Software Testing

Lessons Learned

V 1.0

Uwe Gühl

Winter 2013 / 2014

Contents

- Introduction
- Measures to increase IT quality
  - Requirements – non functional requirements
  - Reviews
  - Communication
  - Prioritization
- Testing and Quality
  - Test Report
  - Test Plan
- Want to learn more?
- Sources / More

Introduction

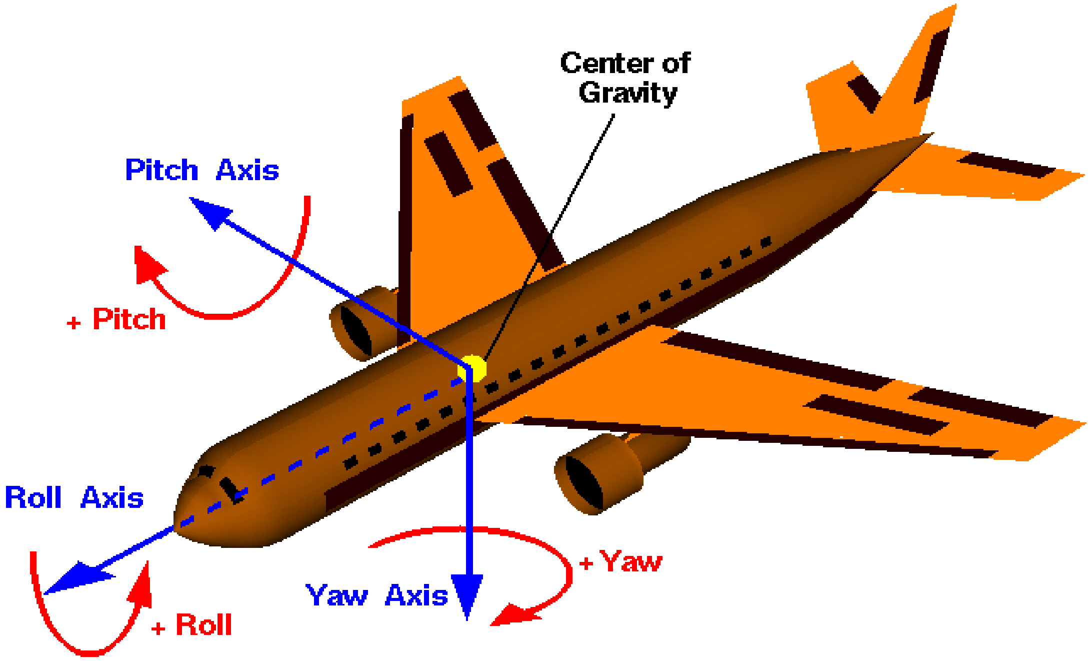
(Fatal) software defects

- 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

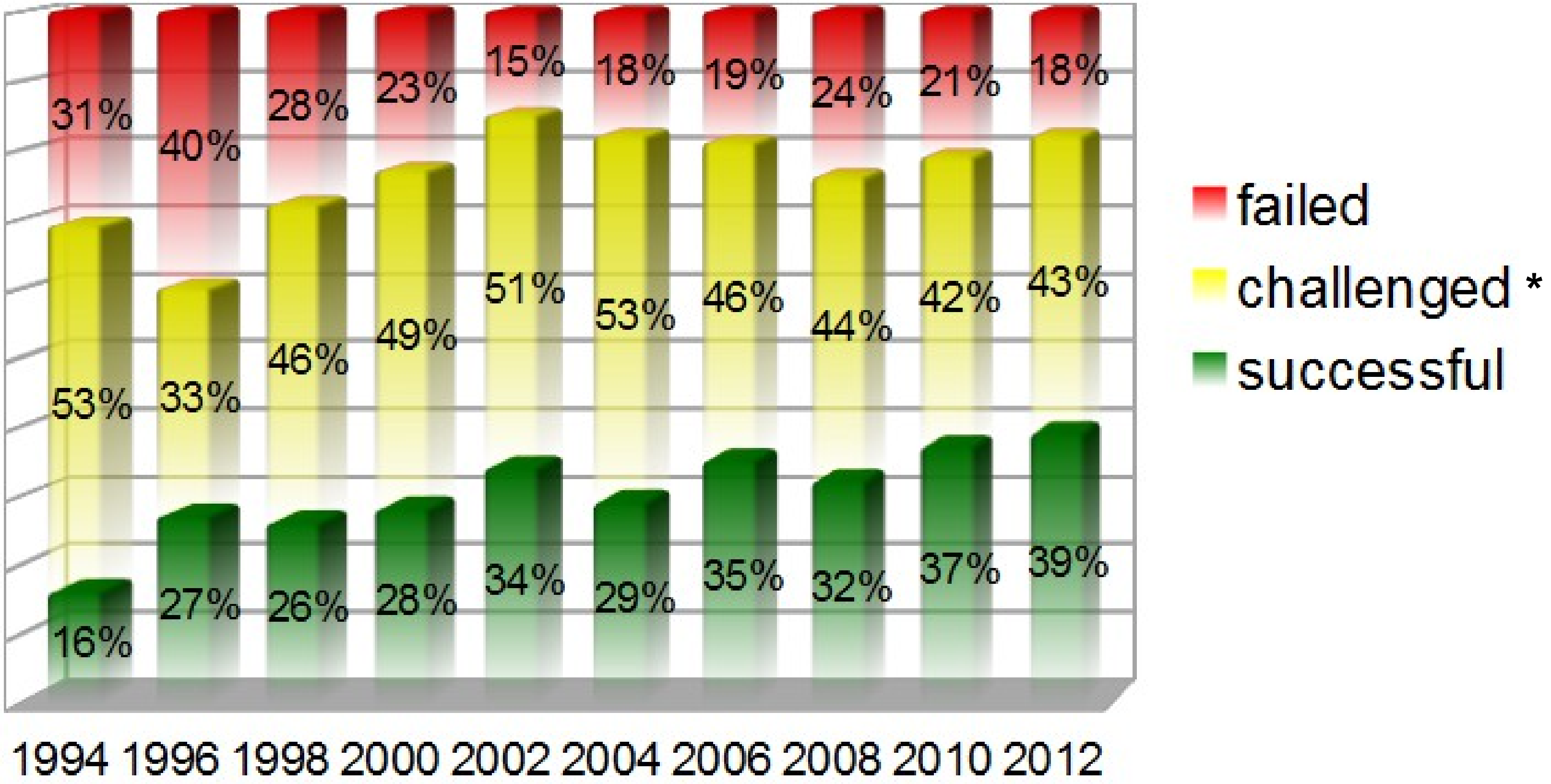
- In 1982 there was a crash of a Lockheed F-117A Night Hawk during takeoff.
- Reason:
  The fly-by-wire system had been hooked up incorrectly
  ("yaw rudder"
  was used instead of "pitch elevator"
  and visa versa)



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

Introduction

Result of an analysis of more than 9000 IT projects
(Standish Group, Chaos Report 2013):



* challenged means overrun in budget and / or time

Introduction

Why do projects fail? [Sta94]

1. Incomplete requirements	13.1%
2. Lack of user involvement	12.4%
3. Lack of resources	10.6%
4. Unrealistic expectations	9.9%
5. Lack of executive support	9.3%
6. Changing requirements and specifications	8.7%
7. Lack of planning	8.1%
8. System no longer needed	7.5%
9. Lack of IT Management	6.2%
10. Technology Illiteracy	4.3%
Other	9.9%

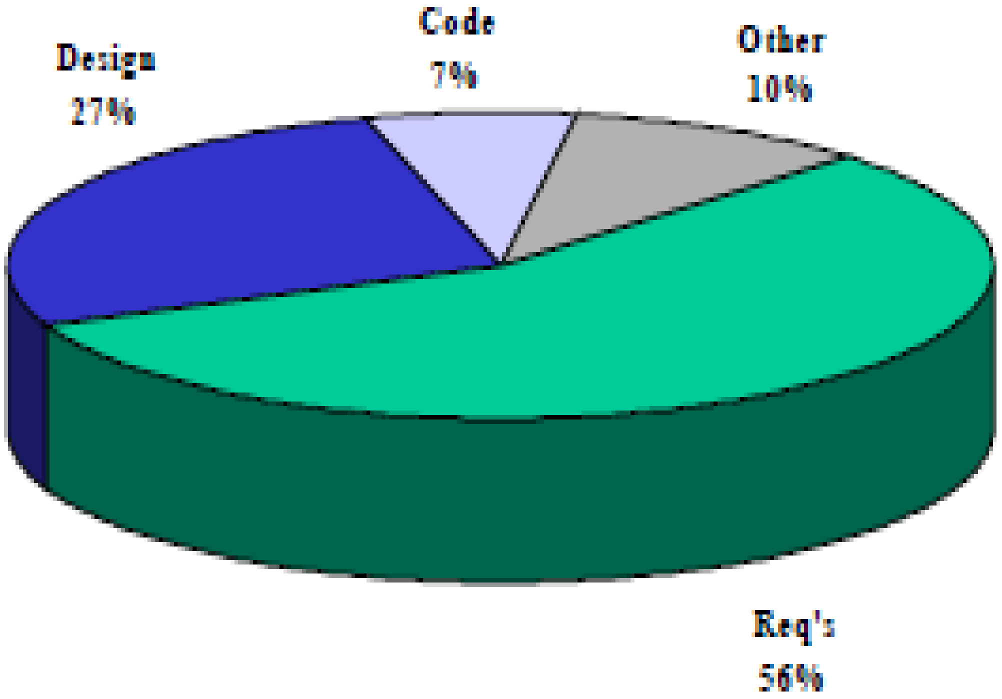
Introduction

Success factors for IT projects: [Sta94]

1. User Involvement	15.9%
2. Executive Support	13.9%
3. Clear Statement of Requirements	13.0%
4. Proper Planning	9.6%
5. Realistic Expectations	8.2%
6. Smaller Project Milestones	7.7%
7. Competent Staff	7.2%
8. Ownership	5.3%
9. Clear Vision & Objectives	2.9%
10. Hard-Working, Focused Staff	2.4%
Other	13.9%

Introduction

What is the source of defects? [Ric05]



⇨ **Requirements play a central role in IT projects**

Introduction

- Prevention, ... not cure
- The earlier a defect is detected, the cheaper is the correction
- More cheaper are defects, which don't occur at all

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]

- Idea: Increasing quality „from scratch“ with corresponding measures: E. g. early reviews of requirements, code, ...

