Software Testing

Lesson 4
Requirements
V1.0

Uwe Gühl

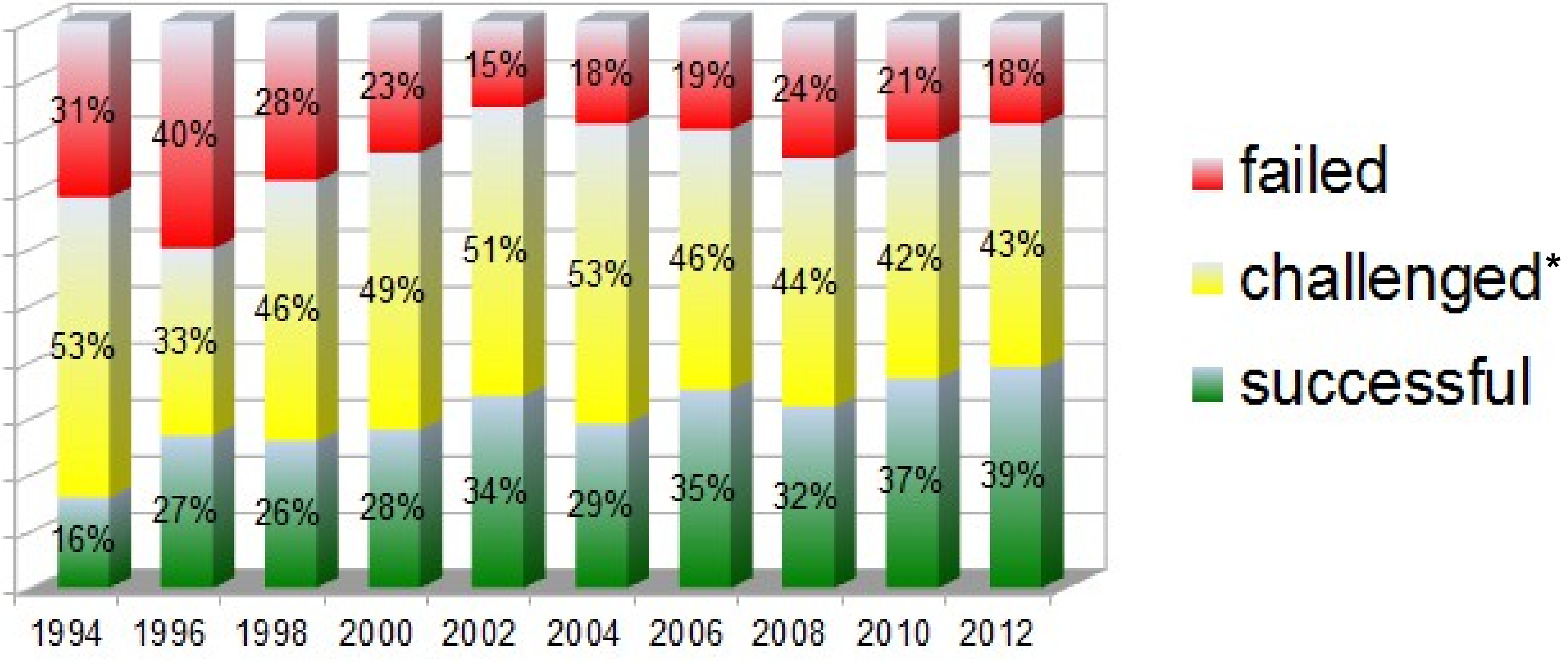
Winter 2013 / 2014

Contents

- Introduction
- Definitions
- Finding Requirements
  Business Scenarios and Non-Functional Requirements
- Writing Requirements
  Guidelines
- Changing Requirements
- Sources / More

Introduction

Result of an analysis of more than 9000 IT projects [Sta13]



