Software Project Management

Lesson 4 – Requirements

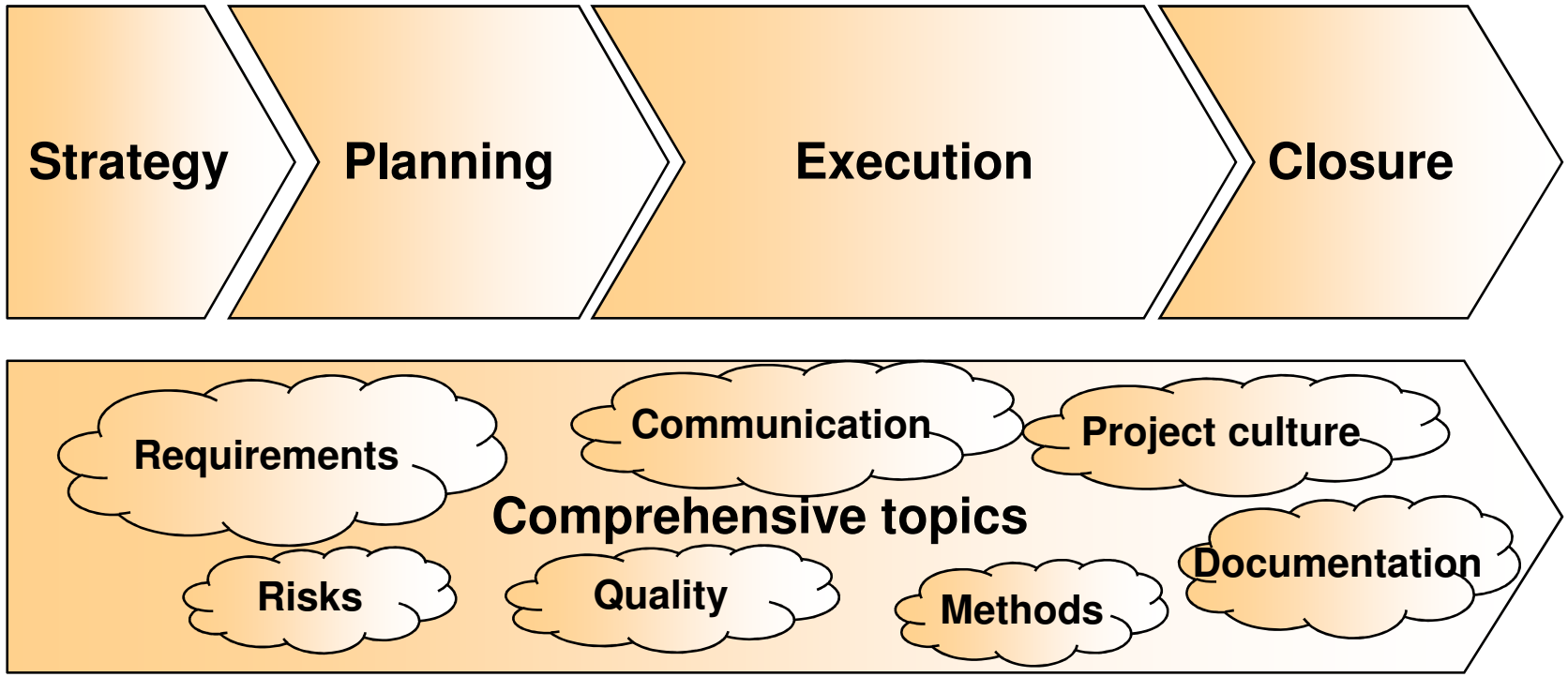
Uwe Gühl

Winter 2015 / 2016

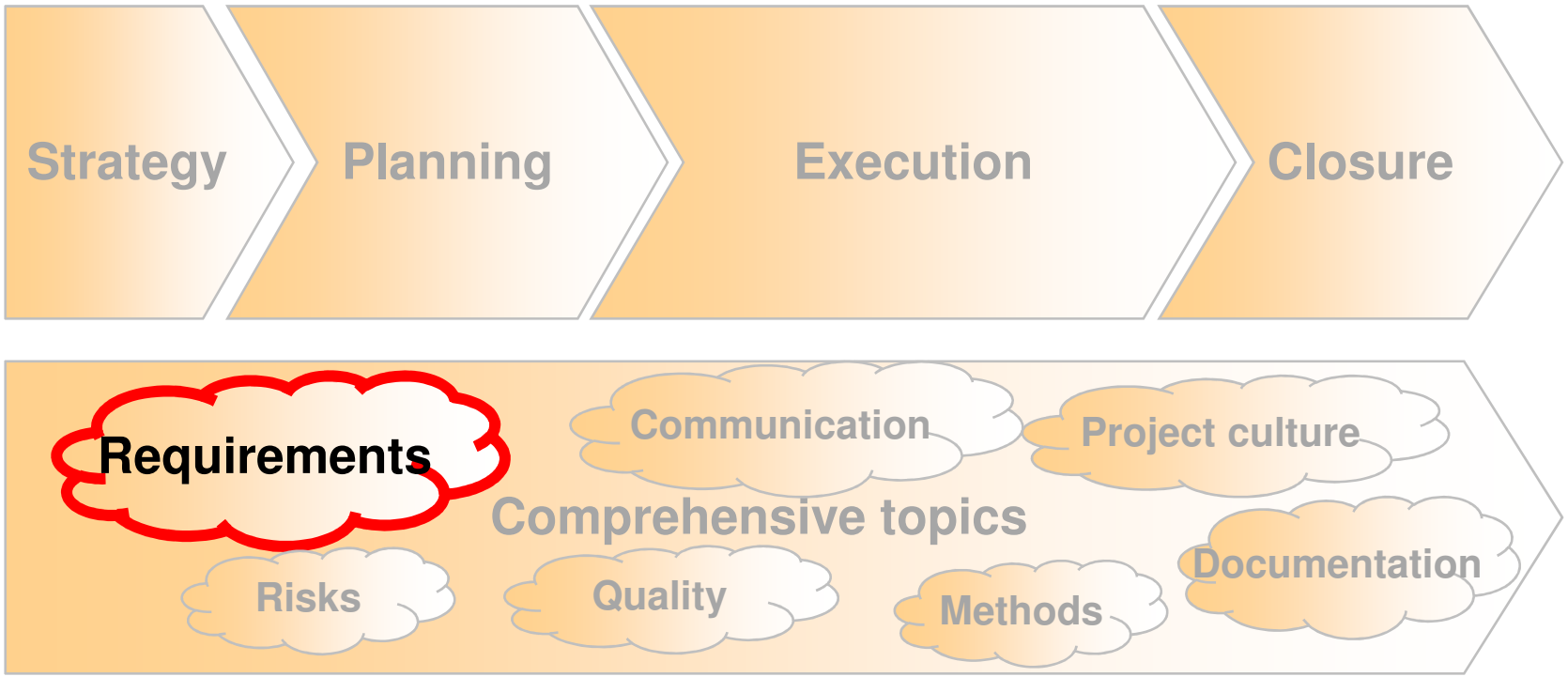
Contents

- Requirements
  - Introduction
  - Definitions
  - Identification of requirements
    - Business Scenarios
    - ISO/IEC 9126
    - Requirements list
  - Design of requirements
  - Review of requirements
  - Change of requirements
  - Summary
  - Sources

Introduction

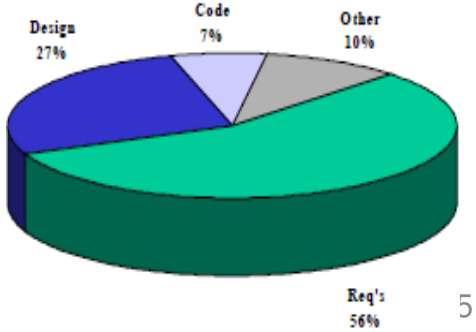


Introduction



Introduction

- Goal of Requirements Engineering:
  Common understanding of principal and contractor concerning a product or system to be developed
- Importance of requirements
  - 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]



Definition

- Requirements Engineering [Gli14]
  A systematic and disciplined approach to the specification and management of requirements with the following goals:
  (1) Knowing the relevant requirements, achieving a consensus among the stakeholders about these requirements, documenting them according to given standards, and managing them systematically
  (2) Understanding and documenting the stakeholders' desires and needs
  (3) Specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs

Definition

- 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).