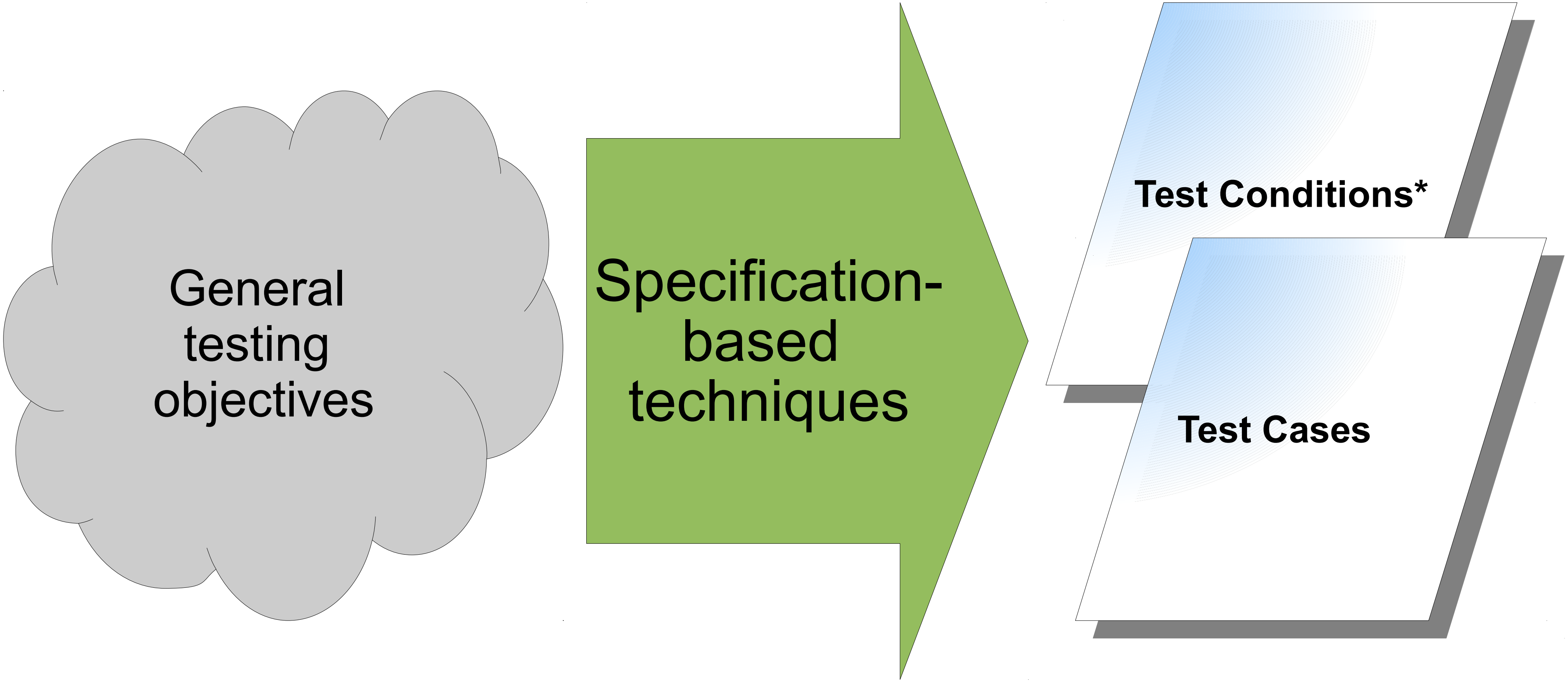
Measures to increase IT quality

So, what to do ?

Requirements

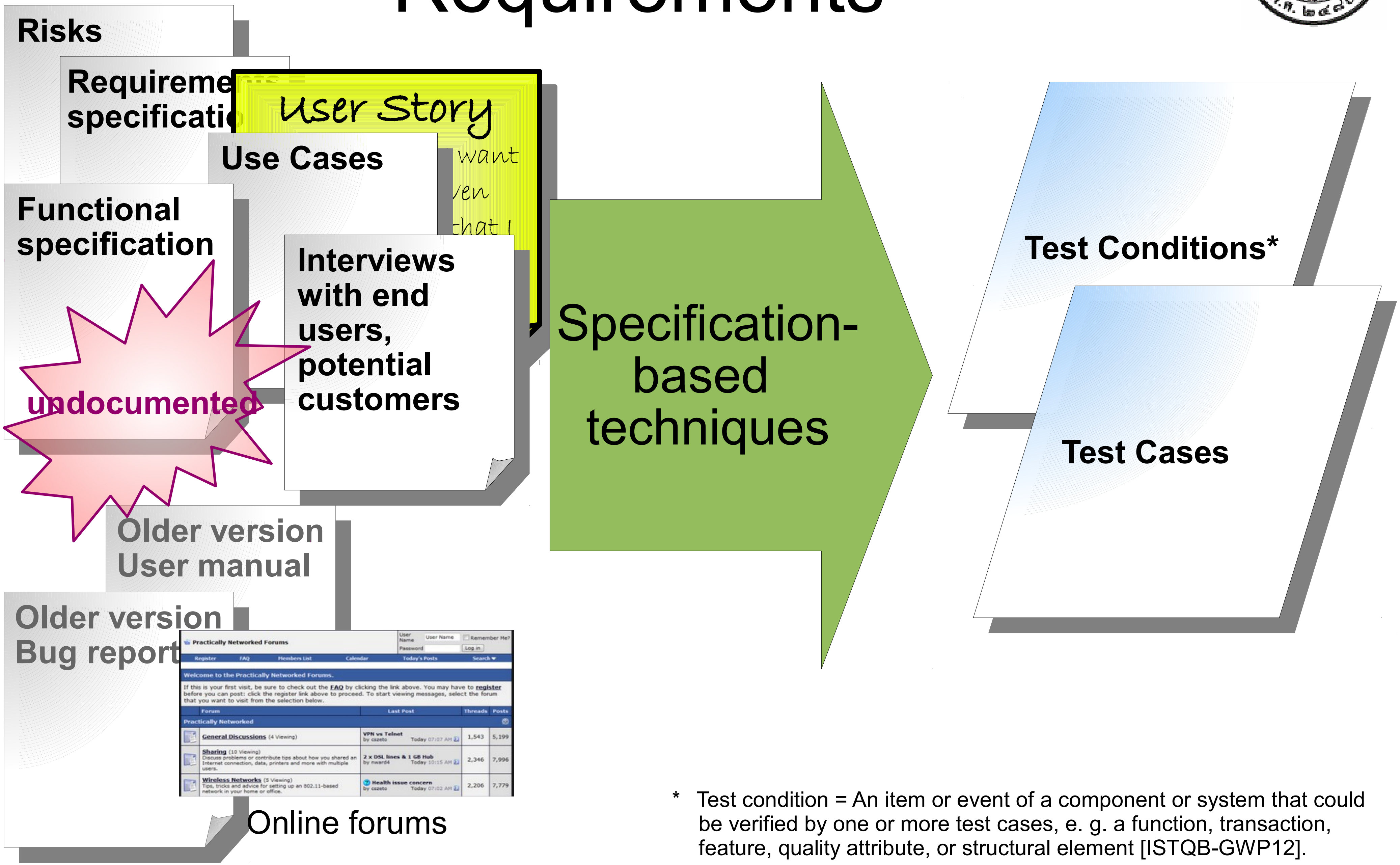
- Requirements and Testing work together
- Requirements are basic for testing
- Testers have to identify the most crucial and most risky requirements
- Gaps in specifications have to be clarified
- Activities to be done, if requirements are missing or not clear, especially non-functional requirements
- Purpose of testing: Focus on high risk areas

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 [ISTQB-GWP12].

Requirements



Practically Networked Forums

User Name: User Name □ Remember Me?
Password: Log in

Register | FAQ | Members List | Calendar | Today's Posts | Search

Welcome to the Practically Networked Forums.

If this is your first visit, be sure to check out the FAQ by clicking the link above. You may have to register before you can post: click the register link above to proceed. To start viewing messages, select the forum that you want to visit from the selection below.

Forum	Last Post	Threads	Posts
Practically Networked			
General Discussions (4 Viewing)	VPN vs Telnet by cszeto Today 07:07 AM	1,543	5,199
Sharing (10 Viewing) Discuss problems or contribute tips about how you shared an Internet connection, data, printers and more with multiple users.	2 x DSL lines & 1 GB Hub by nward4 Today 10:15 AM	2,346	7,996
Wireless Networks (5 Viewing) Tips, tricks and advice for setting up an 802.11-based network in your home or office.	Health issue concern by cszeto Today 07:02 AM	2,206	7,779

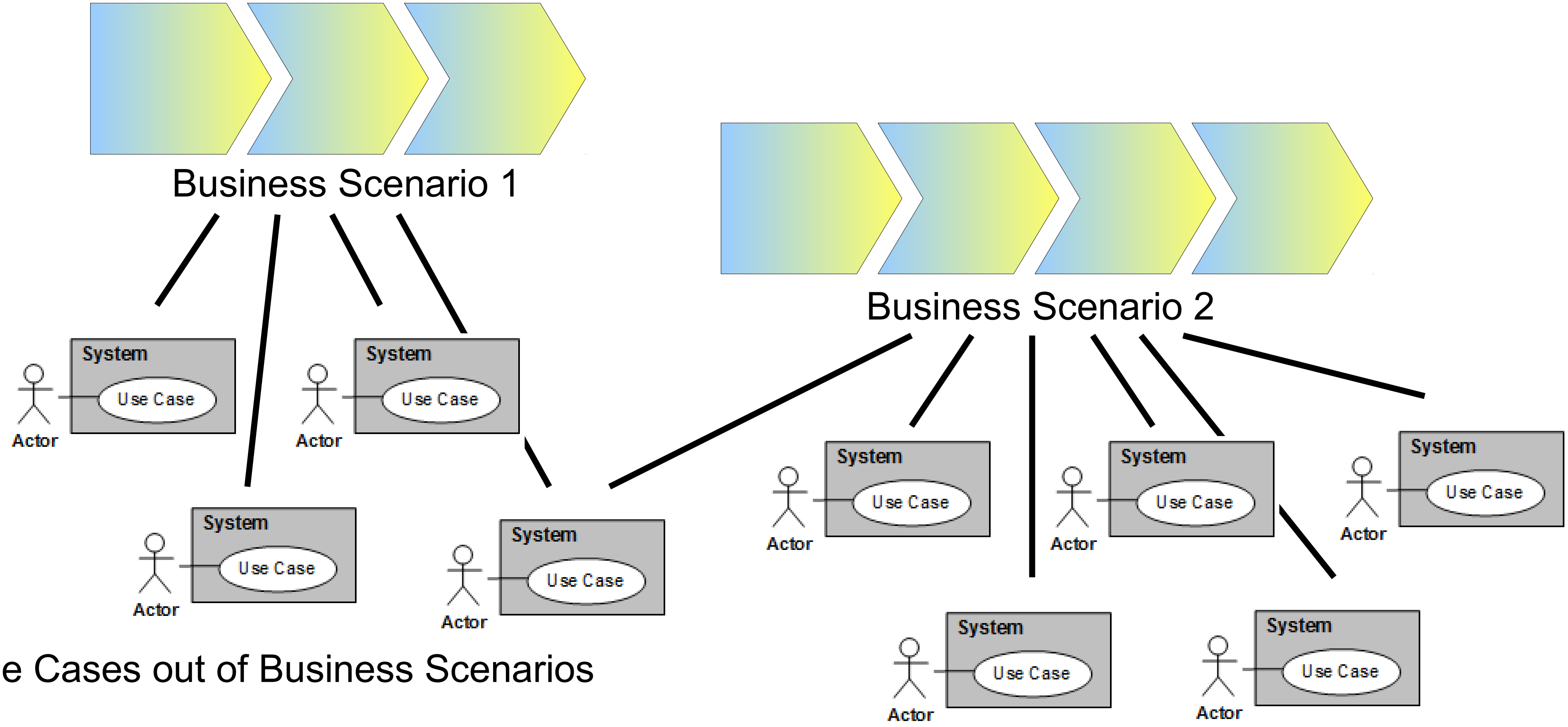
* Test condition = An item or event of a component or system that could be verified by one or more test cases, e. g. a function, transaction, feature, quality attribute, or structural element [ISTQB-GWP12].