*** overrun 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%

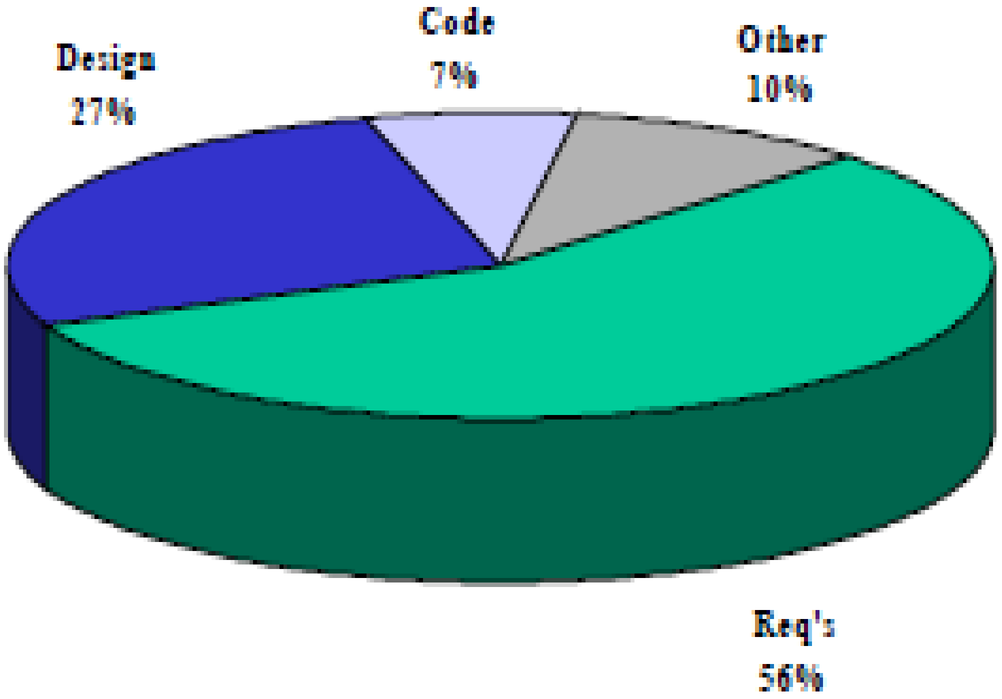
Introduction

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

Source of defects [Ric05]:



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

Definitions

Requirement [IEEE610.90], [Win99]:

(1) A condition or capability needed by a user to solve a problem or achieve an objective.

(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents.

(3) A documented representation of a condition or capability as in (1) or (2).

Definitions

Requirements analysis [IEEE610.90], [Win99]:

(1) The process of studying user needs to arrive at a definition of system, hardware or software requirements.

(2) The process of studying and refining system, hardware or software requirements.

Definitions

Requirements Engineer (1/2) [Mod14]

- Synonyms: Requirements Analyst, Functional Architect, Business Systems Analyst, Business Analyst (generic term).
- There is no industry standards for the scope of the requirements engineer.
  It's something between the IT business analyst and systems analyst.
- Abilities: A Requirements Engineer masters …
  … subject area
  … analysis
  … information technology

Definitions

Requirements Engineer (2/2) [Mod14]

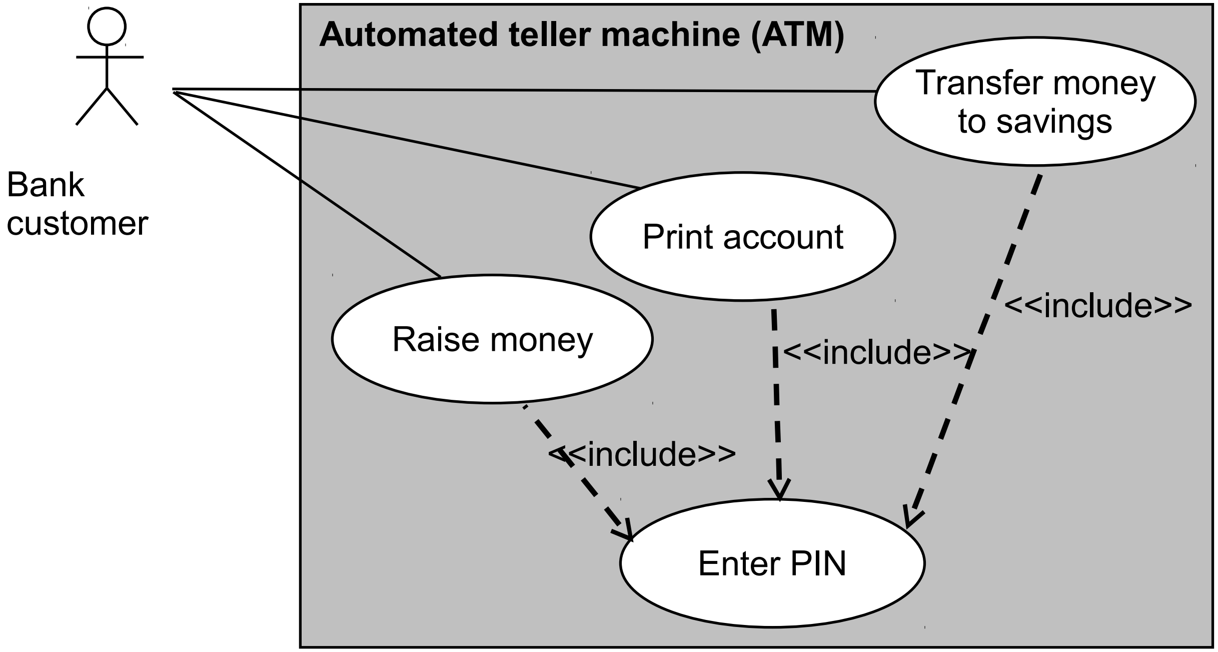
- Role description:
  Working with project stakeholders and end users to
  - detect,
  - understand,
  - analyse, and
  - document

