Requirements Engineering

FEUP-MIEIC-ESOF-2019-20

João Pascoal Faria, Ademar Aguiar

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

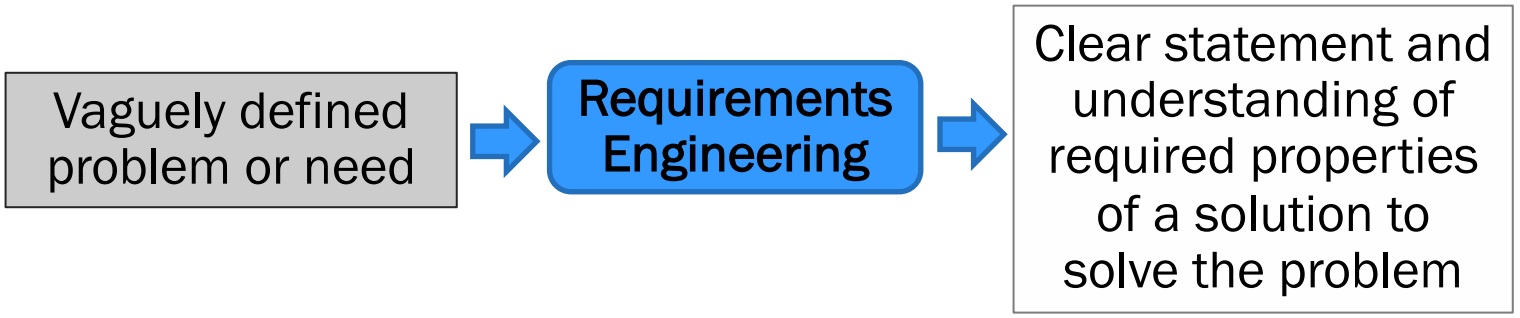
Contents

- Scope, importance and challenges of requirements engineering
- Classification of software requirements
- Requirements engineering in some well-known processes
- Requirements engineering process
- Requirements elicitation techniques

SCOPE, IMPORTANCE AND CHALLENGES OF REQUIREMENTS ENGINEERING

What is requirements engineering?

Definitions



- **Requirements engineering (RE):** the process of studying **customer** and **user** *needs* to arrive at a definition of **system**, hardware, or **software** *requirements*.
  [adapted from IEEE Std Glossary of Soft. Eng. Terminology IEEE Std 610.12-1990]

- **Software requirement:** a property which must be exhibited by software developed or adapted to solve a particular problem. [Guide to the Software Engineering Body of Knowledge (SWEBOK)]

Example

(Business) Needs:

- Need an ICT-based solution to reduce road accidents under reduced visibility conditions

requirements engineering activities

(System) Requirements:

- The system shall be based on special purpose devices installed in each vehicle
- The device shall monitor hazardous events, such as airbag inflation event
- The device shall broadcast corresponding geo-referenced radio alerts
- The device shall receive radio alerts sent from nearby vehicles and alert the driver as appropriate
- (…)