Definition

- 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.

Definition

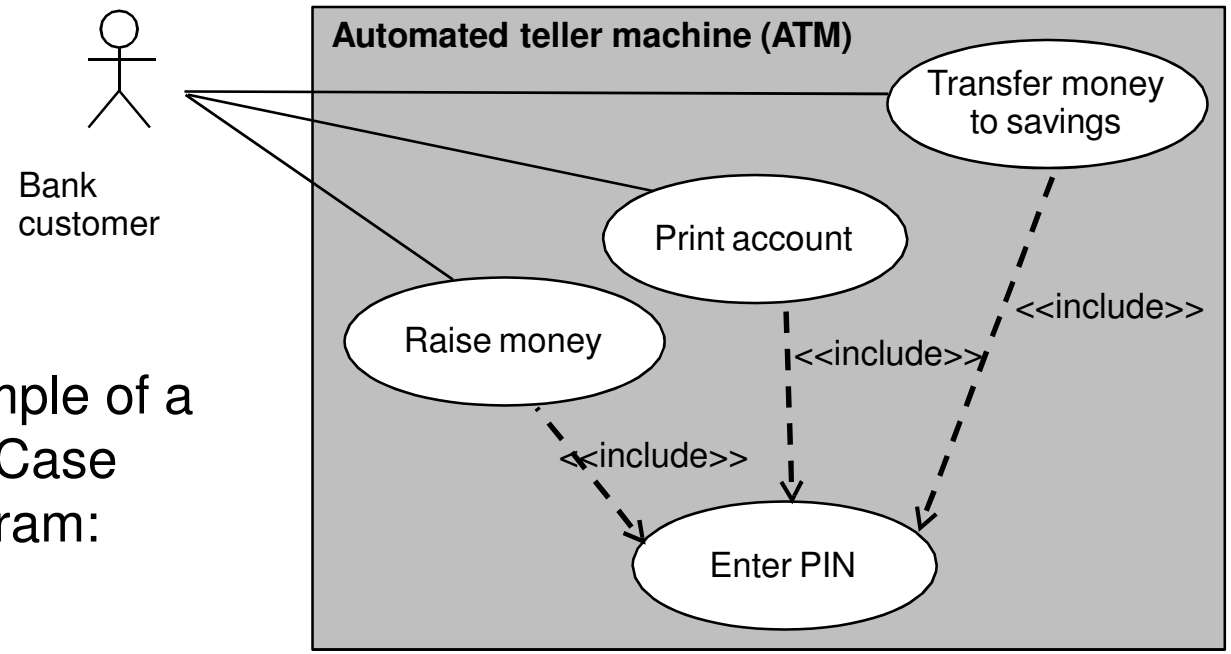
- Requirements Engineer [Mod16]
  Synonyms: Requirements Analyst, Functional Architect, Business (Systems) Analyst, Business
  - There is no industry standards for the scope of the requirements engineer.
  - It's something between the IT business analyst and systems analyst.
- A requirements engineer
  - masters subject area, analysis, and IT
  - works with project stakeholders to detect, understand, analyze, and document the requirements for a system

Definition

Use Case [Wik16a]

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

Example of a Use Case Diagram:



Definition

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: a. Reserve a car b. Enter credit card information c. Print contract and sign Otherwise: Adapt item 2. or 3., if possible

Definition

User Story [Mou16]

- 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

Definition

Business Scenario
(Synonym Business Use Case)

- Collection of related, structured activities or tasks, so that a particular customer achieves a particular goal
- Typically composed of a set of Use Cases (Use Case chains)

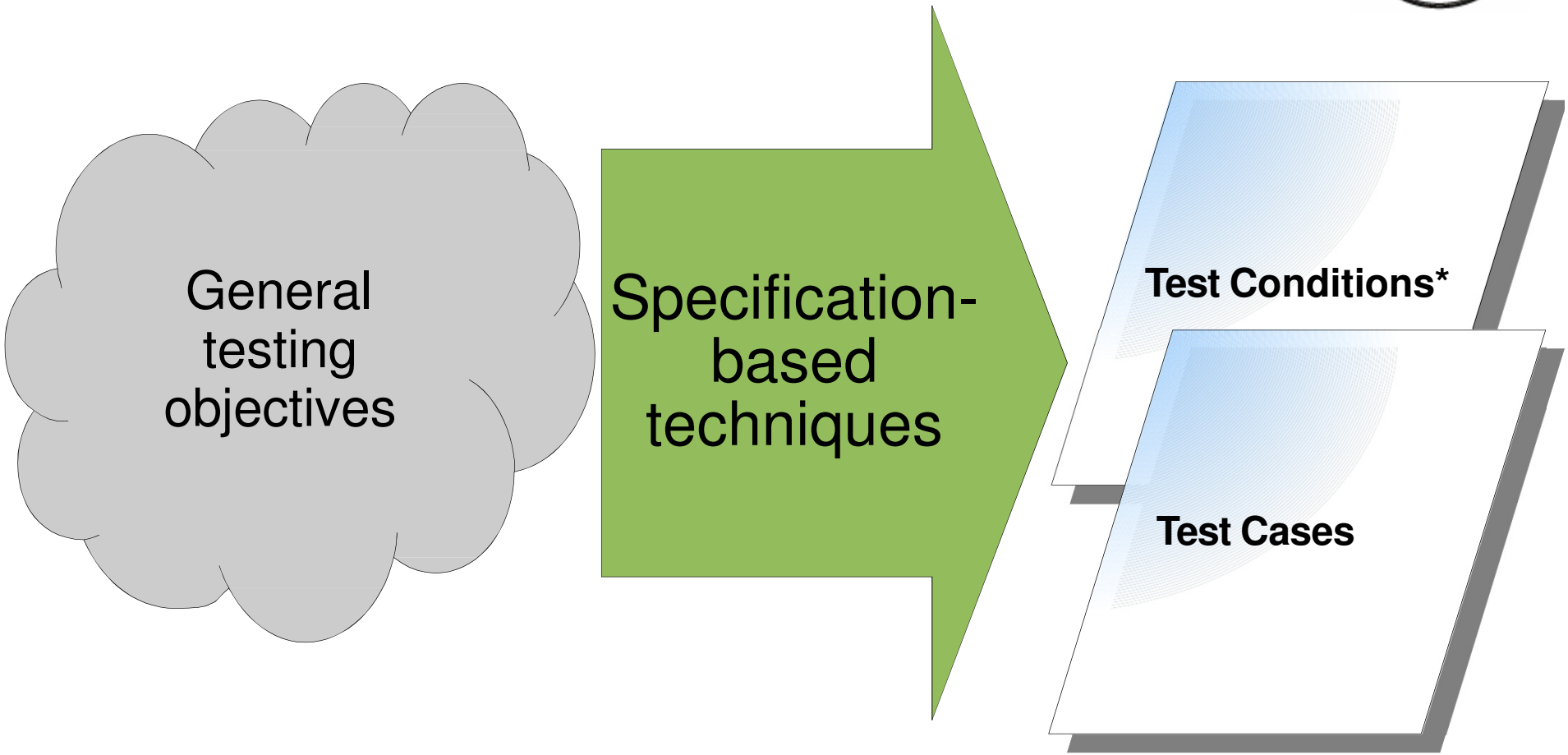
Identification of requirements

- How to identify requirements / risks?
  - Interviews with stakeholders
    E.g. Principal, marketing, sales, end user, project manager, ...
    Based on stakeholder analysis, environment analysis
  - Definition of Business Scenarios
    - … to identify business needs
    - … to define use cases (Top down approach)
    - … to identify test scenarios
    - … to prioritize testing activities
  - Methods: Interviews, paper prototyping, desktop tests, workshops