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

Requirements

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



Use Cases out of Business Scenarios

Requirements

- Requirements 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 prioritized test cases.

Non-Functional Requirements Motivation

- Unknown Non-Funtional 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!

Non-Functional Requirements ISO/IEC 9126 Quality Model

- ISO/IEC 9126 Software engineering – Product quality [Wik14]
  - 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

Non-Functional Requirements ISO/IEC 9126 Quality Model

1 Functionality

2 Reliability

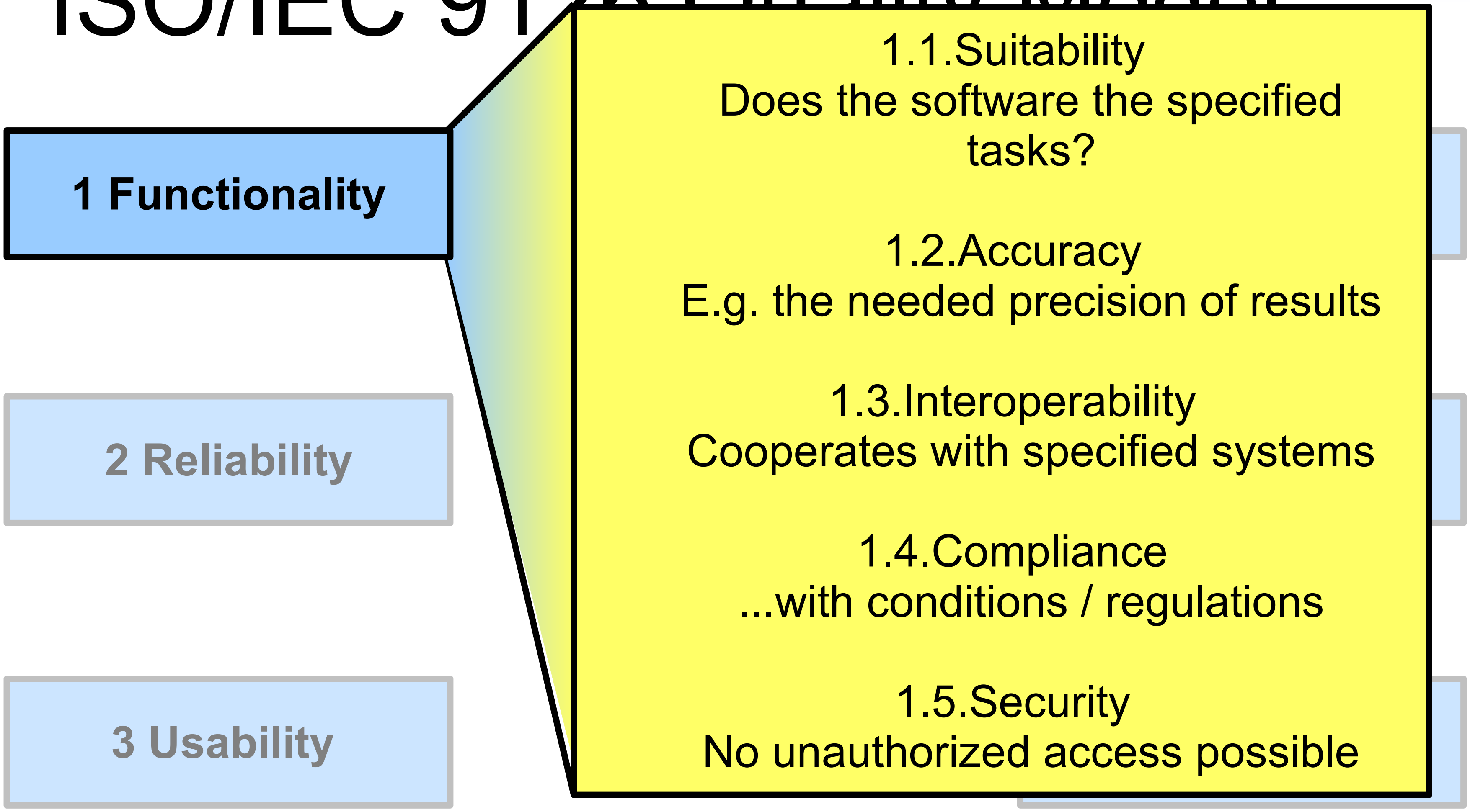
3 Usability

4 Efficiency

5 Maintainability

6 Portability

Non-Functional Requirements ISO/IEC 9126 Quality Model



1 Functionality

2 Reliability

3 Usability

1.1.Suitability
Does the software the specified tasks?

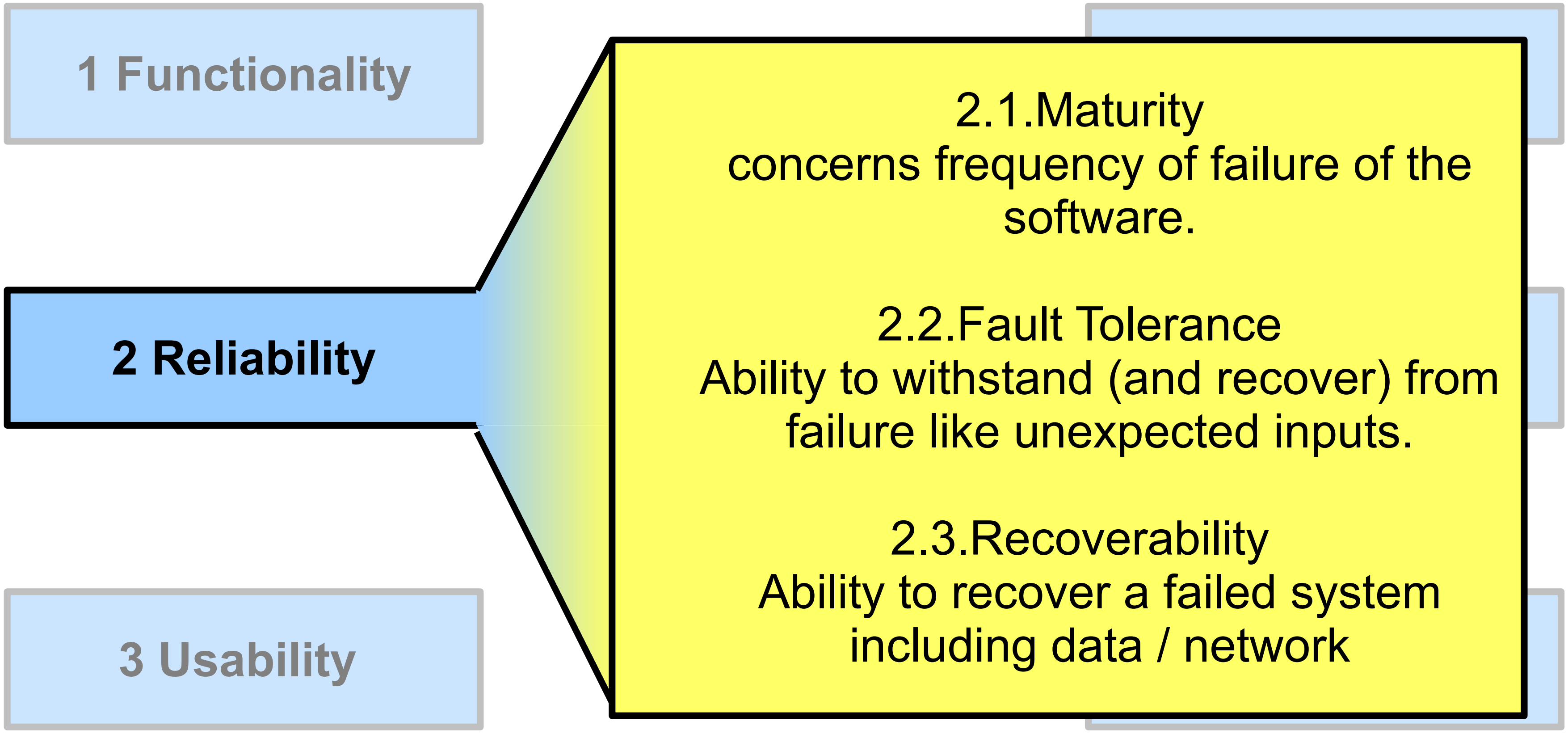
1.2.Accuracy
E.g. the needed precision of results