  the requirements for a system in order to solve a given business problem.

Definitions

Use Case [Wik14a]

- List of steps, typically defining interactions between an actor and a system, to achieve a goal.



Example of a Use Case Diagram

Definitions

Example of a
Use Case Description ▶

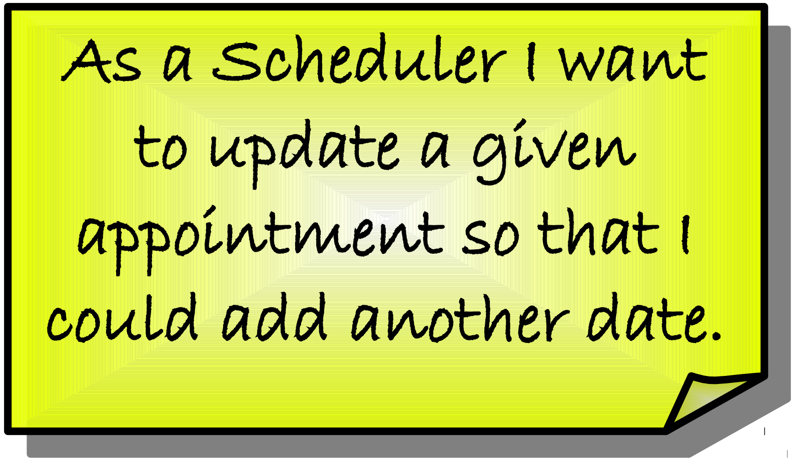
Id / Name	214 / Rent a car
Short description	A customer comes to the car rental agency and chooses a car which he rents for a fixed period
Actors	Customer, agent
Trigger	Customer asks agent
Pre condition	The rental system is ready to get customer data and to realize a lease contract
Result	Leasing is done, and the customer has signed the contract
Post condition	The rental system is ready to get customer data and to realize a lease contract
Activities	1. Enter customer data. If customer is yet not registered ⇨ UC *12 Register customer.* 2. Enter desired car category 3. Enter desired leasing period 4. If a car is available in the desired period: 1. Reserve a car 2. Enter credit card information 3. Print contract and sign Otherwise: Adapt item 2. or 3., if possible

Definitions

User Story [Mou14]

Short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. Proposed template:

As a <type of user>,
I want <some goal>
so that <some reason>.



As a Scheduler I want to update a given appointment so that I could add another date.

Example of a User Story

Definitions

Business Scenario
(Synonym Business Use Case)

- A Business Scenario is a collection of related, structured activities or tasks, so that a particular customer achieves a particular goal.
- A Business Scenario is typically composed of a set of Use Cases (Use Case chains).