Identification of requirements

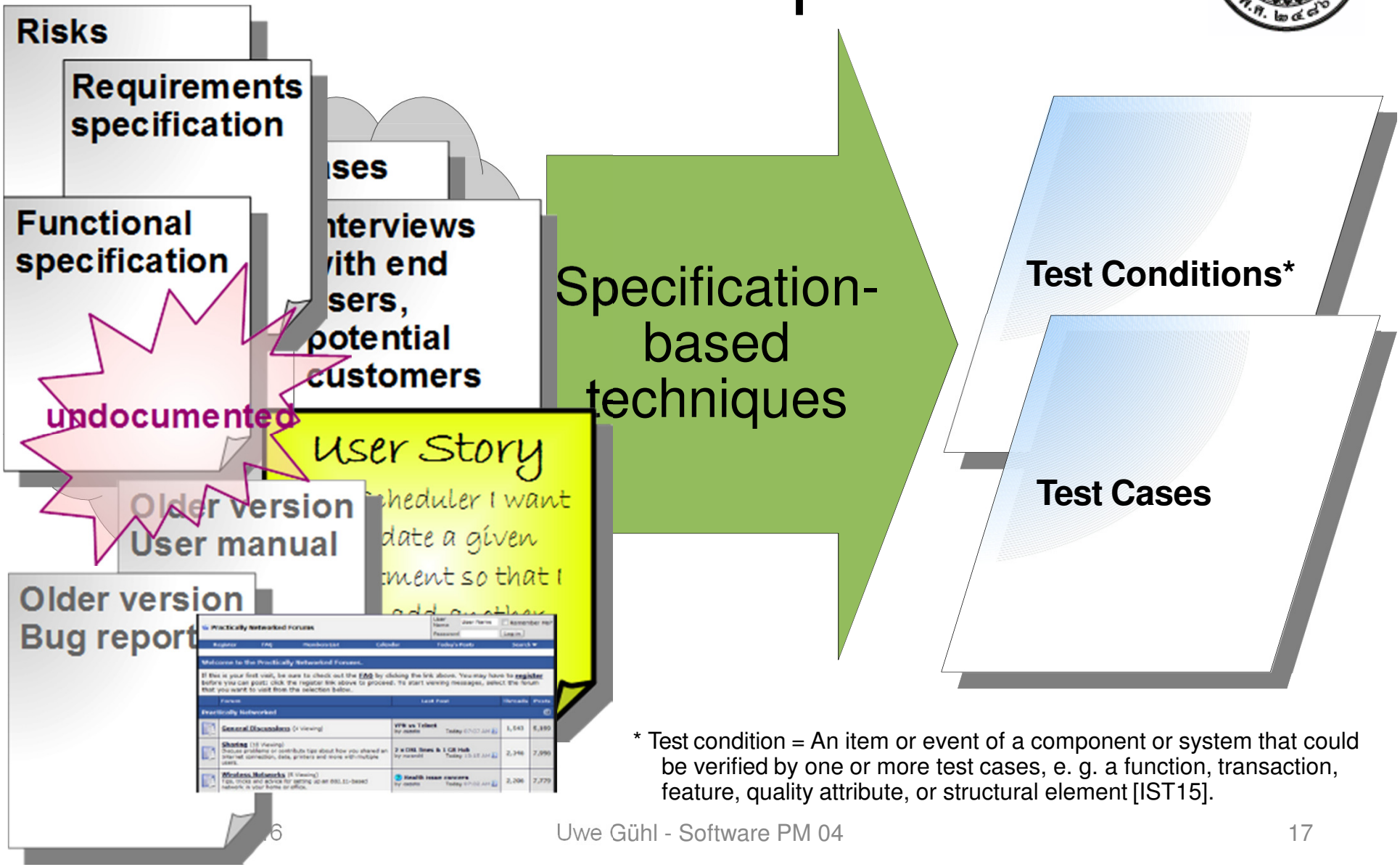
- Unknown Non-functional Requirements are a big risk in IT projects, if so called “self evident requirements” are not fulfilled (security, performance, load).
- Specification documents often leave the area “Non- Functional Requirements” empty or imprecise (“fast”, “easy to use”, “secure”)
  → IT Architecture cannot follow conditions
  → No proper test planning / design / execution
- Proposal: Early identification of non-functional requirements, e.g. using quality model defined by ISO 9126 [Wik16]

Identification of requirements



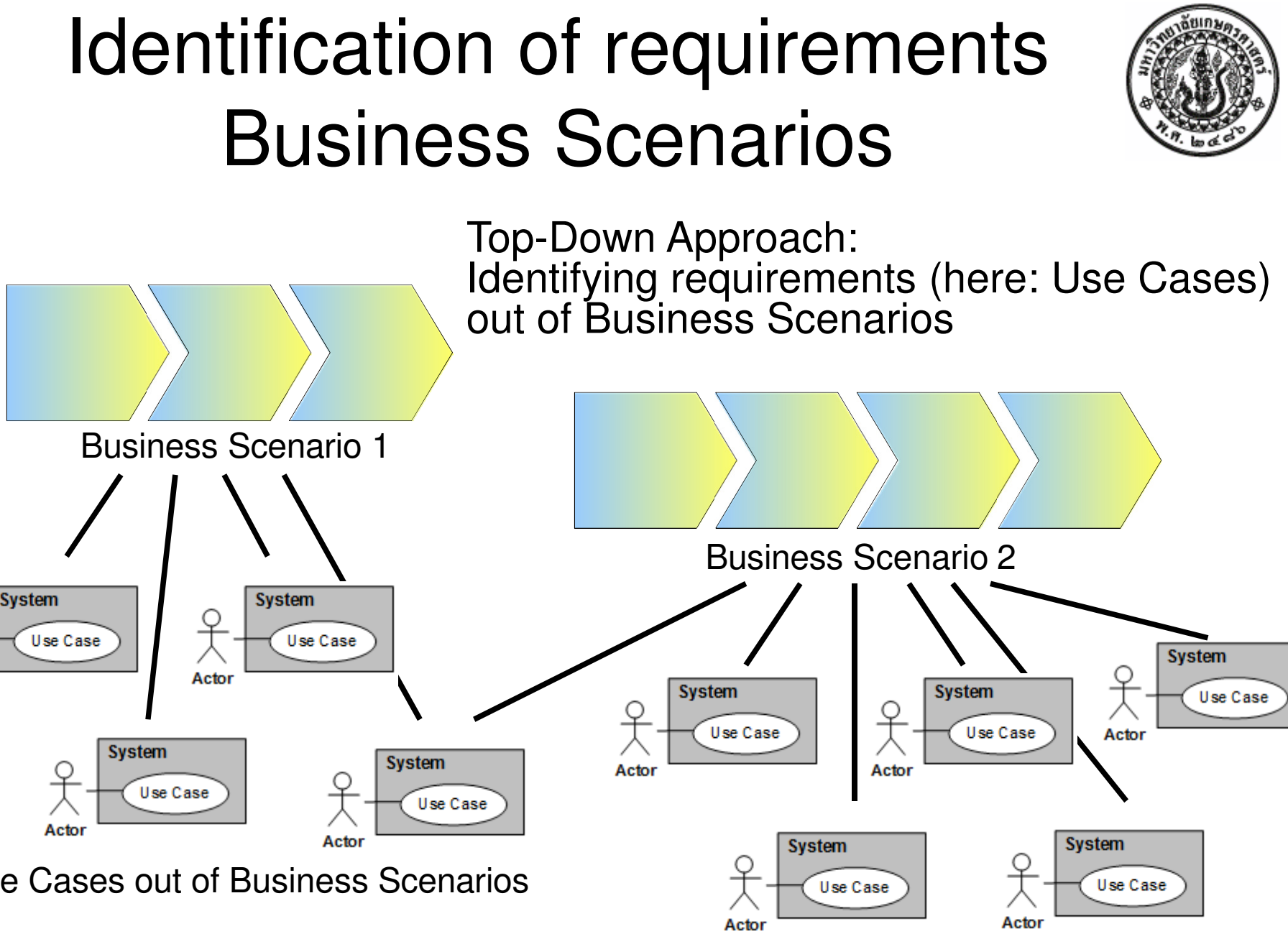
* Test condition = An item or event of a component or system that could be verified by one or more test cases, e. g. a function, transaction, feature, quality attribute, or structural element [IST15].