1.3.Interoperability
Cooperates with specified systems

1.4.Compliance
...with conditions / regulations

1.5.Security
No unauthorized access possible

Non-Functional Requirements ISO/IEC 9126 Quality Model



1 Functionality

2 Reliability

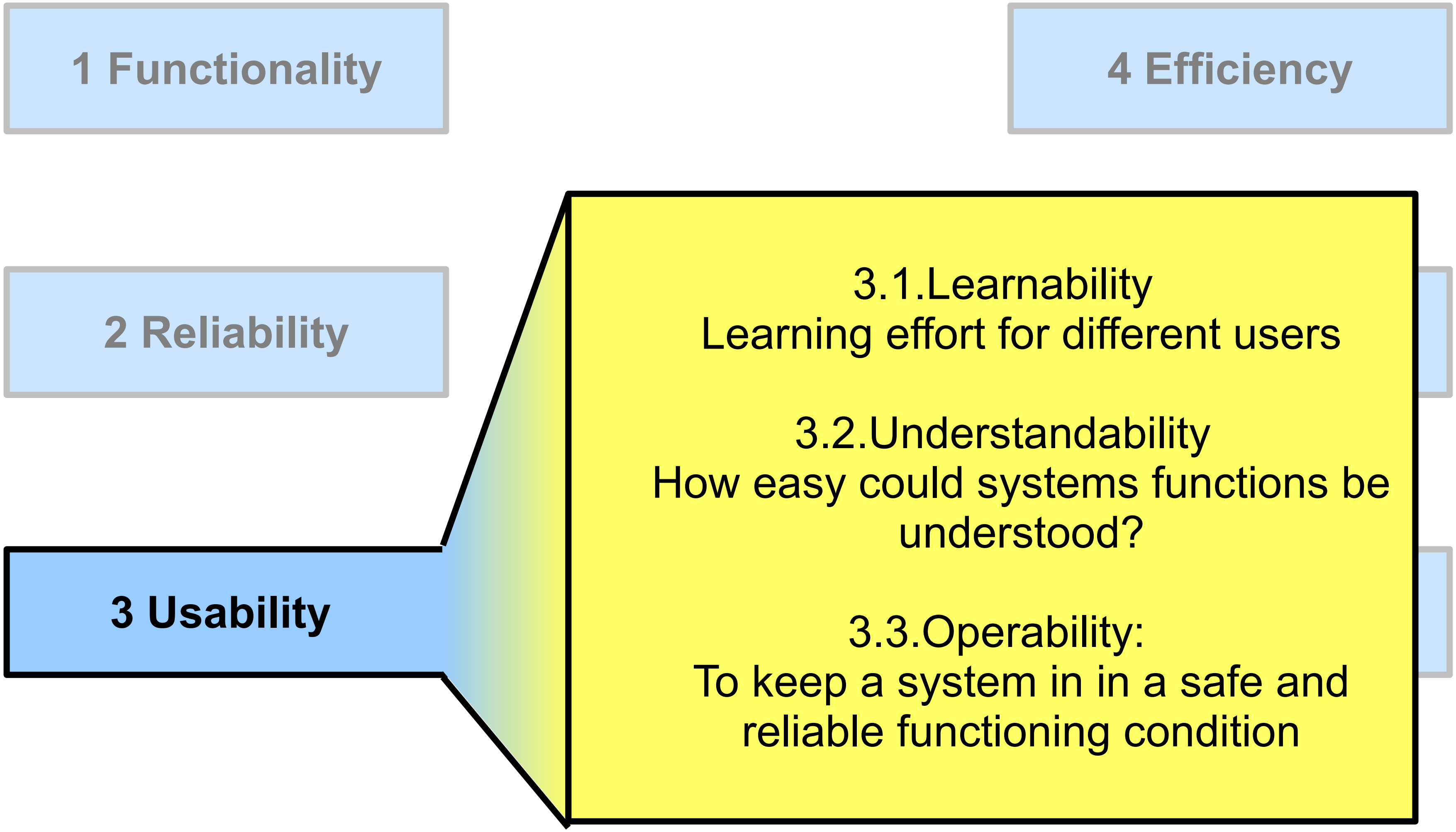
3 Usability

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

Non-Functional Requirements ISO/IEC 9126 Quality Model



1 Functionality

4 Efficiency

2 Reliability

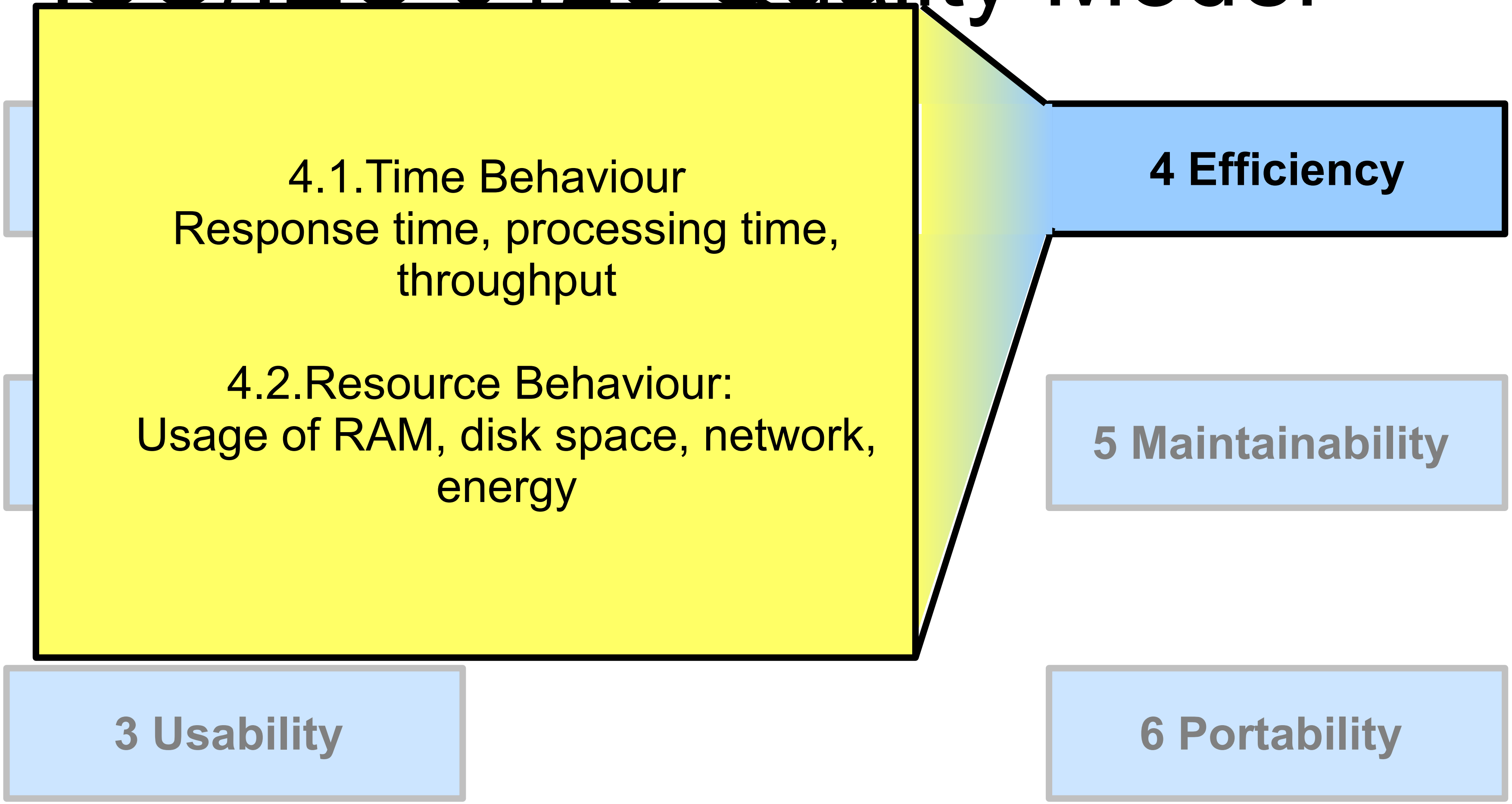
3 Usability

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

Non-Functional Requirements 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

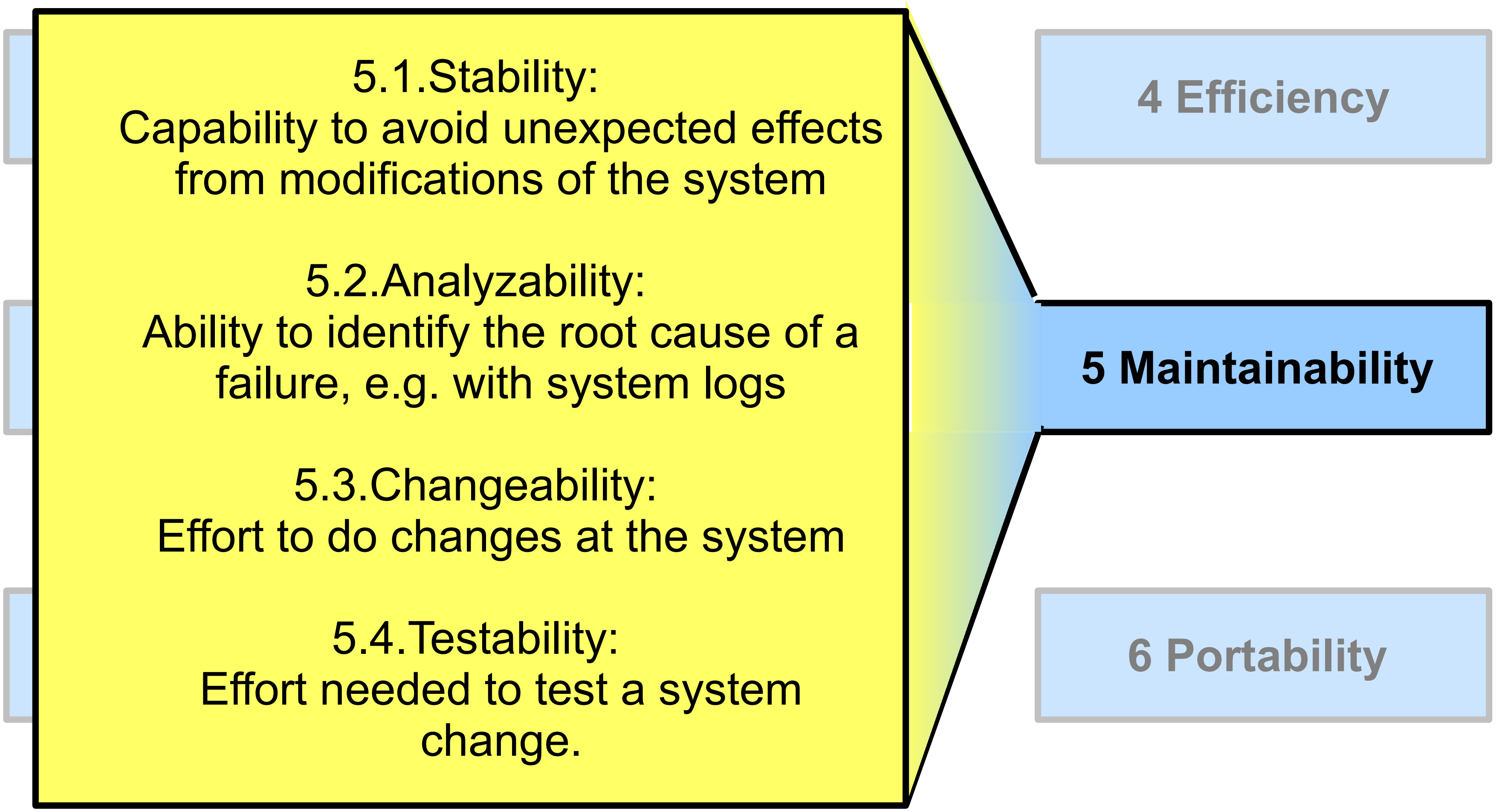
4 Efficiency

5 Maintainability

3 Usability

6 Portability

Non-Functional Requirements 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