Finding Requirements

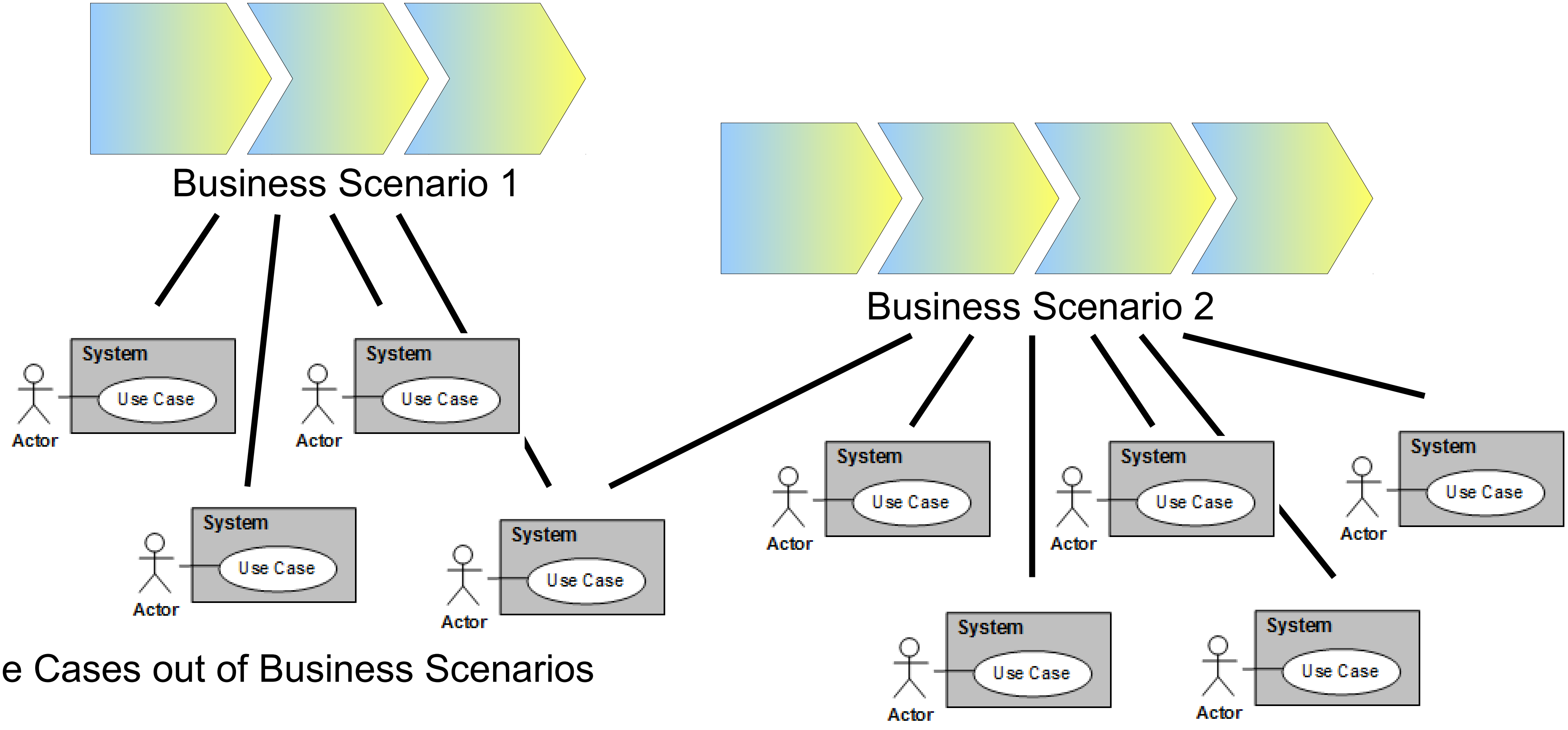
- A goal of Requirements Engineering is to get a complete, consistent, modifiable, and traceable software requirement specification [Wie99].
- How to get “complete” requirements?
- Find “the right people”, e. g. in using
  - Stakeholder analysis
  - Environment analysis

Finding Requirements

- There are many good ideas around how to identify requirements:
  - Manuals of older / comparable systems
  - Requirements workshops
  - Interviews with stakeholder and end users
  - Paper prototyping
- We will focus on identifying
  - Business Scenarios
  - Non-Functional Requirements

Business Scenarios

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



Use Cases out of Business Scenarios

Business Scenarios

Example

1. User enters a search term
2. User gets a list of results
3. User chooses out of the list of results a document
4. User changes for the document the font size to 44 pixel
5. User overlays the document with a grid
6. User adopts setting for all documents

Business Scenarios

Example

1. User enters a search term
2. User gets a list of results
3. User chooses out of the list of results a document

UC 1

4. User changes for the document the font size to 44 pixel
5. User overlays the document with a grid

UC 2

6. User adopts setting for all documents

UC 3

Business Scenarios Guideline

- A Business Scenario should describe a concrete, unambiguous, and complete action on process level.
- Definition of a main scenario, contenting all important features (success story)
- Definition of important branches as second step
- Definition of important exceptions / faults
  (e. g. what happens if a search finds no result)

Business Scenarios Guideline

- Use Case diagrams and activity diagrams as well as visualization with screenshots could be used for better communication.
- Active description with numbering of the steps
- Avoid generalization like
  - “and so on”
  - “etc.”
  - “easy”
  - “different options”

Business Scenarios Proceeding

There are several possibilities to identify Business Scenarios.