Identification of 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].

Identification of requirements
Business Scenarios



Identification of requirements 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

Identification of requirements Business Scenarios

Example

1. User enters a search term **UC 1**
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 **UC 2**
6. User adopts setting for all documents **UC 3**

Identification of requirements Business Scenarios

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

Identification of requirements Business Scenarios

- 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”

Identification of requirements Business Scenarios

- Advantages
  - bring forward the common understanding of business processes and their importance
  - are a basic for identification and prioritization of use cases
  - could help for project controlling (Which Business Scenario will be realized in which release?)

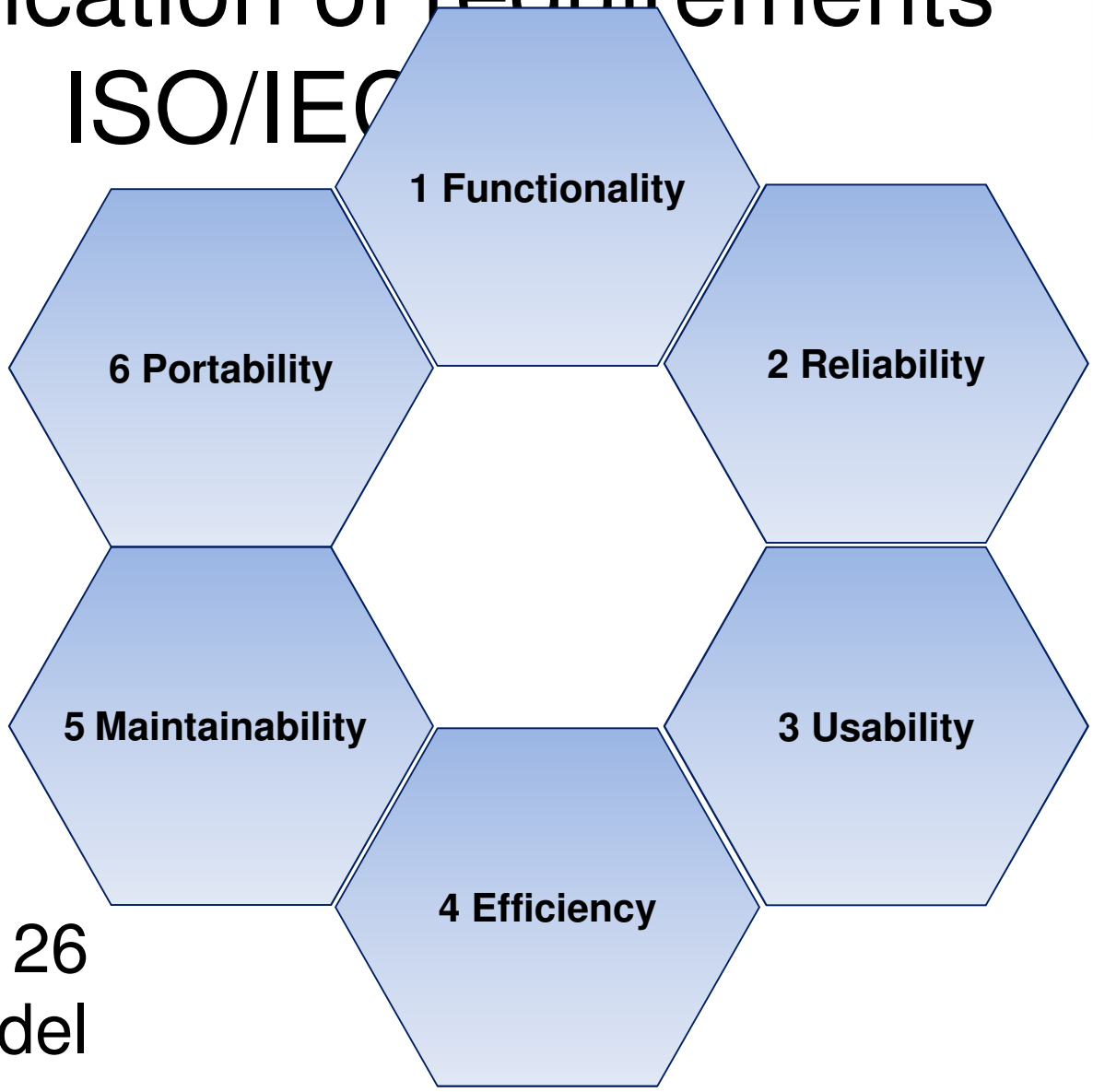
Identification of requirements ISO/IEC 9126

- Proposal: Using ISO/IEC 9126 [Wik16] to identify functional and non-functional requirements
- Expected result
  - Prioritization of quality criteria
  - List of corresponding requirements including acceptance criteria

Identification of requirements ISO/IEC 9126

- ISO/IEC 9126 Software engineering – Product quality [Wik16]
  - 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

Identification of requirements ISO/IEC



ISO/IEC 9126
Quality Model

Identification of requirements ISO/IEC 9126

Expected result: (1) Prioritization of quality criteria

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

Identification of requirements ISO/IEC 9126

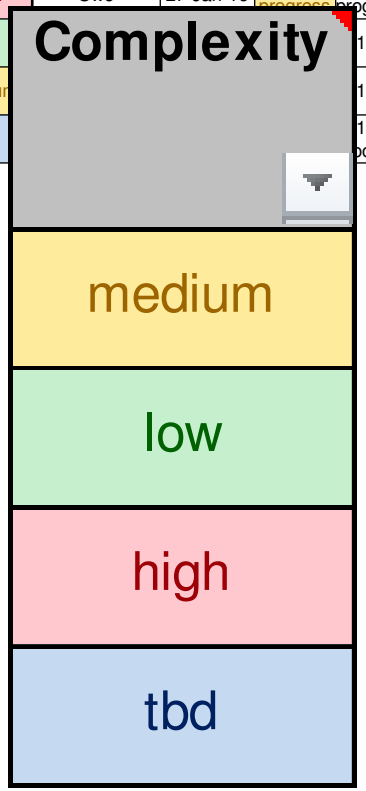
Expected result: (2) Collection of requirements, acceptance criteria, tasks to be executed, etc.

				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				

Identification of requirements
Requirements list

	Requirement description				Quantification		Implementation			
Id	Requirement	Stakeholder role	Stakeholder contact person	Initial date	Complexity	Priority	Responsible	Due date	Status	Actions
Req001	Example of a requirement, low complex, with high priority. Already in progress	operation	Ben	27-Jan-16	medium	high	Uwe	27-Jan-16	in progress	2016-01-26 [Uwe] Approved, implementation in progress
Req002	Business requirement, low priority	business	Dek	27-Jan-16	low	low				16-01-26 [Uwe] approved and delivered.
Req003	End user requirement, high complex	end user	John	27-Jan-16	high	mediur				16-01-26 [Uwe] declined, will not be implementd
Req004	Developer requirement, complexity and priority to be defined (tbd)	developer	Jim	27-Jan-16	tbd	tbd				16-01-26 [Uwe] Meeting planned to discuss oceeding

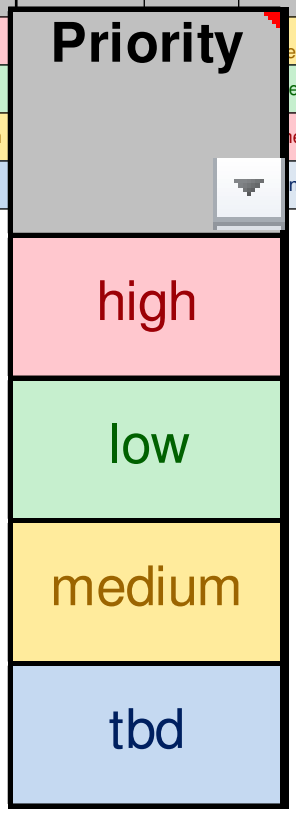
- Complexity:
  Attribute to calculate effort for implementation and testing
  - tbd = to be defined