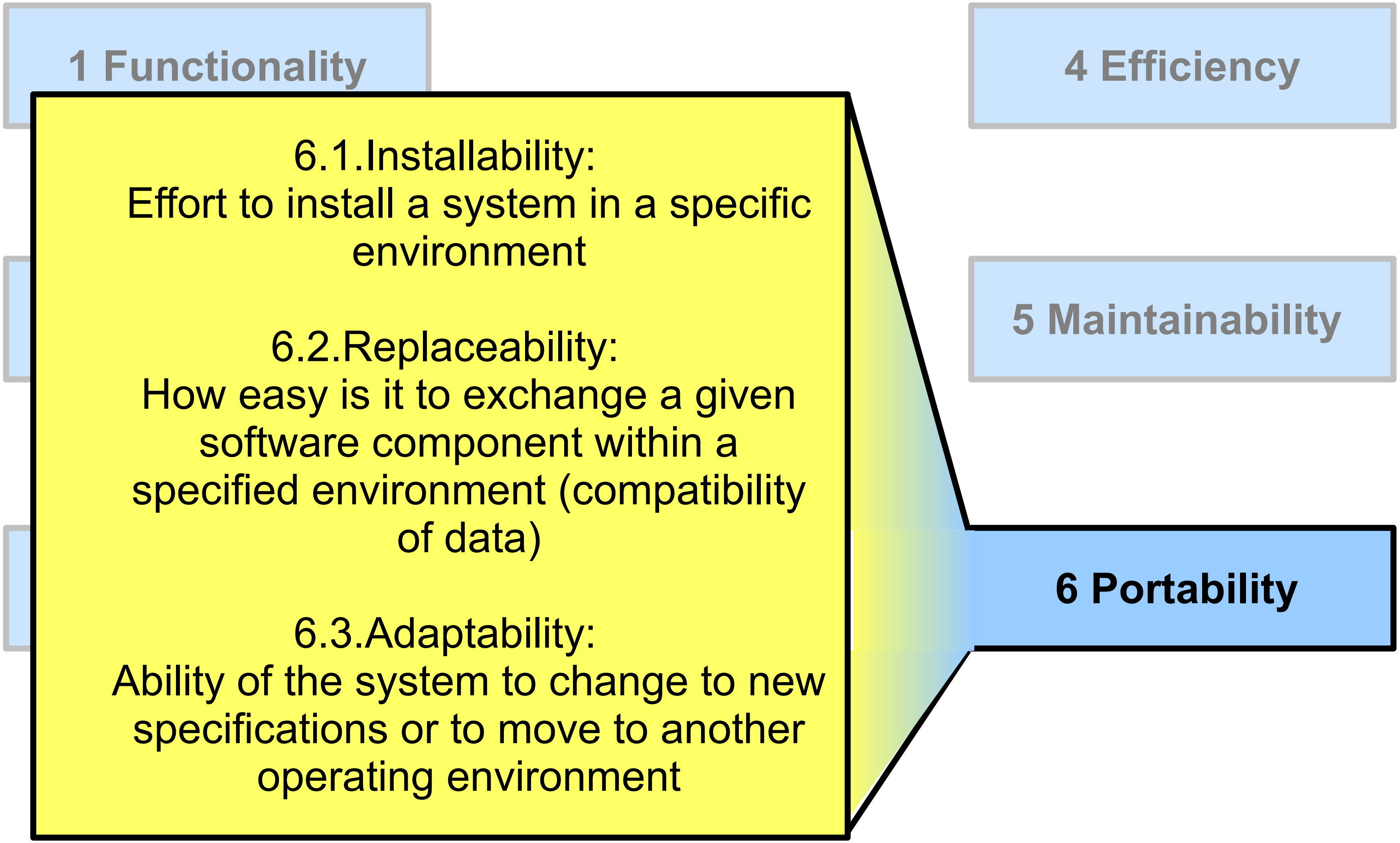
5.4.Testability:
Effort needed to test a system change.

4 Efficiency

5 Maintainability

6 Portability

Non-Functional Requirements ISO/IEC 9126 Quality Model



1 Functionality

4 Efficiency

5 Maintainability

6 Portability

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

Non-Functional Requirements Proceeding

Proposal: Performing a work shop

1. Presentation of current status of software project (status of requirements, general set-up, system interfaces, architecture)
2. **Start:** Presentation and explanation of non-functional requirements
3. **Prio:** Prioritization of characteristic / sub-characteristic criteria
4. **Tasks:** Definition of concrete quality / acceptance criteria and next activities

Non-Functional Requirements Proceeding – Example (Start)

High priority	Medium priority	Low priority
		1.2. Accuracy
		4.1. Time Behaviour
		5.4. Testability
		6.3. Replaceability

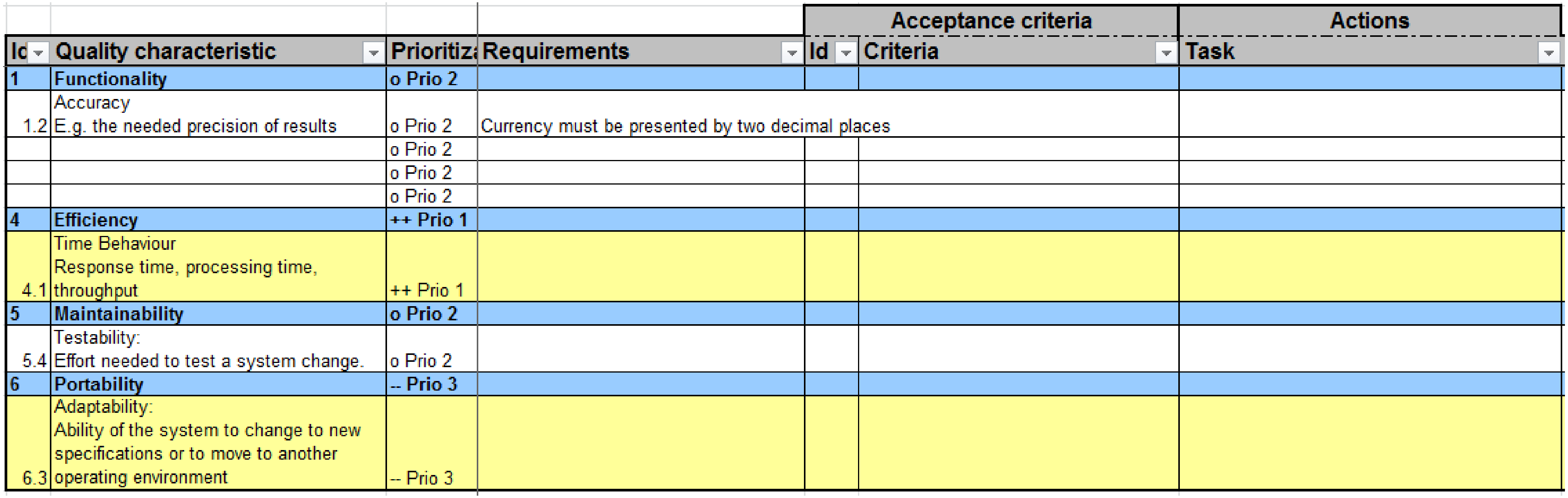
Non-Functional Requirements Proceeding – Example (Prio)

High priority	Medium priority	Low priority
4.1. Time Behaviour 🔵🔵🔵🔴🔴	**1.2. Accuracy** 🔵🔵🔵	**6.3. Replaceability**
	5.4. Testability 🔴🔴🔴	

Prioritization done by workshop participants, IT (red dots), Business (blue dots)

Non-Functional Requirements Proceeding – Example (Tasks)

- Collection of requirements, acceptance criteria, tasks to be executed, etc.



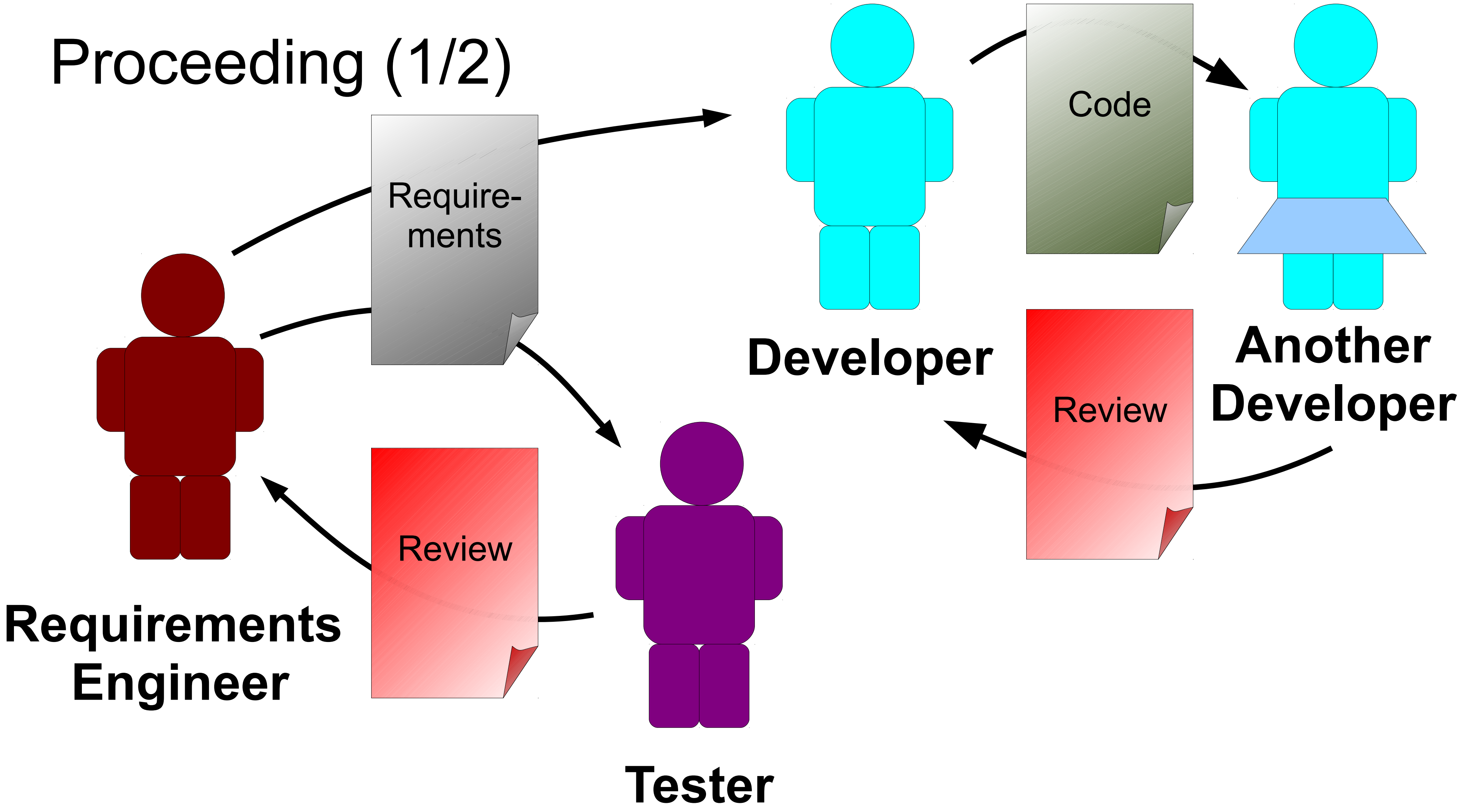
				Acceptance criteria		Actions
Ic	Quality characteristic	Prioritiz	Requirements	Id	Criteria	Task
1	**Functionality**	**o Prio 2**				
1.2	Accuracy E.g. the needed precision of results	o Prio 2	Currency must be presented by two decimal places			
		o Prio 2				
		o Prio 2				
		o Prio 2				
4	**Efficiency**	**++ Prio 1**				
4.1	Time Behaviour Response time, processing time, throughput	++ Prio 1				
5	**Maintainability**	**o Prio 2**				
5.4	Testability: Effort needed to test a system change.	o Prio 2				
6	**Portability**	**-- Prio 3**				
6.3	Adaptability: Ability of the system to change to new specifications or to move to another operating environment	-- Prio 3				

