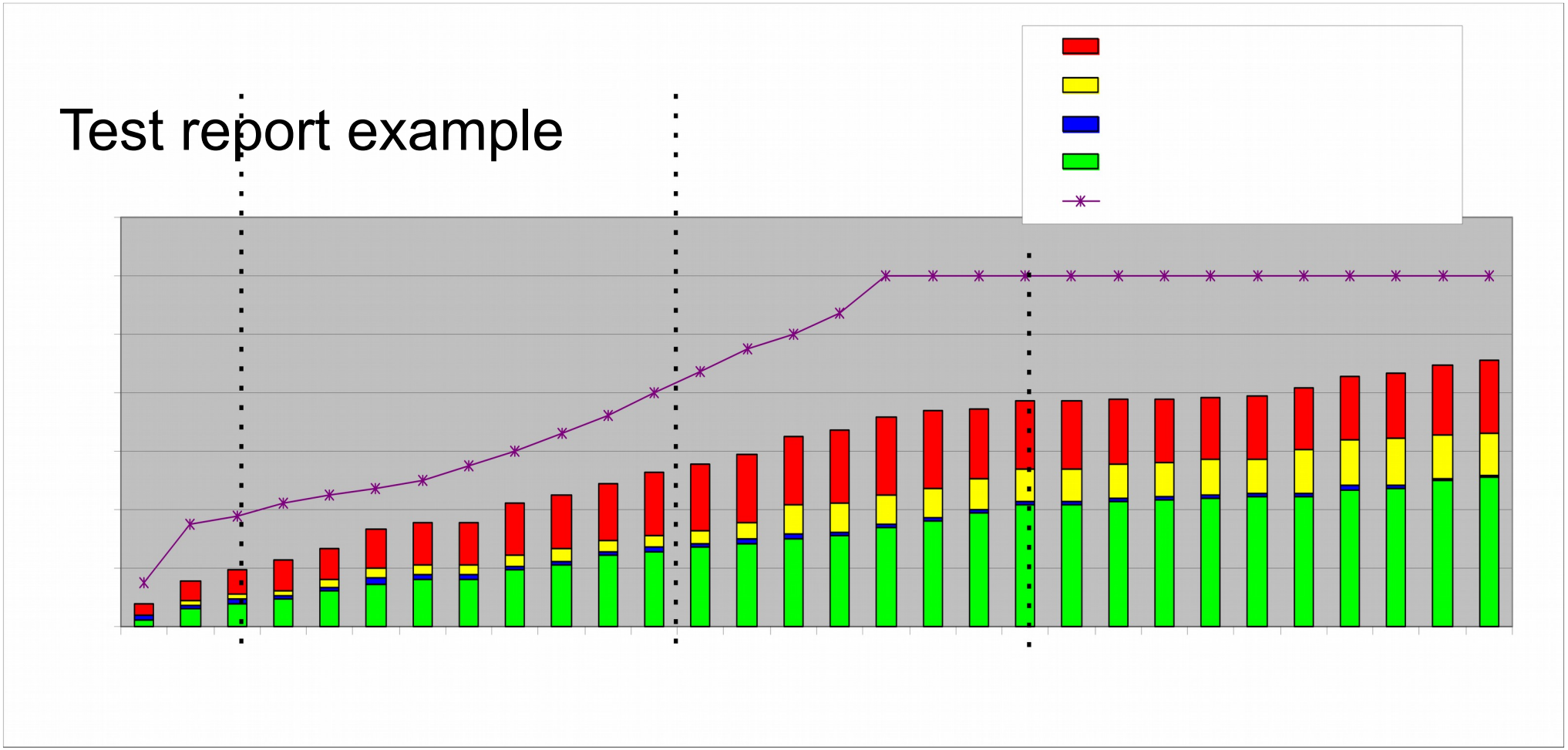
Communication

Test report example



Test execution by area

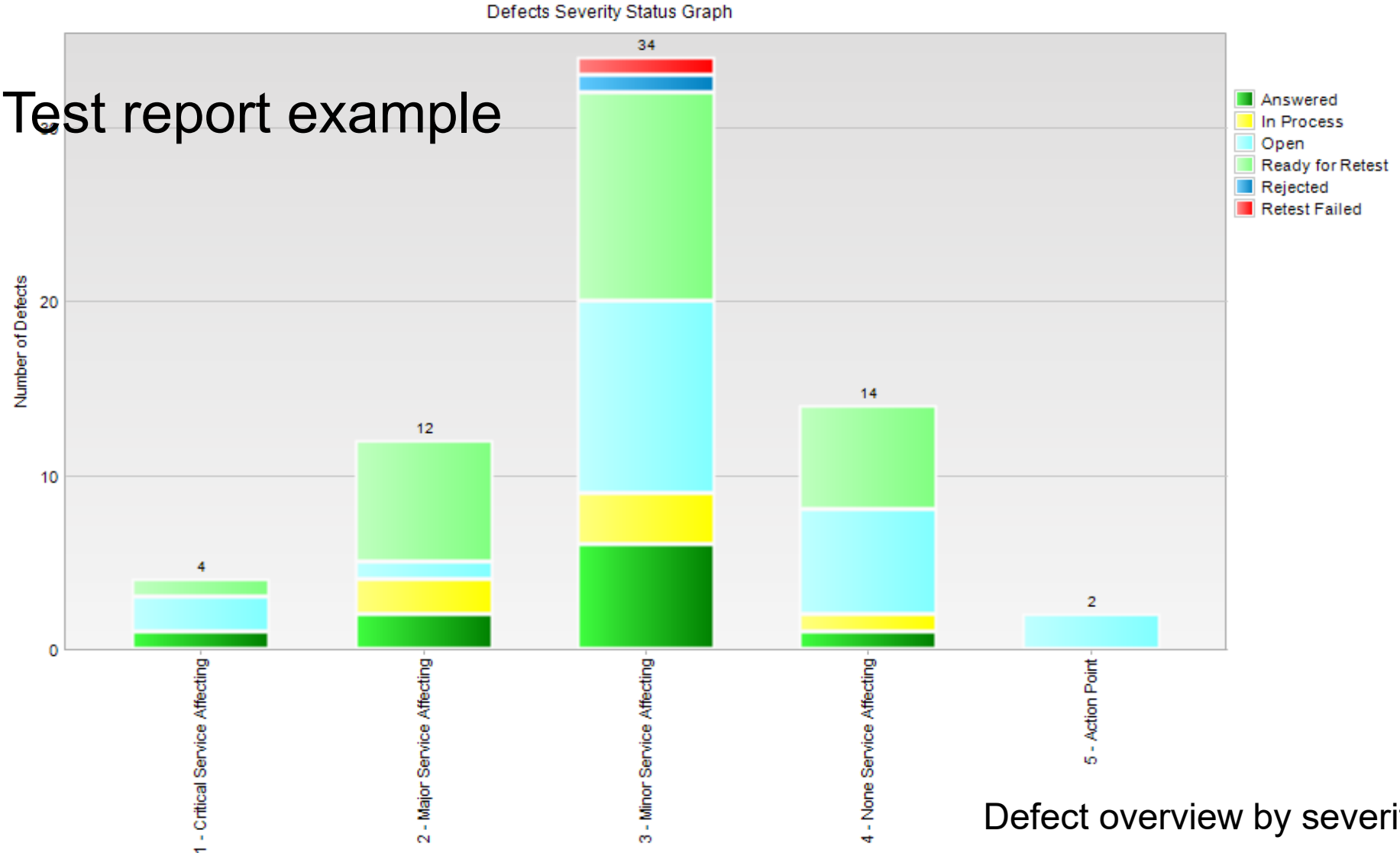
Communication



Test execution progress

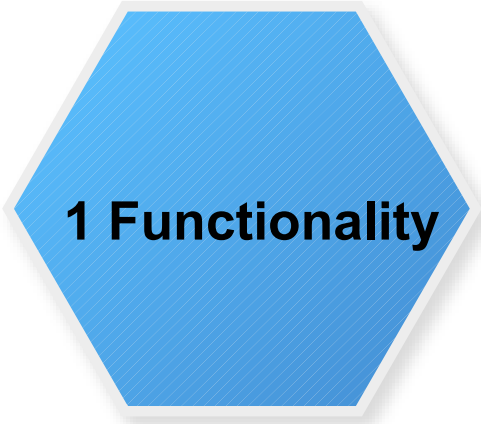
Communication

Test report example



Defect overview by severity

ISO/IEC 9126 Quality Model



1.1.Suitability
Does the software the specified tasks?

1.2.Accuracy
E.g. the needed precision of results

1.3.Interoperability
Cooperates with specified systems

1.4.Compliance
...with conditions / regulations

1.5.Security
No unauthorized access possible

ISO/IEC 9126 Quality Model

2.1.Maturity
concerns frequency of failure of the software.

2.2.Fault Tolerance
Ability to withstand (and recover) from failure like unexpected inputs.

2.3.Recoverability
Ability to recover a failed system including data / network

ISO/IEC 9126 Quality Model

3.1.Learnability
Learning effort for different users

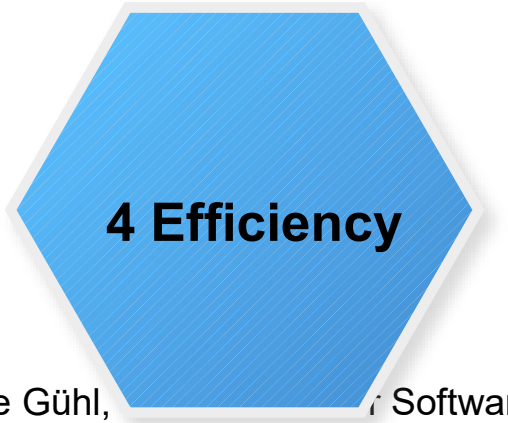
3.2.Understandability
How easy could systems functions be understood?

3.3.Operability:
To keep a system in in a safe and reliable functioning condition

ISO/IEC 9126 Quality Model

4.1.Time Behaviour
Response time, processing time, throughput

4.2.Resource Behaviour:
Usage of RAM, disk space, network, energy



ISO/IEC 9126 Quality Model

5.1.Stability:
Capability to avoid unexpected effects from modifications of the system

5.2.Analyzability:
Ability to identify the root cause of a failure, e.g. with system logs

5.3.Changeability:
Effort to do changes at the system

5.4.Testability:
Effort needed to test a system change.

ISO/IEC 9126 Quality Model

6.1.Installability:
Effort to install a system in a specific environment

6.2.Replaceability:
How easy is it to exchange a given software component within a specified environment (compatibility of data)

6.3.Adaptability:
Ability of the system to change to new specifications or to move to another operating environment

Test Data Management

- Challenge: Dealing with test data
- Measures:
  - Consider different test data status levels
    - Initial test data set
    - Working test data set
  - Especially concerning migration projects and further deployment of given systems:
    Decision which kind of test data to use:
    - Anonymous test data
    - Processed real world data out of production system
    - Copy of real world data out of production system

Interface testing

- Challenge: Integration of software modules or subsystems
- Measures:
  - Continuous integration
  - Fitting integration strategy
  - Test environment
    - Planning and set up at an early stage
    - Early technical system connection tests, if technical integration of systems / system components is working