Identification of requirements
Requirements list

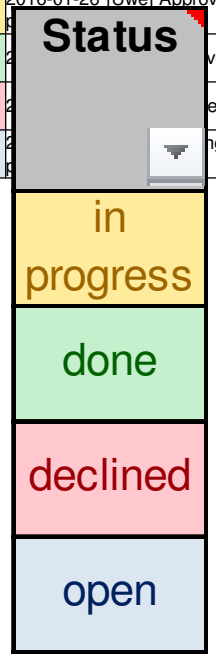
	Requirement description				Quantification		Implementation			
Id	Requirement	Stakeholder role	Stakeholder contact person	Initial date	Complexity	Priority	Responsible	Due date	Status	Actions
Req001	Example of a requirement, low complex, with high priority. Already in progress	operation	Ben	27-Jan-16	medium	high			ess	2016-01-26 [Uwe] Approved, implementation in progress
Req002	Business requirement, low priority	business	Dek	27-Jan-16	low	low			e	2016-01-26 [Uwe] approved and delivered.
Req003	End user requirement, high complex	end user	John	27-Jan-16	high	medium			ed	2016-01-26 [Uwe] declined, will not be implementd
Req004	Developer requirement, complexity and priority to be defined (tbd)	developer	Jim	27-Jan-16	tbd	tbd			n	2016-01-26 [Uwe] Meeting planned to discuss proceeding

- Priority:
  Urgency of implementation
  - tbd = to be defined



Identification of requirements
Requirements list

	Requirement description				Quantification		Implementation			
Id	Requirement	Stakeholder role	Stakeholder contact person	Initial date	Complexity	Priority	Responsible	Due date	Status	Actions
Req001	Example of a requirement, low complex, with high priority. Already in progress	operation	Ben	27-Jan-16	medium	high	Uwe	27-Jan-16	in progress	2016-01-26 [Uwe] Approved, implementation in
Req002	Business requirement, low priority	business	Dek	27-Jan-16	low	low	Uwe	26-Feb-16	done	ved and delivered.
Req003	End user requirement, high complex	end user	John	27-Jan-16	high	medium	Uwe	25-Mrz-16	declined	ed, will not be implementd
Req004	Developer requirement, complexity and priority to be defined (tbd)	developer	Jim	27-Jan-16	tbd	tbd	Uwe	29-Apr-16	open	ng planned to discuss



- Status:
  - open
    No activities yet
  - Declined
    No implementation
  - in progress
  - done
    Additional field “result” to be considered

Design of requirements

Motivation

- 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

Design of requirements

Good requirements:

- 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.

Design of requirements

Prioritization

- What are the most crucial and most risky requirements?
  - To be developed first
  - To be tested first
- 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.
- Important for scope management

Design of requirements

- 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 person responsible.

Source: [Wie99]

Design of requirements

- There should be no gaps, no inconsistencies
- Important: Acceptance criteria

  Excerpt (out of agile software development):
  "Definition of done" is an agreement to decide, when a realization of a user story is "done", means could be accepted by the customer.
  E.g. presentation successful, automated test cases passed.
- Use of concrete examples – the more realistic, the better => Basis for test cases
- To see if a requirement statement is sufficiently well defined, read it from the developer's perspective.

Design of requirements

- “Right” granularity
  - A helpful granularity guideline is to write individually testable requirement with a small number of related tests
  - Watch out for multiple requirements that have been aggregated into a single statement. "and" / "or" in a requirement
    → Several requirements might have been combined.

Source: [Wie99]

Design of requirements

- 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.

Source: [Wie99]

Design of requirements

- 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

Source: [Wie99]

Design of 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.

Design of 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

Review of requirements

Try to be active in reviewing requirements.

- Problems?
  Ask questions

- Proposals?
  Propose better statements

Source: [Wie99]

Review of requirements

Example 1
"The product shall provide status messages at regular intervals not less than every 60 seconds."

- Problems?
- Proposals?

Source: [Wie99]

Review of requirements

Example 1
"The product shall provide status messages at regular intervals not less than every 60 seconds."

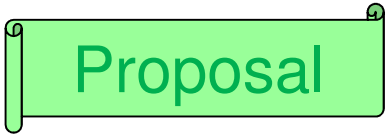
- Problems?
  1. What are the status messages and how are they supposed to be displayed to the user?
  2. What part of "the product" are we talking about?
  3. Is the interval between status messages really supposed to be at least 60 seconds, so showing a new message every 10 years is okay?
  4. Perhaps the intent is to have no more than 60 seconds elapse between messages; would 1 millisecond be too short?
  5. The word "every" just confuses the issue.
  6. Verifiable: As a result of these problems, the requirement is not verifiable.

Source: [Wie99]

Review of requirements

Example 1
"The product shall provide status messages at regular intervals not less than every 60 seconds."

- Proposals?



1. Status Messages.
   1.1. The Background Task Manager shall display status messages in a designated area of the user interface at intervals of 60 plus or minus 10 seconds.
   1.2. If background task processing is progressing normally, the percentage of the background task processing that has been completed shall be displayed.
   1.3. A message shall be displayed when the background task is completed.
   1.4. An error message shall be displayed if the background task has stalled.

Source: [Wie99]

Review of requirements

Example 1

"The product shall provide status messages at regular intervals not less than every 60 seconds."

- Proposal – Assessment
  - Splitting into multiple requirements
    - Each will require separate test cases
    - Each will be separately traceable.
  - If several requirements are strung together in a paragraph, it is easy to overlook one during construction or testing.

Source: [Wie99]

Review of requirements

Example 2

"The HTML Parser shall produce an HTML markup error report which allows quick resolution of errors when used by HTML novices"

- Problems?

- Proposals?

Source: [Wie99]

Review of requirements

Example 2
"The HTML Parser shall produce an HTML markup error report which allows quick resolution of errors when used by HTML novices"

- Problems?
  1. What goes into the error report?
  2. What is "quick"?
  3. How to find someone who calls herself an HTML novice and see if she can resolve errors quickly enough using the report?

Source: [Wie99]

Review of requirements

Example 2
"The HTML Parser shall produce an HTML markup error report which allows quick resolution of errors when used by HTML novices"

Proposal

- 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."

Source: [Wie99]

Review of requirements

Example 2
"The HTML Parser shall produce an HTML markup error report which allows quick resolution of errors when used by HTML novices"

- Proposal – Assessment
  - Defined: What needs to go into the error report,
  - Benefit: Designer decides what the report should look like.
  - Plus: An exception condition is defined:
    If there aren't any errors, don't generate a report.

Source: [Wie99]

Changing requirements

- Requirements are changing [Sch01]
  - Small projects: Up to 25% of requirements are changing
  - Large projects: Up to 50% of requirements are changing
- Possible reasons:
  - Stakeholder does not like delivered solution.
  - Market requests changed.

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

- Required: Positive attitude concerning change:
  - Changes are okay – better to change instead of implementing something “wrong”!
- Change management process
  Clear rules have to be defined, agreed and followed
  - Classical: Change management including a change control board (CCB)
  - Agile: Continuous update of backlog and considering changed requirements in sprint planning

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:

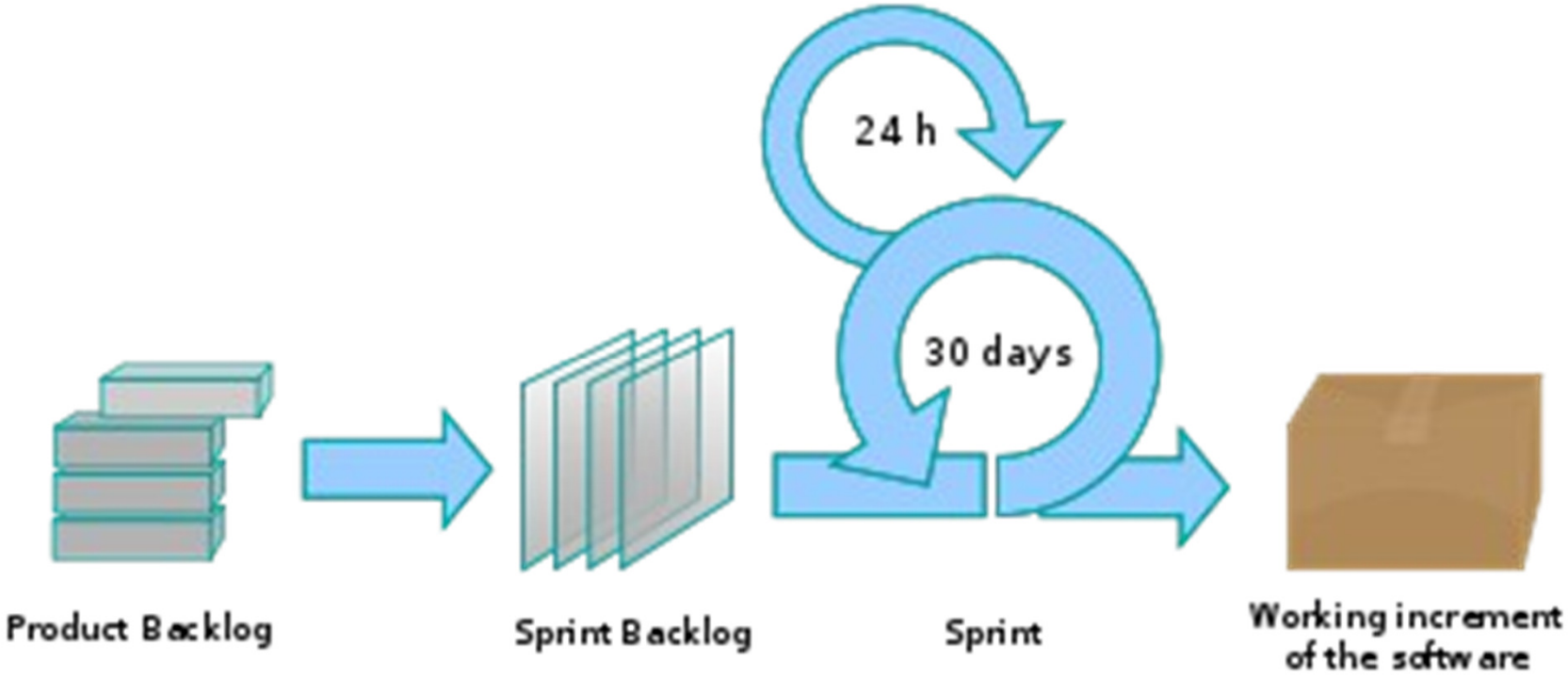


Image source: https://commons.wikimedia.org/wiki/File:Scrum_process.svg

Summary

- 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.
  - Non-Functional Requirements
    - to be taken serious
    - to be identified e.g. with ISO / IEC 9126 as check list.

Summary

- There are a lot of techniques, “how-to”, and ideas to identify, to write, and to update requirements.
- A constructive, willing to learn organization is extremely helpful for successful requirements engineering.

Want to learn more?

- Professional organizations
  - International Requirements Engineering Board, [IRE16]
    → 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
- Example for a specification: "The project Aardvark Spec" @ http://www.joelonsoftware.com/articles/AardvarkSpec.html

Sources

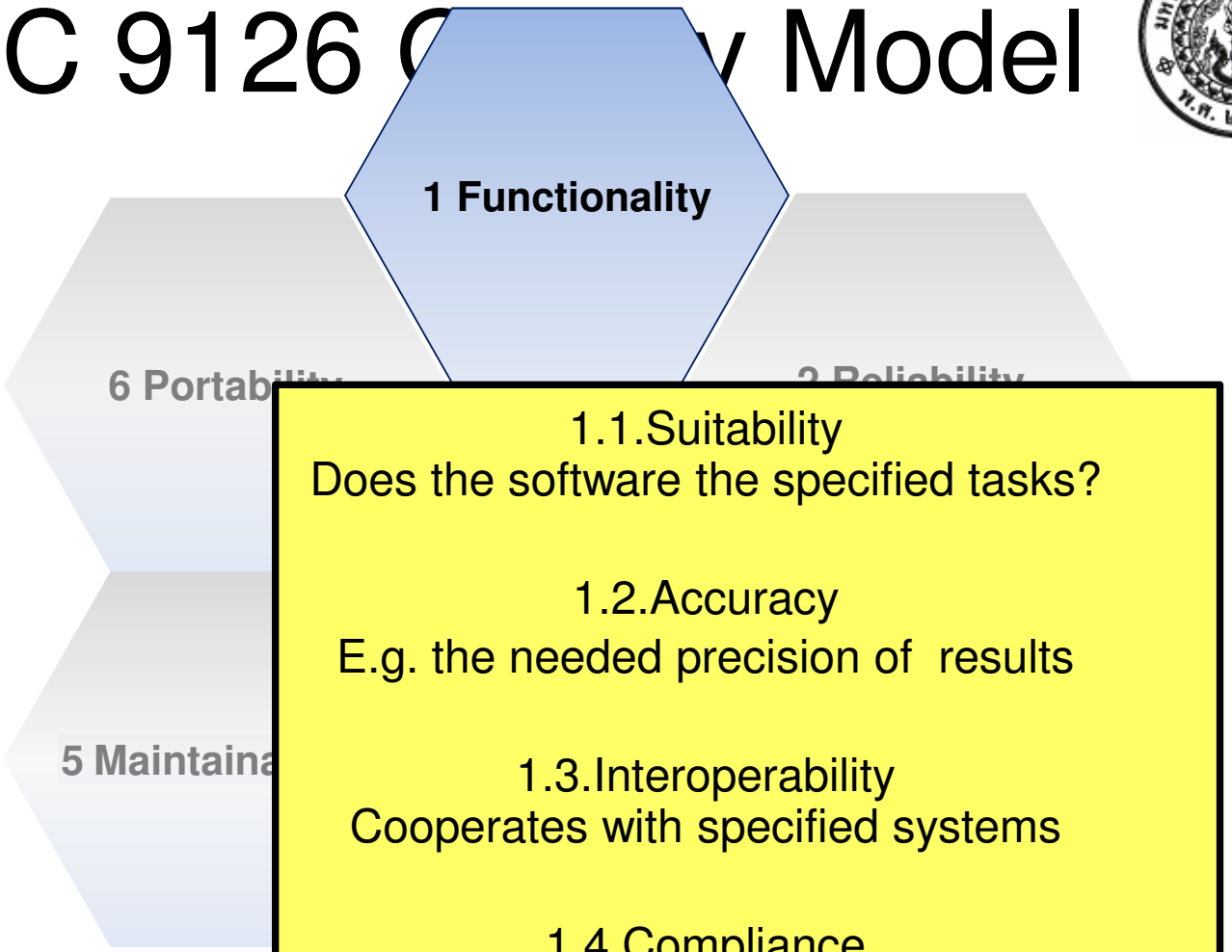
- [ADA15] Application Developers Alliance, Developer Insights Report, August 2015, http://www.appdevelopersalliance.org/developer-insights-report-2015
- [AG16] Daud Alam, Uwe Gühl: Projektmanagement für die Praxis, Springer, 2016 (in German)
- [Gli14] Martin Glinz, A Glossary of Requirements Engineering Terminology, Version 1.6, May 2014, https://www.ireb.org/en/downloads/
- [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
- [Mod16] MODERNanalyst.com: The Requirements Engineer Role; http://www.modernanalyst.com/TheProfession/Roles/RequirementsEngineer/tabid/188/Default.aspx
- [Mou16] Mountain Goat: User Stories; An Agile Requirements Approach, 2016, http://www.mountaingoatsoftware.com/agile/user-stories

Sources

- [PR11], Klaus Pohl, Chris Rupp: Basiswissen Requirements Engineering. Aus- und Weiterbildung zum ”Certified Professional for Requirements Engineering”, 3rd edition, 2011, dpunkt.verlag
- [Ric05] Randall W. Rice: STBC The Economics of Testing, 2005, http://www.riceconsulting.com/public_pdf/STBC-WM.pdf
- [Sch01] Bruno Schienmann: Kontinuierliches Anforderungsmanagement, Prozesse – Techniken – Werkzeuge, Addison-Wesley, 2001
- [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
- [Wik16] Wikipedia: ISO/IEC 9126, 2016, https://en.wikipedia.org/wiki/ISO/IEC_9126
- [Wik16a] Wikipedia: Use Case, 2016, https://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

Backup

ISO/IEC 9126 Q y Model



1.1.Suitability
Does the software the specified tasks?