It's important to find people who could help in defining the Business Scenarios.

- Interviews
- Paper Prototyping
- Desktop Tests
- Workshops

Business Scenarios Advantages

- … bring forward the common understanding of business processes and their importance
- … are a basic for identification of Use Cases
- … basic for project controlling (Which Business Scenario will be realized in which release?)
- … help in prioritization of features and Use Cases
- … show from the end user point of view advantages of specified features

Non-Functional Requirements Motivation

- Unknown Non-Funtional Requirements may cause problems in IT projects, if so called “self evident requirements” are not fulfilled (security, performance, load).
- Requirement documents often leave the area “Non-Functional Requirements” empty or imprecise (“fast”, “easy to use”, “secure”) → IT Architecture cannot follow conditions

Non-Functional Requirements Motivation

- Late changes in software architecture are often complex and time-consuming
→ Early communication and common understanding concerning non-functional requirements is necessary
- Proposal:
Early identification of non-functional requirements!
- Presented proceeding was applied successfully in a media company

Non-Functional Requirements ISO/IEC 9126 Quality Model

Software quality – ISO/IEC 9126 [Wik14]

- ISO/IEC 9126 Software engineering – Product quality
  - is 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

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

Non-Functional Requirements ISO/IEC 9126 Quality Model

1 Functionality

2 Reliability

3 Usability

4 Efficiency

5 Maintainability

6 Portability

Non-Functional Requirements Proceeding

Execution of a workshop;
Agenda could cover:

1. Current Status
   Goal: Common understanding
   1. Overview, status of requirements
   2. System context, general set-up, actors, interfaces to systems to be considered
   3. System architecture ideas
2. **Start:** Presentation and explanation of non-functional requirements

Non-Functional Requirements Proceeding

Agenda (extract)

3. **Prio:** Prioritization of characteristic / sub-characteristic criteria by requirements engineers / development
4. **Tasks:** Definition of concrete quality criteria / acceptance criteria, assigning 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

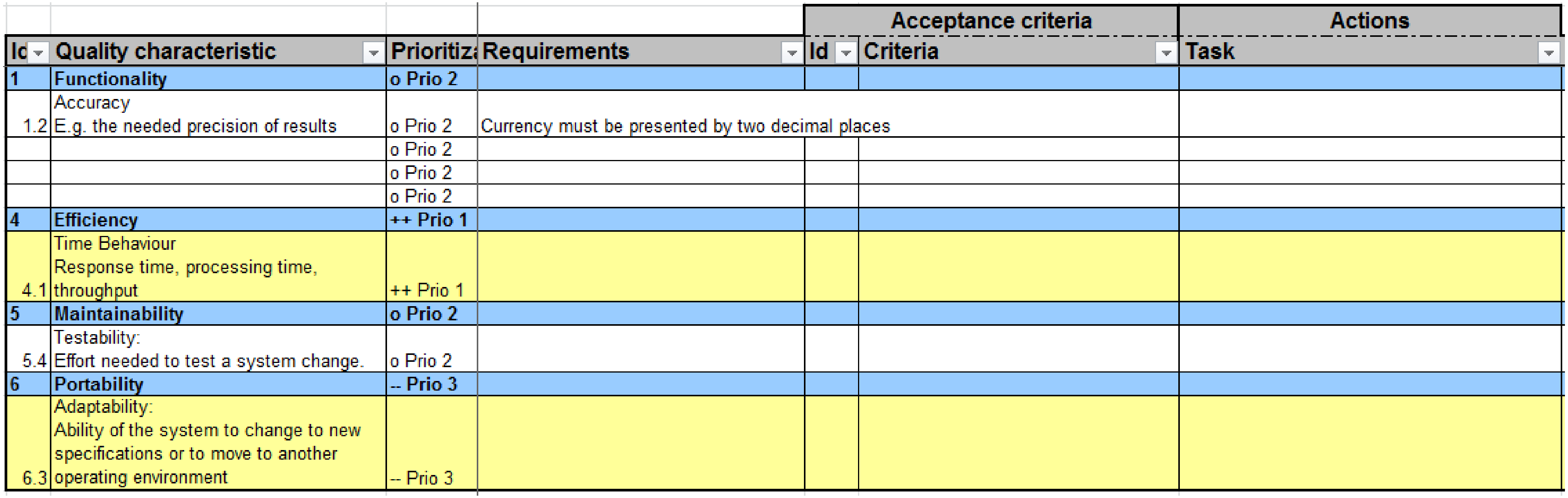
Non-Functional Requirements Proceeding – Example (Prio)