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

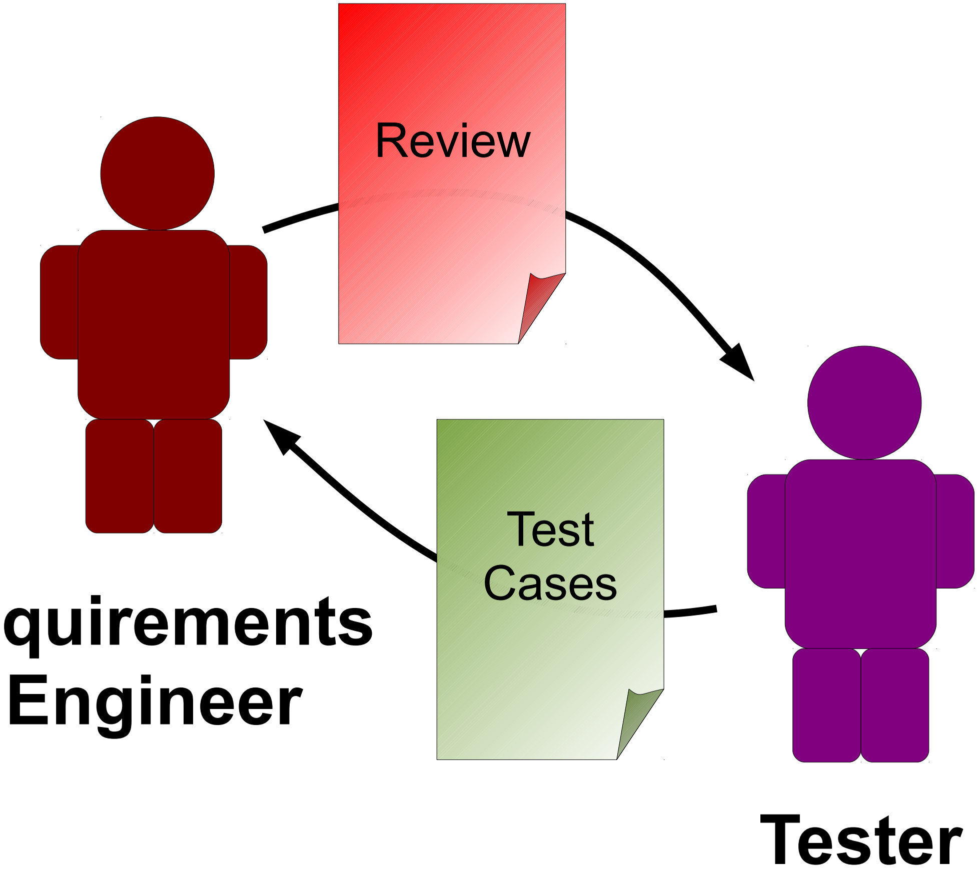
Could be performed as a "Peer Review" by colleagues of the producer of the product

Reviews


Proceeding (1/2)
Require-
ments
Code
Developer
Another
Developer
Review
Review
Requirements
Engineer
Tester

Reviews

Proceeding (2/2)



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 (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

- „Peer reviews“ – capable experts review the work
  ***Use:** will detect about 31 % up to 93 % of all defects, average: 60 %*
- “Perspective review” – evaluators use the work for own tasks (For example specification: Generation of test cases, or a manual out of it)
  ***Use:** 35 % more defects are detected compared to non-purposeful reviews*

Reviews

Be active in reviewing requirements.

- Problems?
  Ask questions

- Proposals!
  Propose better statements

Reviews

Example: "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."

[Wie99]

Communication

- Test planning
  - To identify scope:
    Customer, project sponsor, project manager
  - To identify risks:
    Test team, developer, sales, architect, end user, people related to similar projects, investigation
- Test reporting:
  … to all project stakeholders
- Enforce Communication
  Requirements Engineer ⇔ Developer ⇔ Tester

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:

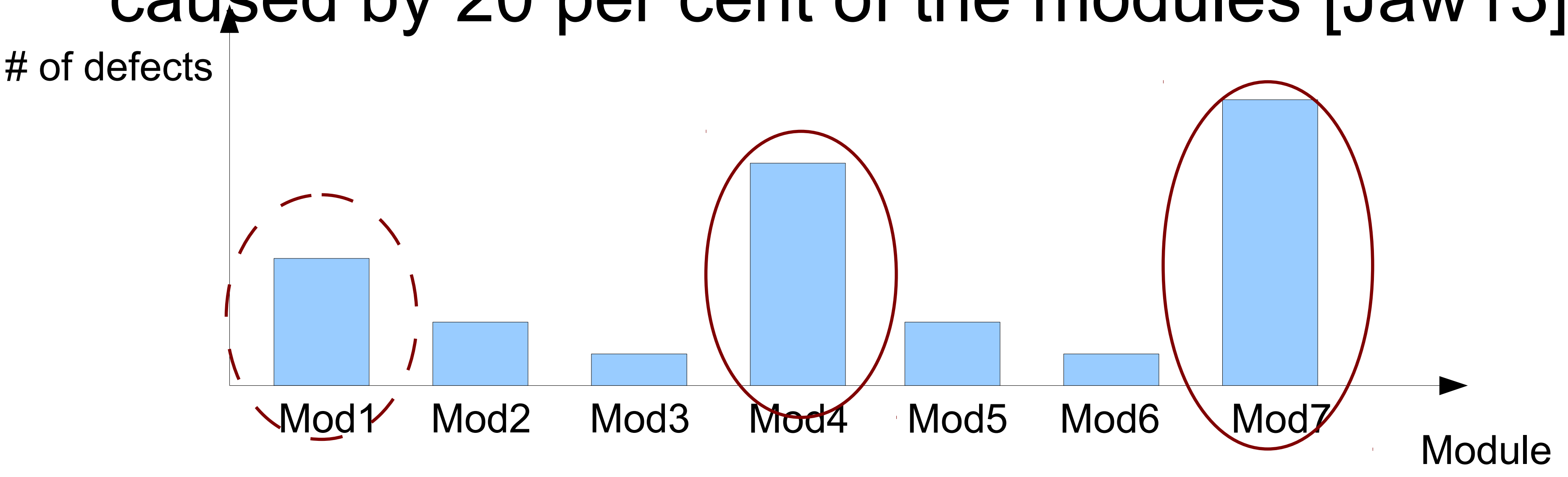
- $2^{16} * 2^{16} * 2^{16} = 2^{48}$ combinations
  = 281.474.976.710.656 combinations
- Duration: About 90 years

Prioritization

- So: You can't test everything
- What to do?
  - Risk based testing
    ==> Identify risks – remember requirements
  - 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

- A small number of modules usually contains most of the defects.
- Defect clustering is based on the **Pareto principle** – the 80-20 rule. Approximately 80 per cent of the problems are caused by 20 per cent of the modules [Jaw13].



… more

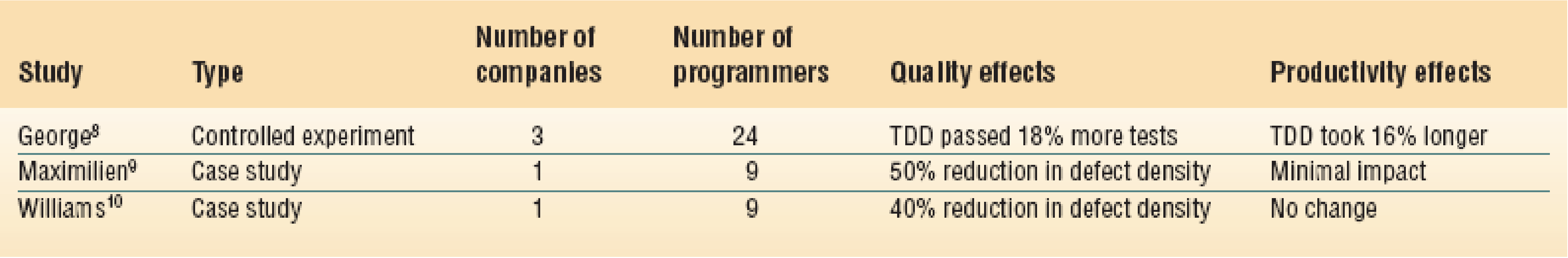
Pair programming

Quality is rising when doing pair programming [TDD05]

TDD research studies in industry

> *„… showed that programmers using TDD produced code that passed 18 percent to 50 percent more external test cases than code produced by corresponding control groups“*

with minimal impact to productivity



Study	Type	Number of companies	Number of programmers	Quality effects	Productivity effects
George[8]	Controlled experiment	3	24	TDD passed 18% more tests	TDD took 16% longer
Maximilien[9]	Case study	1	9	50% reduction in defect density	Minimal impact
Williams[10]	Case study	1	9	40% reduction in defect density	No change

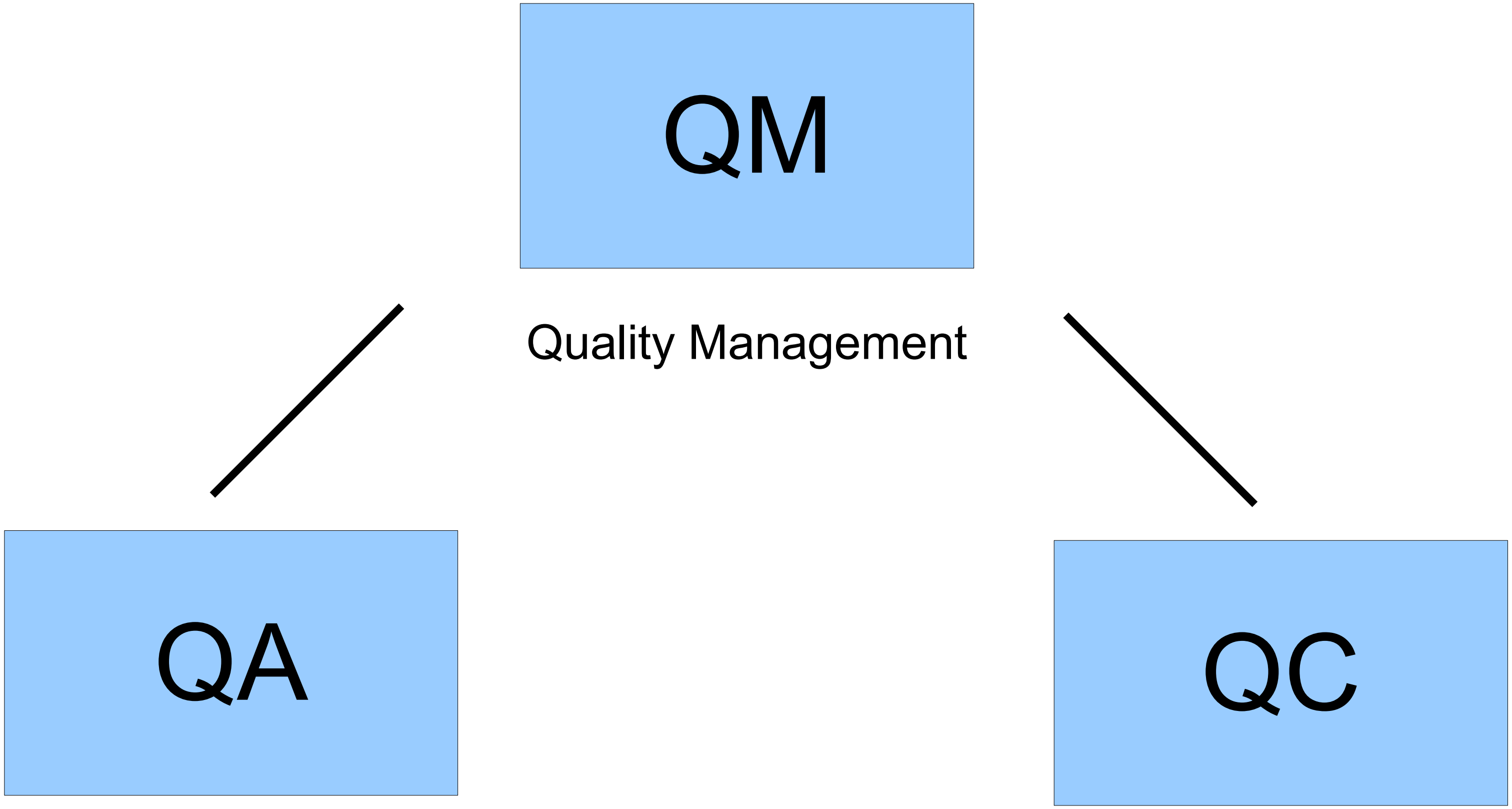
… more

- Continuous Integration
  … to detect integration issues as soon as possible.
  Consider automated regression test after every major integration.
- 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