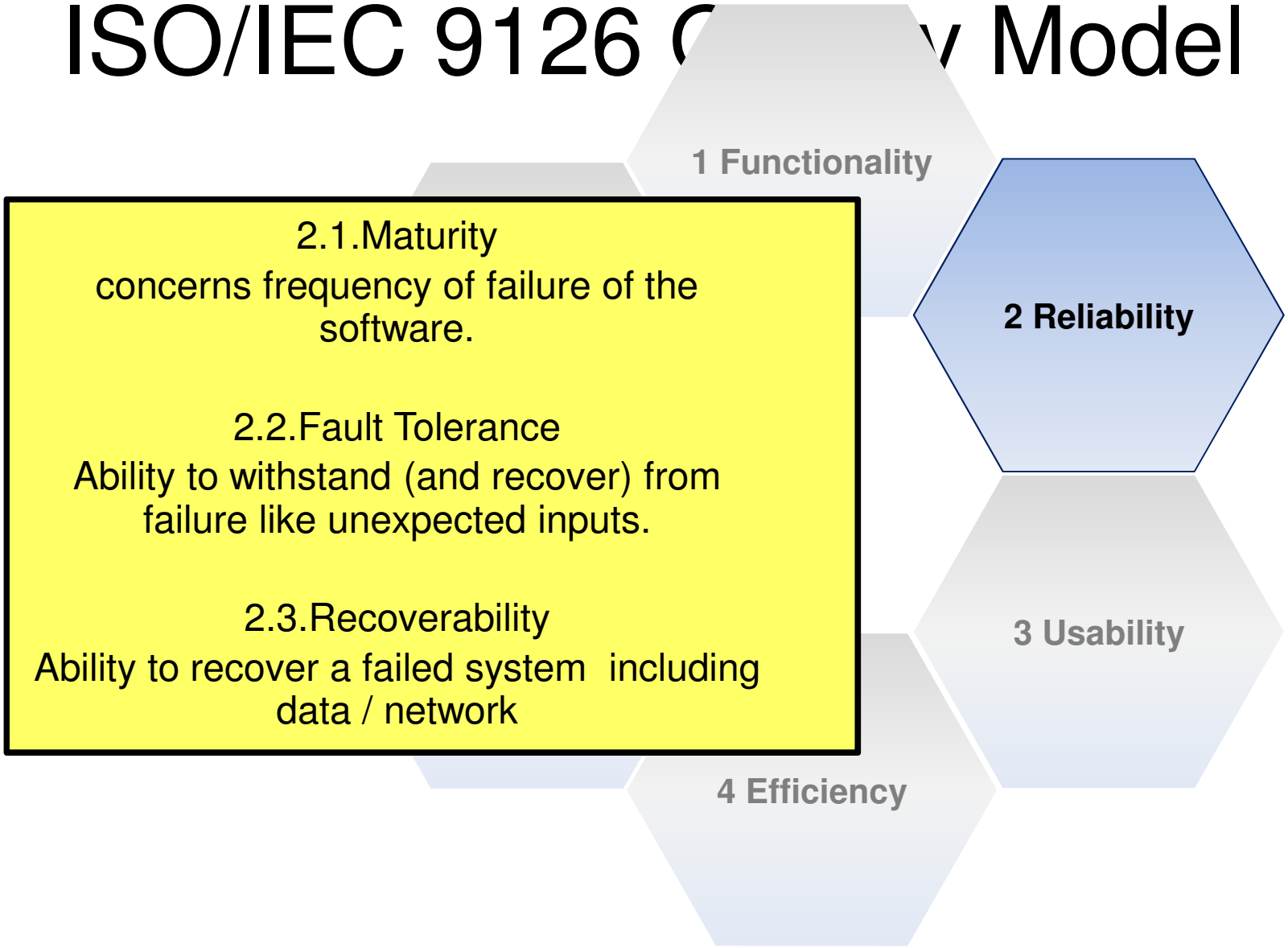
1.2.Accuracy
E.g. the needed precision of results

1.3.Interoperability
Cooperates with specified systems

1.4.Compliance
...with conditions / regulations

1.5.Security
No unauthorized access possible

ISO/IEC 9126 Q y Model

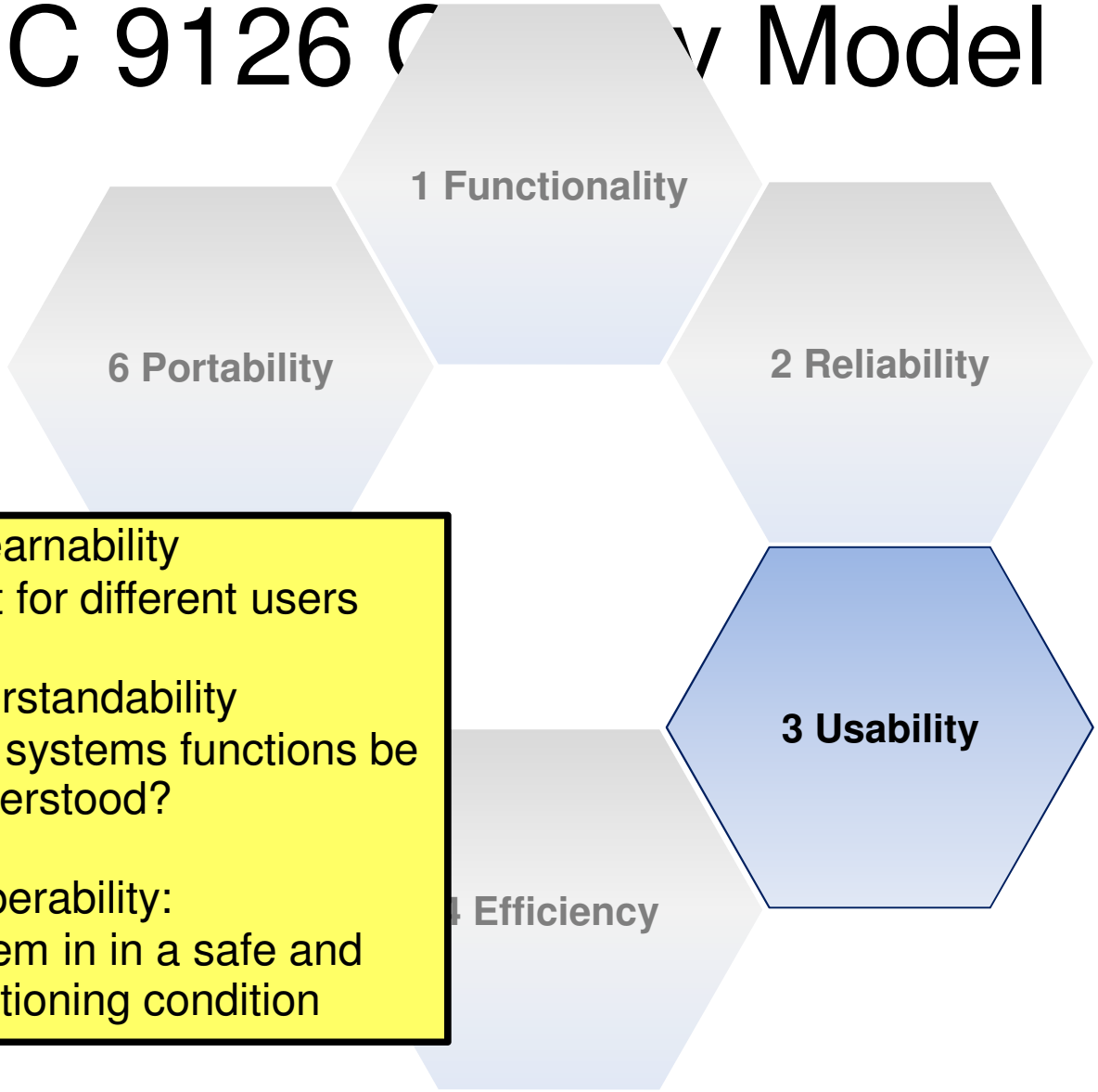


2.1.Maturity
concerns frequency of failure of the software.