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
Id	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				

Writing Requirements

- Assumption: Requirements / Ideas are found
  - … as text fragments
  - … as minutes of workshops
  - … as pictures of story cards collected on a wall
- Now look into it.
  Goal: Writing good requirements
  HowTo: Using guidelines

Writing Requirements

Good requirements are [Sca11]:

- Correct: They have to say the right things.
- Consistent : They can’t contradict each other.
- Unambiguous: Each must have one interpretation.
- Complete: They cover all the important stuff.
- Relevant: Each must meet a customer need.
- Testable: There must be a way to tell if they are satisfied.
- Traceable: There must be a way to determine their origin.

Writing Requirements Guidelines

- KISS – Keep it simple and smart
  - Keep sentences and paragraphs short.
  - Use the active voice.
  - Use proper grammar, spelling, and punctuation.
  - Use terms consistently and define them in a glossary or data dictionary.

Quality measure

Glossary to speak “the same language”.
There should be only one common glossary.
There should be one responsible.

[Wie99]

Writing Requirements Guidelines

- Prioritize the 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.
- Excerpt (out of agile software development)
  In iteration planning requirements are selected out of a product backlog to be realized – following prioritization by customer.

Writing Requirements Guidelines

- Add to defined requirements acceptance criteria
  - Use concrete examples.
  - 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.

Writing Requirements Guidelines

- Use the “right” granularity
  - A helpful granularity guideline is to write individually testable requirements.
    If you can think of a small number of related tests to verify correct implementation of a requirement, it is probably written at the right level of detail.
  - Watch out for multiple requirements that have been aggregated into a single statement. "and" / "or" in a requirement
    ⇨ Several requirements might have been combined.

[Wie99]

Writing Requirements Guidelines

- Consistent level of detail
  - Not too detailed
    For example, "A valid color code shall be R for red" and "A valid color code shall be G for green" might be split out as separate requirements.
  - Not too general
    For example, "The product shall respond to editing directives entered by voice" describes an entire subsystem, not a single functional requirement.

[Wie99]

Writing Requirements Guidelines

- Once and only once
  - Avoid stating requirements redundantly in the specification.
  - Reason
    If there are multiple instances of requirements:
    - Difficult maintenance of the requirements specification document
    - Source for inconsistencies, if not all redundant requirements get updated at the same time

[Wie99]

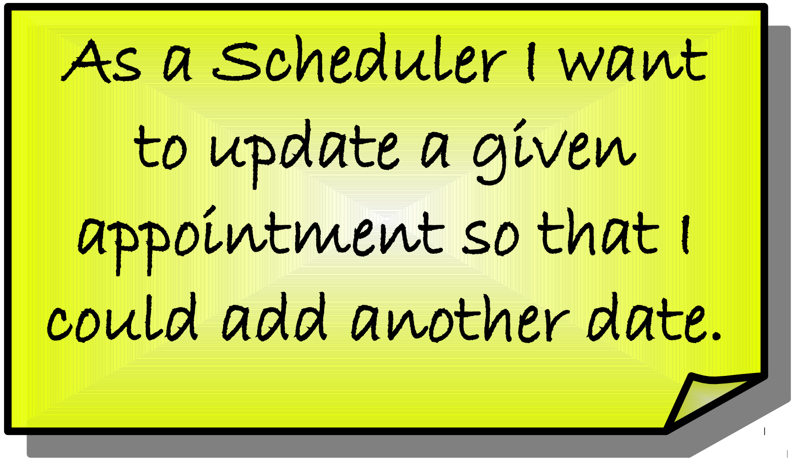
Writing Requirements Guidelines

- Change perspective
  - To see if a requirement statement is sufficiently well defined, read it from the developer’s perspective.
  - Mentally add the phrase, "call me when you’re done" to the end of the requirement and see if that makes you nervous!
- Use check lists, e.g. for a use case descriptions

[Wie99]

Writing Requirements Example: User Stories

- User Stories are high-level requirements
- Large User Stories are known as Epics
(compare to Business Scenario)
– typically too big to be implemented in an iteration
- User Stories are often written on index cards or sticky notes, and stored on walls.
- They shift the focus from writing about features to discussing them.
- User Story is something like a promise to talk.