… more

- Checklists
  - Cheap and efficient
  - Challenge: “Right” checklist
    Idea: Common preparation
  - Good to use for milestones / quality gates

Testing and Quality



QM

Quality Management

QA

QC

Quality Assurance

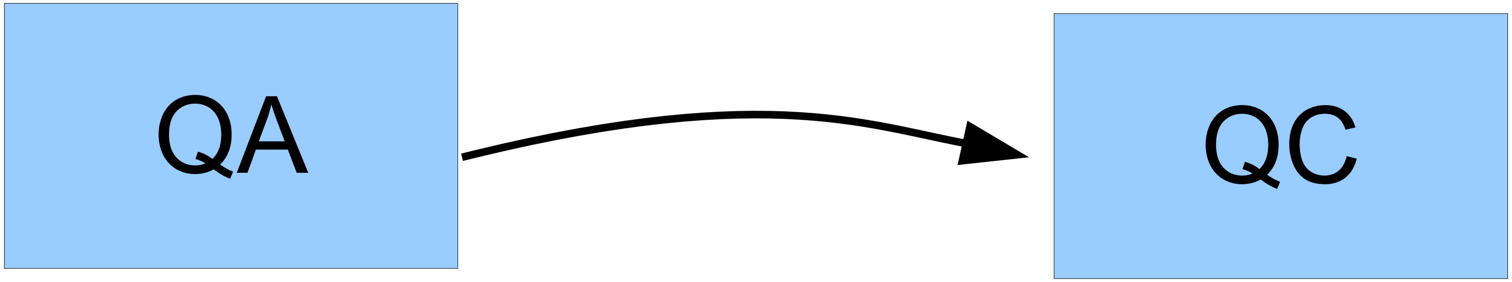
Are we building the product right?
Prevention of faults
by inspecting and testing the process

Quality Control

Are we building the right product?
Detection of faults
by inspecting and testing the product

Testing and Quality

- Relationship QA – QC
  As QA inspects the processes, it investigates in test processes as well, test process improvements e. g. with TPI [Sog14] or TMMI [TMMI14]



Quality Assurance

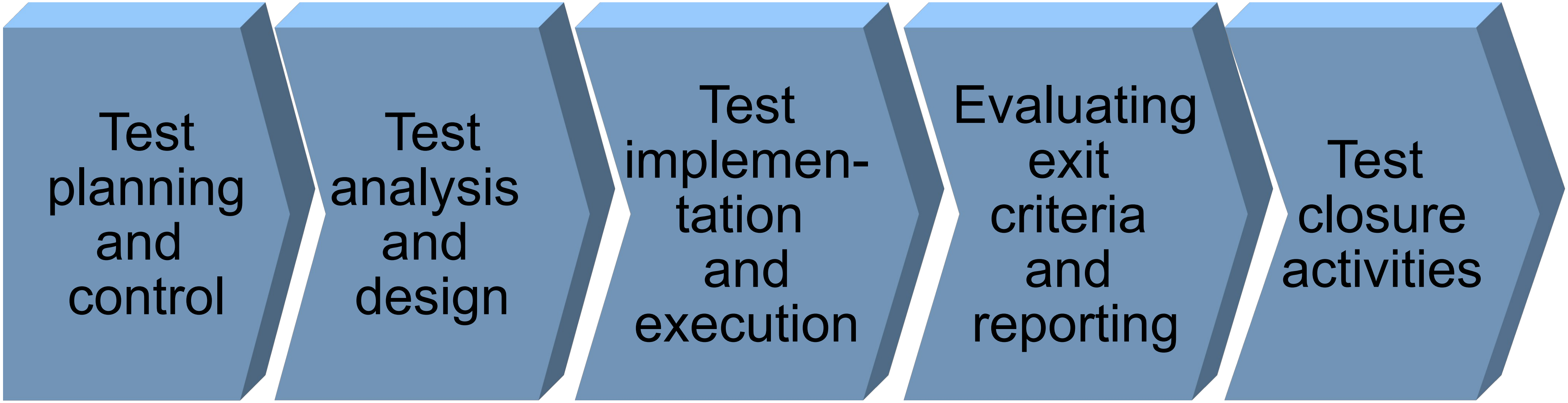
Quality Control

Examples for test processes and test work products

- Defect Management Process
- Test Case Creation Process
- Test Cases
- Test Reports

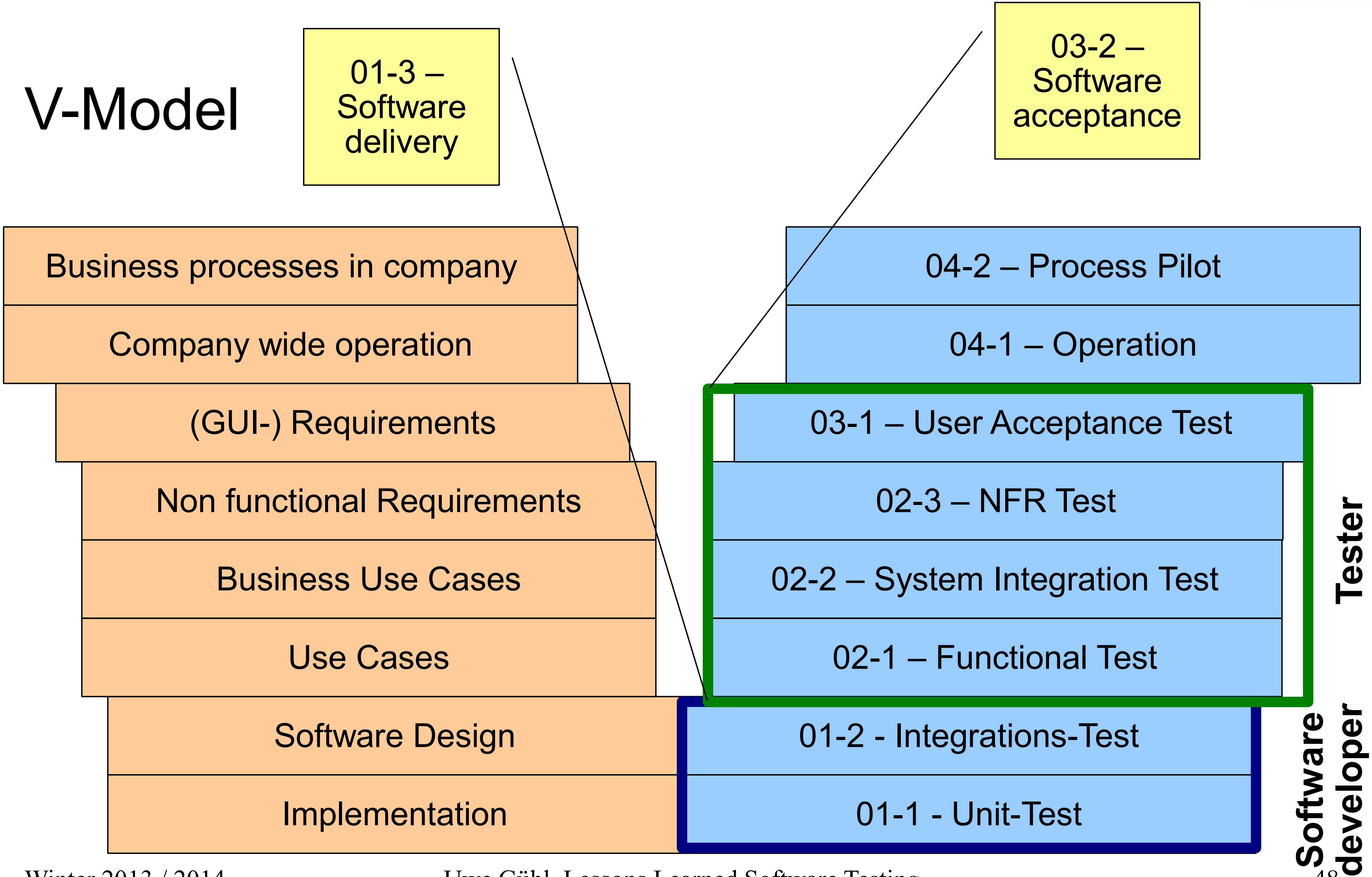
Testing and Quality

Fundamental Test Process


Test planning and control
Test analysis and design
Test implementation and execution
Evaluating exit criteria and reporting
Test closure activities

Testing and Quality

V-Model



01-3 – Software delivery

03-2 – Software acceptance

Business processes in company

Company wide operation

(GUI-) Requirements

Non functional Requirements

Business Use Cases

Use Cases

Software Design

Implementation

04-2 – Process Pilot

04-1 – Operation

03-1 – User Acceptance Test

02-3 – NFR Test

02-2 – System Integration Test

02-1 – Functional Test

01-2 - Integrations-Test

01-1 - Unit-Test

Tester

Software developer

Testing and Quality
Test Report

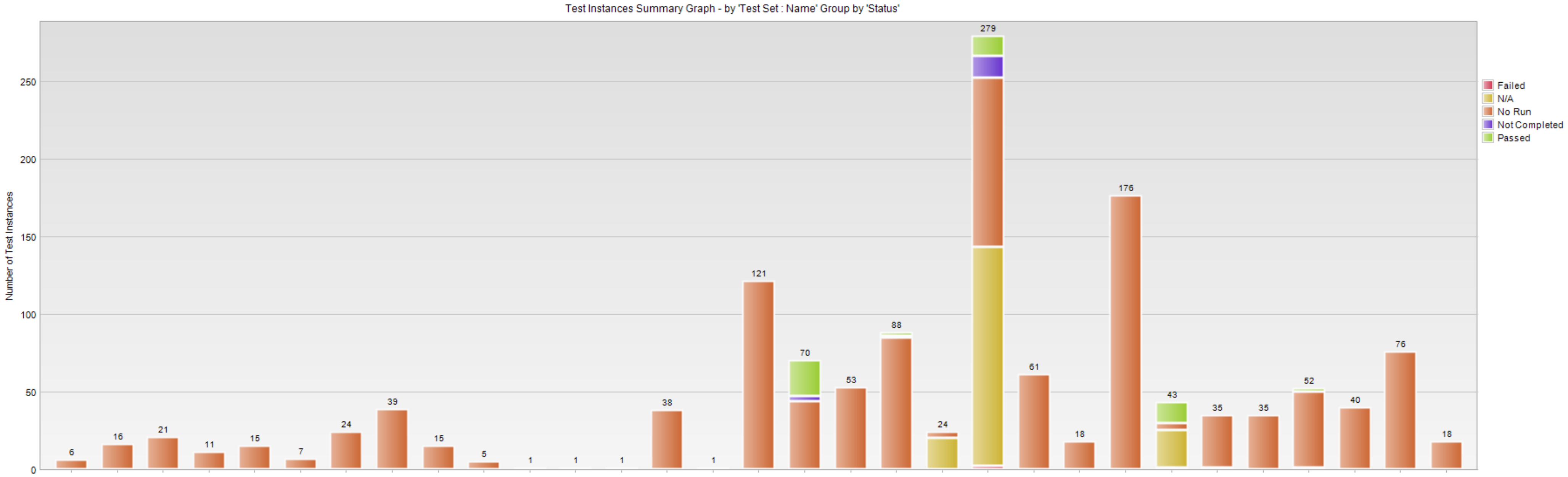
- The test report
  - is the working result of the test team
  - is the business card of the test team