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 Q y Model

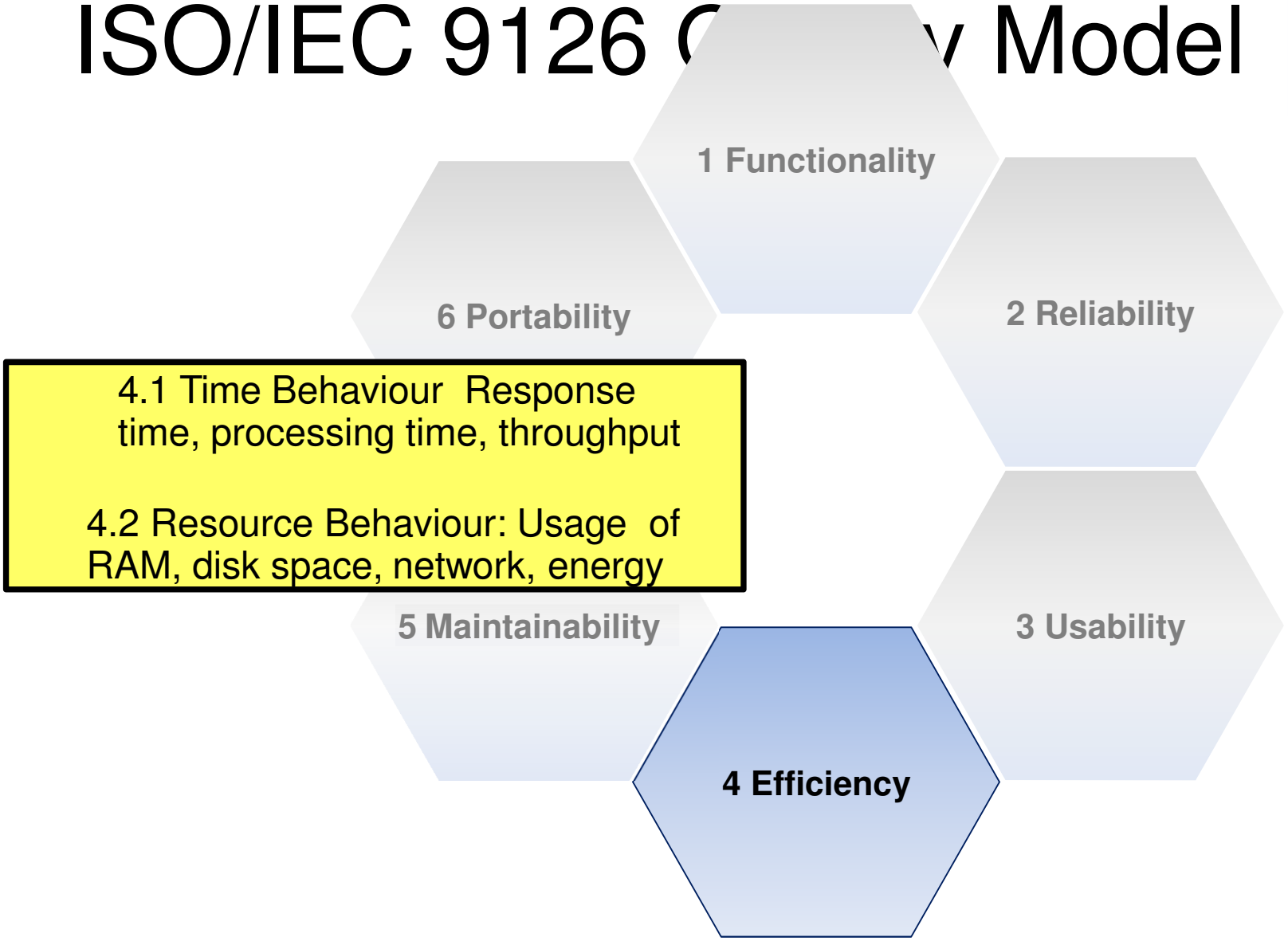


3.1.Learnability
Learning effort for different users

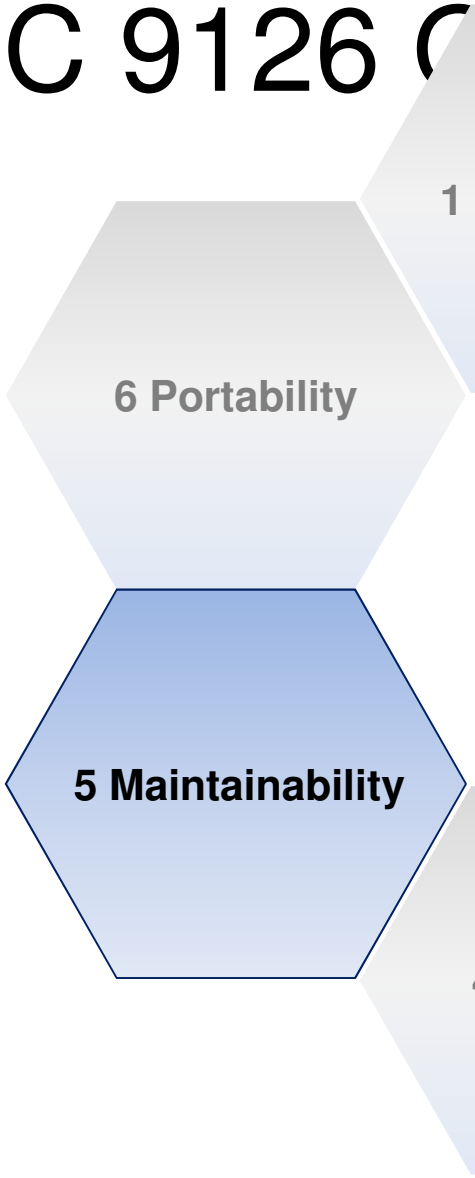
3.2.Understandability
How easy could systems functions be understood?

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

ISO/IEC 9126 Quality Model



ISO/IEC 9126 Quality Model



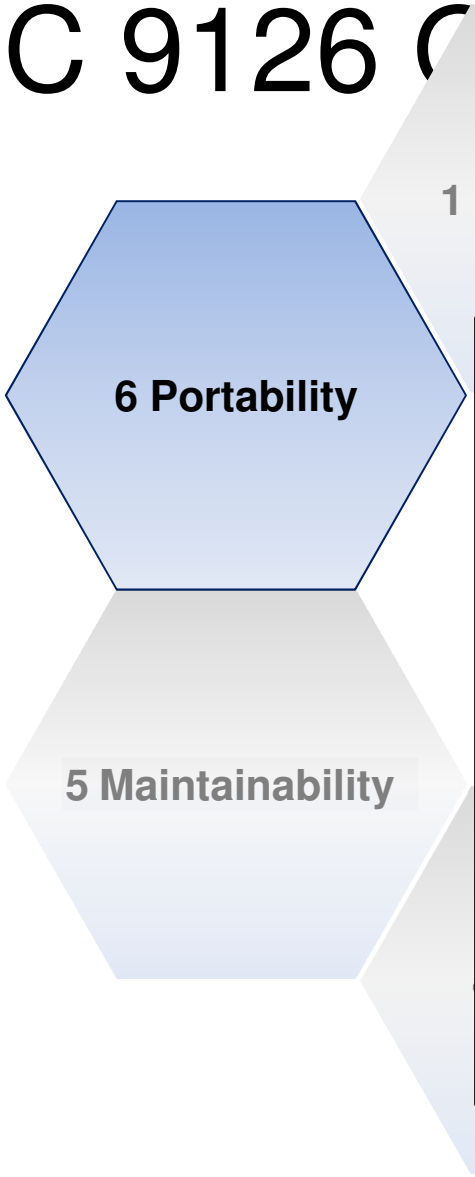
5.1.Stability:
Capability to avoid unexpected effects from modifications of the system

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