As a Scheduler I want to update a given appointment so that I could add another date.

Example of a User Story

Writing Requirements Example: User Stories

- Well written user stories should follow the **INVEST** model [Wak03]
  - **I** ndependent – no overlap, no dependencies
  - **N** egotiable – captures the essence, not details
  - **V** aluable – a specified value for the customer
  - **E** stimable – to help in planning and prioritization
  - **S** mall – should be conducted in a sprint
  - **T** estable – more effective, if tests were written before implementation

Changing Requirements

- Imagine: In a 2 years project all the requirements defined in the first 2 months get realized as specified …

  What do you think?

- If we don't want to take the requirements “as is” we have to look into it and to adapt in case.

Changing Requirements

Possible reasons:

- Stakeholder does not like delivered solution.
- Market changed.

Early changes could be required after reviews.
Review Technique: Try to be active:

- Problems?
  Ask questions!
- Proposals?
  Propose better statements!

[Wie99]

Changing Requirements

Regular look at the requirements as they are living!

- Prioritization
  Focus on the most important requirements and on the requirements to be implemented next.
- Enforce Communication
  Requirements Engineer ⬄ Developer ⬄ Tester
- Regular Milestones, short development cycles
  Regular Feedback concerning implementation of requirements.

Changing Requirements

Develop project culture:

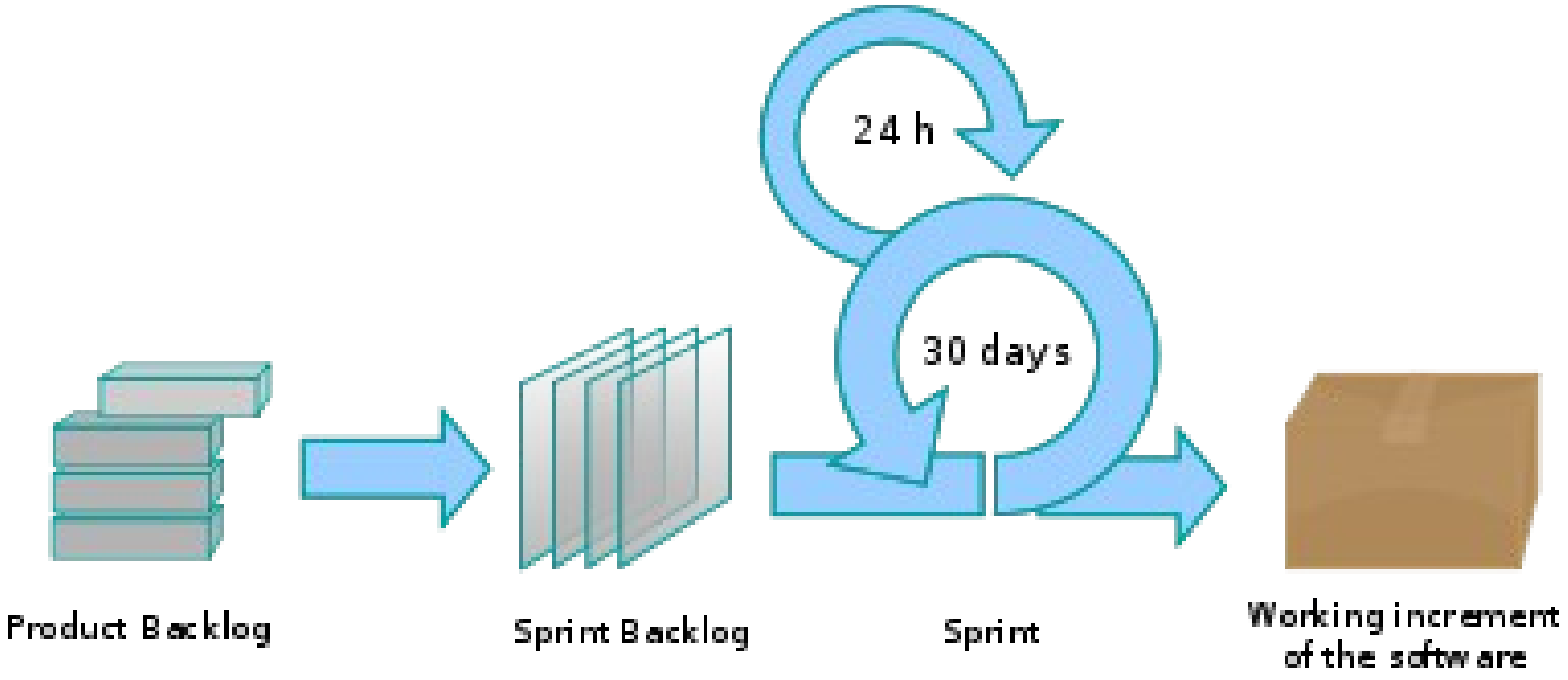
- Love Changes!
  - Changes are okay – better to change instead of implementing something “wrong”!
  - Clear rules have to be defined, agreed and followed (Change Management Process).
- Love Defects!
  - The earlier we detect defects, the cheaper the elimination.
  - All defects we detect, the customer won't find.