- Contents is based on test plan: Compare what has been planned and what has been achieved.
  - Test coverage
  - Defect situation
  - Quality statements

Testing and Quality
Test Report

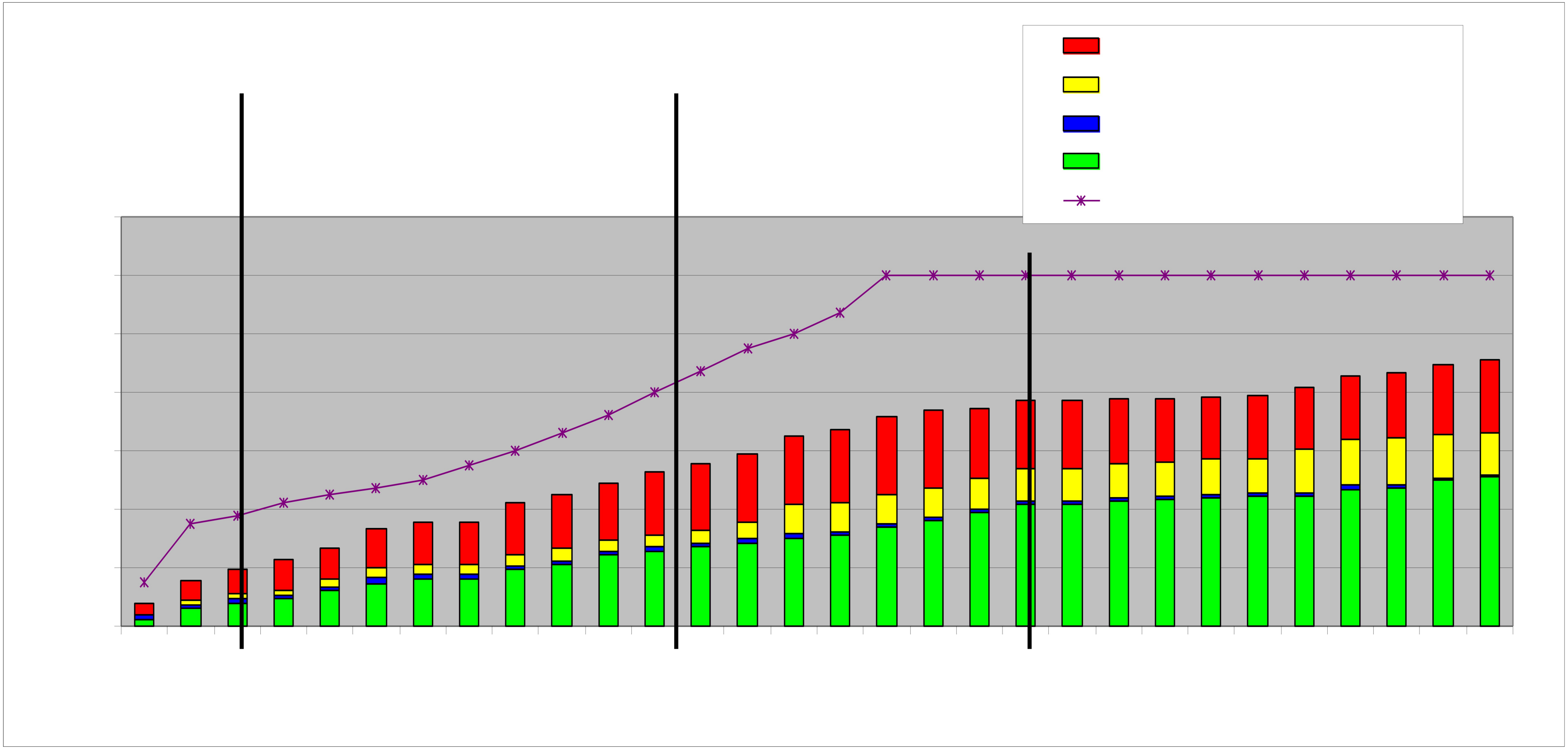
- Basic information:
  - Work done
    Test preparation, test execution, plan/actual comparison, defect situation
  - Work not achieved / delayed
    Explanation of issues, consequences, measures.
  - Work planned
    What to do until next reporting cycle
  - Urgent discussion points
    Issues, risks

Testing and Quality
Test Report



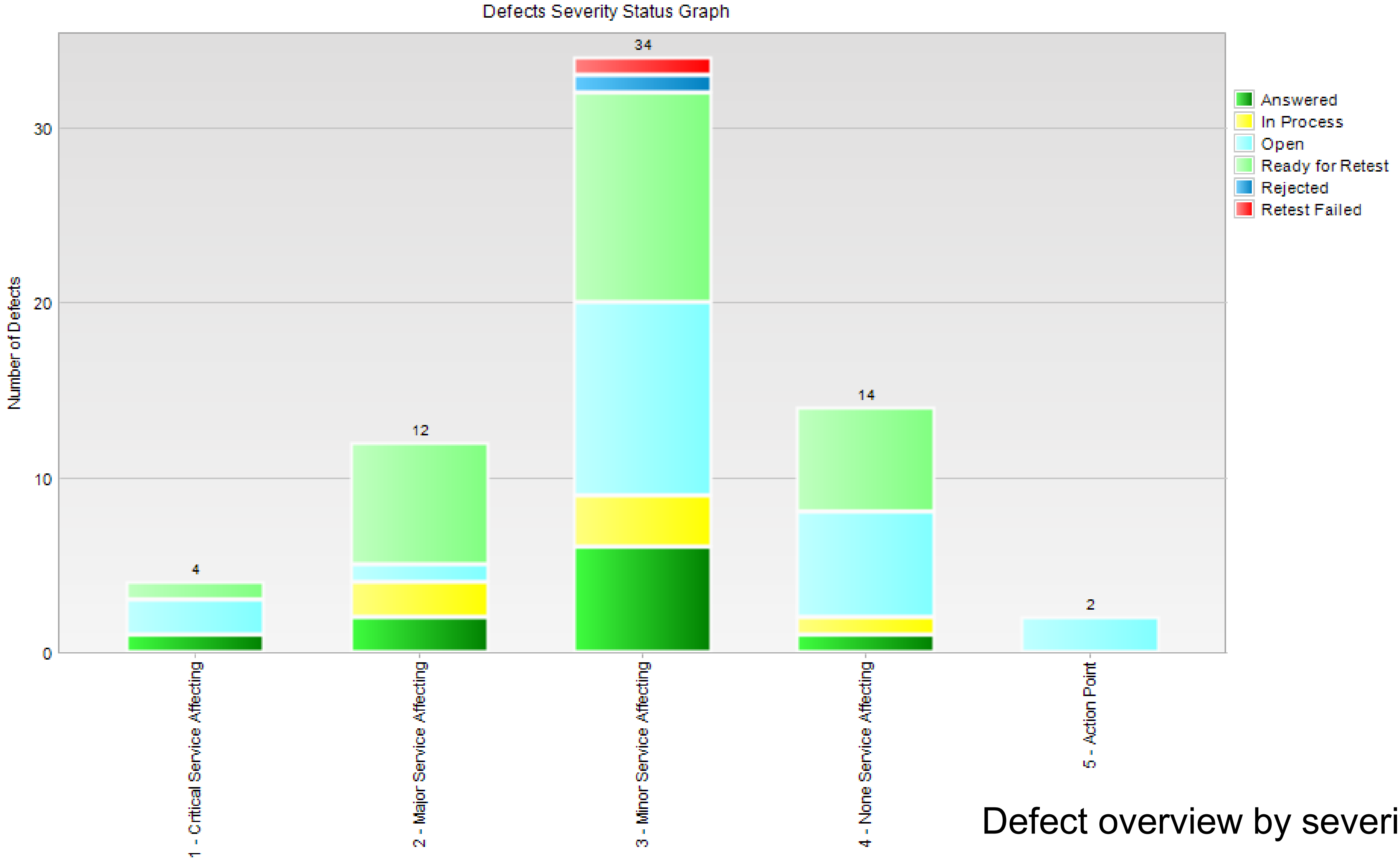
Test execution by area

Testing and Quality
Test Report



Test execution progress

Testing and Quality
Test Report



Defect overview by severity

Testing and Quality
Test Plan

- 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
- Goal of test plan: Understand what to test how intense.

Testing and Quality
Test Plan

- What is the effort for testing in a software project? What do I have to calculate?
- Approach [Whi11] [Whi11a]: Focus on
  - Attributes such as fast, usable, secure, etc.
  - Components like classes, module names and features of the application.
  - Capabilities – verbs that describe user actions and activities.

Testing and Quality
Test Plan

- Basic estimations
  - How many test cases?
  - Time for creation / review / overworking of one test case
  - Time for execution of one test case
  - How many defects do we expect?
  - Time to manage one defect

Want to learn more?

- Get educated!
- Professional organizations, e.g.
  - International Software Testing Qualifications Board, http://www.istqb.org; Certified Testers:
    - Foundation Level
    - Advanced Level
    - Expert Level
  - Americas Requirements Engineering Association [ARA14]
  - International Requirements Engineering Board, [IREB14]; “Certified Professional for Requirements Engineering”

Want to learn more?

- Books
  - 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
  - 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

Sources (1/2)

[ARA14] Americas Requirements Engineering Association, http://a-re-a.org/

[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

[ISTQB-GWP12] Glossary Working Party of International Software Testing Qualifications Board: Standard glossary of terms used in Software Testing, Version 2.2, 2012, http://www.istqb.org/downloads/glossary.html

[Jaw13] Ranjeet Jawale: Defect clustering & Pesticide paradox, 2013, http://www.softwaretestingclub.com/profiles/blogs/defect-clustering-pesticide-paradox

[IREB14] International Requirements Engineering Board, 2014, http://www.ireb.org/

Sources (2/2)

[Ric05] Randall W. Rice: STBC The Economics of Testing, 2005, http://www.riceconsulting.com/public_pdf/STBC-WM.pdf

[Sta94] The Standish Group, Standish Group survey 1994

[TDD05] Test-Driven Development: Concepts, Taxonomy, and Future Direction, IEEE Sep 2005

[Whi11] James Whittaker: EuroSTAR Software Testing Video: Ten Minute Test Plan with James Whittaker, 2011, http://www.youtube.com/watch?v=QEu3wmgTLqo

[Whi11a] James Whittaker: The 10 Minute Test Plan, 2011 , http://googletesting.blogspot.com/2011/09/10-minute-test-plan.html

[Wie99] Karl E. Wiegers: Writing Quality Requirements, 1999, http://processimpact.com/articles/qualreqs.html

[Wik14] Wikipedia: ISO/IEC 9126, 2014, http://en.wikipedia.org/wiki/ISO/IEC_9126