Importance of RE: project success factors

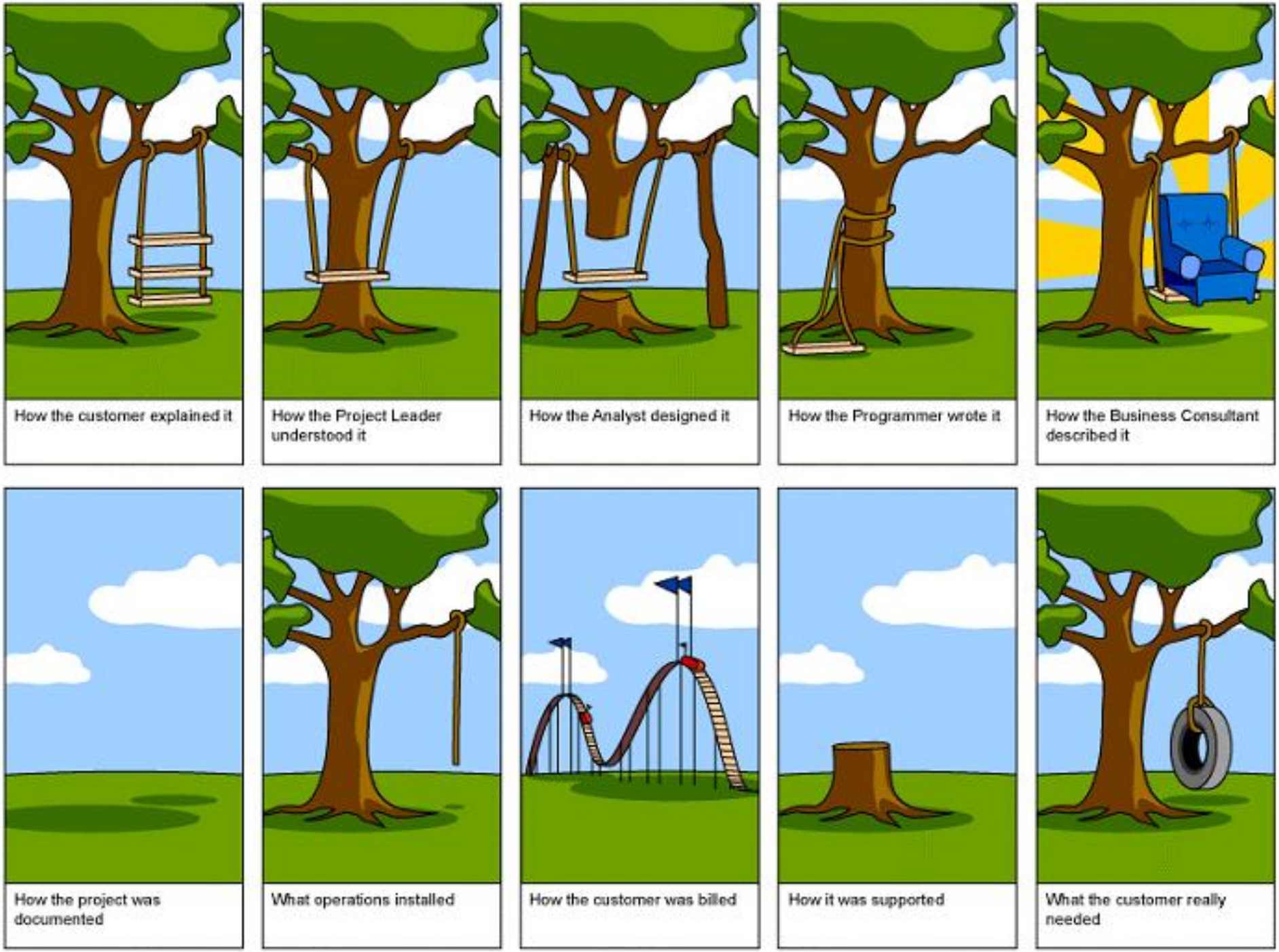
1	User Involvement	16 %
2	Executive Management Support	14 %
3	Clear Statement of Requirements	13 %
4	Proper Planning	10 %
5	Realistic Expectations (scope, budget, schedule,...)	8 %
6	Smaller Project Milestones	8 %
7	Competent Staff	7 %
8	Ownership	5 %
9	Clear Vision & Objectives	3 %
10	Hard-working, Focused Staff	2 %
11	Other	14 %

(source: Standish group CHAOS report)