Changing Requirements

- Ideas as discussed before result in “Agile software development”; example Scrum:
  - Basics: User Stories as “atomic requirements”.
  - Collection of User Stories as basic wish list what makes the product great.
  - Regular planning: Agreement, which user stories to be implemented in next sprint
    → Following prioritization by customer.
  - Regular review: Acceptance of delivered solution.

Changing Requirements

- Ideas as discussed before result in "Agile software development"; example Scrum



http://en.wikipedia.org/wiki/File:Scrum_process.svg

Summary (1/2)

- Requirements Engineering
  … to get better projects
  … to face main problems of IT projects.
- First activity: Identification of requirements.
- Business Scenarios
  - to focus on business related requirements
  - to find Use Cases with top down approach)
  - to implement the most important requirements first.

Summary (2/2)

- Non-Functional Requirements
  - to be taken serious
  - to be identified e.g. with ISO / IEC 9126 as check list.
- There are a lot of techniques, “how-to”, and ideas to identify, to write, and to update requirements.
- A constructive, willing to learn organisation is extremely helpful for successful requirements engineering.

Want to learn more?

- Professional organizations, e.g.
  - Americas Requirements Engineering Association [ARA14]
  - International Requirements Engineering Board, [IREB14] → offer a certification program to get “Certified Professional for Requirements Engineering”.
- Books
  - Klaus Pohl, Chris Rupp: Requirements Engineering Fundamentals, 1st edition, Rocky Nook Inc., 2011
  - Karl E. Wiegers: More About Software Requirements: Thorny Issues and Practical Advice, Microsoft Press, 2005
  - Ian Alexander, Ljerka Beus-Dukic: Discovering Requirements – How to Specify Products and Services, Wiley, 2009

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

[Mou14] Mountain Goat: User Stories; An Agile Requirements Approach, http://www.mountaingoatsoftware.com/agile/user-stories., 2014

[IEEE610.90] IEEE Standard Glossary of Software Engineering Terminology IEEE Standard 610, IEEE, New York, 1990

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

[Mod14] MODERNanalyst.com: The Requirements Engineer Role; http://www.modernanalyst.com/TheProfession/Roles/RequirementsEngineer/tabid/188/Default.aspx, 2014

Sources (2/2)

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

[Sca11] Christopher Scaffidi: Requirements, Lecture, 2011, http://web.engr.oregonstate.edu/~cscaffid/courses/CS361_F11/lecture3-requirements.ppt

[Sta94] The Standish Group, Standish Group survey 1994

[Sta13] The Standish Group: CHAOS MANIFESTO 2013; ChaosManifesto2013.pdf

[Wak03] Bill Wake: INVEST in Good Stories, and SMART Tasks, 2003, http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

[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

[Wik14a] Wikipedia: Use Case, 2014, http://en.wikipedia.org/wiki/Use_case

[Win99] Mario Winter: Qualitätssicherung für objektorientierte Software: Anforderungsermittlung und Test gegen die Anforderungsspezifikation, 1999, http://deposit.fernuni-hagen.de/2527/1/dissWinter.pdf