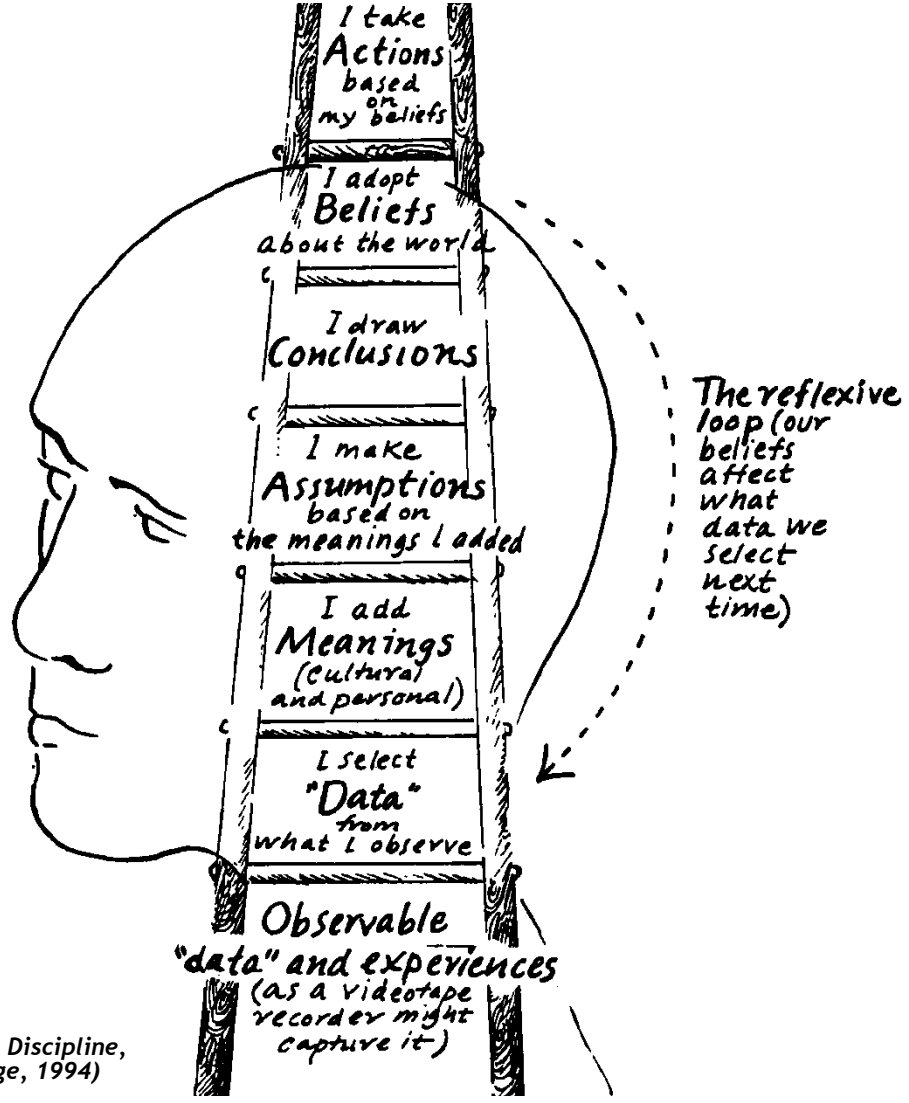
Main problems of RE

- Requirements (mis)communication & (mis)understanding
- Evolving requirements:
  - Changing
  - Growing
- Requirements creep: uncontrolled changes or continuous growth in a project's requirements

"I think you misunderstood me when I said I wanted our factory to go all green."


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

The ladder of inference: how our assumptions can cause miscommunication



(The Fifth Discipline, Peter Senge, 1994)

You should never assume ...

http://www.youtube.com/watch?v=R6jaKkE0RsI

CLASSIFICATION OF SOFTWARE REQUIREMENTS

Levels of requirements

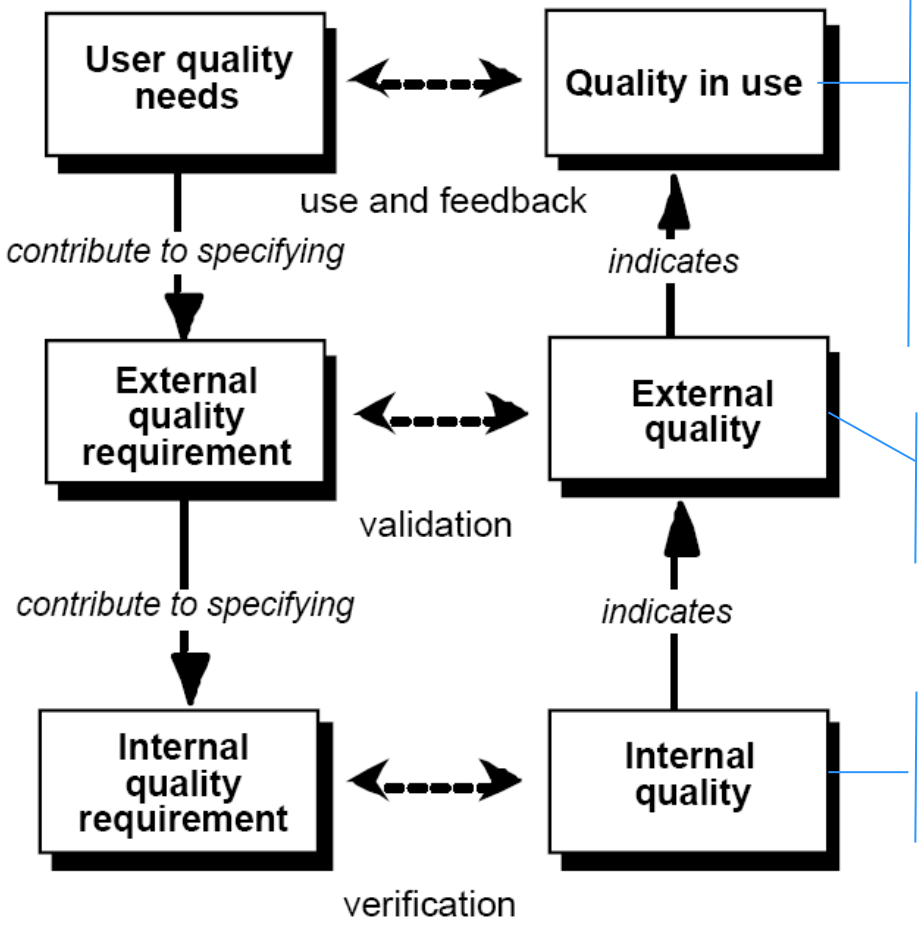
- **Business requirements/needs** represent high-level objectives of the organization or **customer** who requests the system
  - Can be captured in vision and scope document
- **User requirements/needs** describe user goals or tasks that the **user** must be able to perform with the product
  - Can be captured in use case models
- **System requirements** are the requirements for the system as a whole (which frequently include hardware & software components)
  - Can be captured in a system requirements specification document
- **Software requirements** are derived from system requirements (by allocating system req.'s to software components and detailing them)
  - Can be captured in a software requirements specification (SRS) document

Types of requirements

- Functional requirements describe the functions that the software is to execute
  - Also known as capabilities
  - Example: The system shall send an e-mail notification to the customer when the items he/she ordered are dispatched
- Nonfunctional requirements are the ones that act to constrain the solution
  - Most of them are quality requirements
    - Can be defined based on quality characteristic and quality metrics (see next)
    - Example: The maximum system down-time should be 8 hours per year
  - Can also include development process requirements
    - Example: The product should be developed in Java (requirement source: maintenance company)

ISO/IEC 25010: Quality model framework

- Distinguishes three views of software product quality:

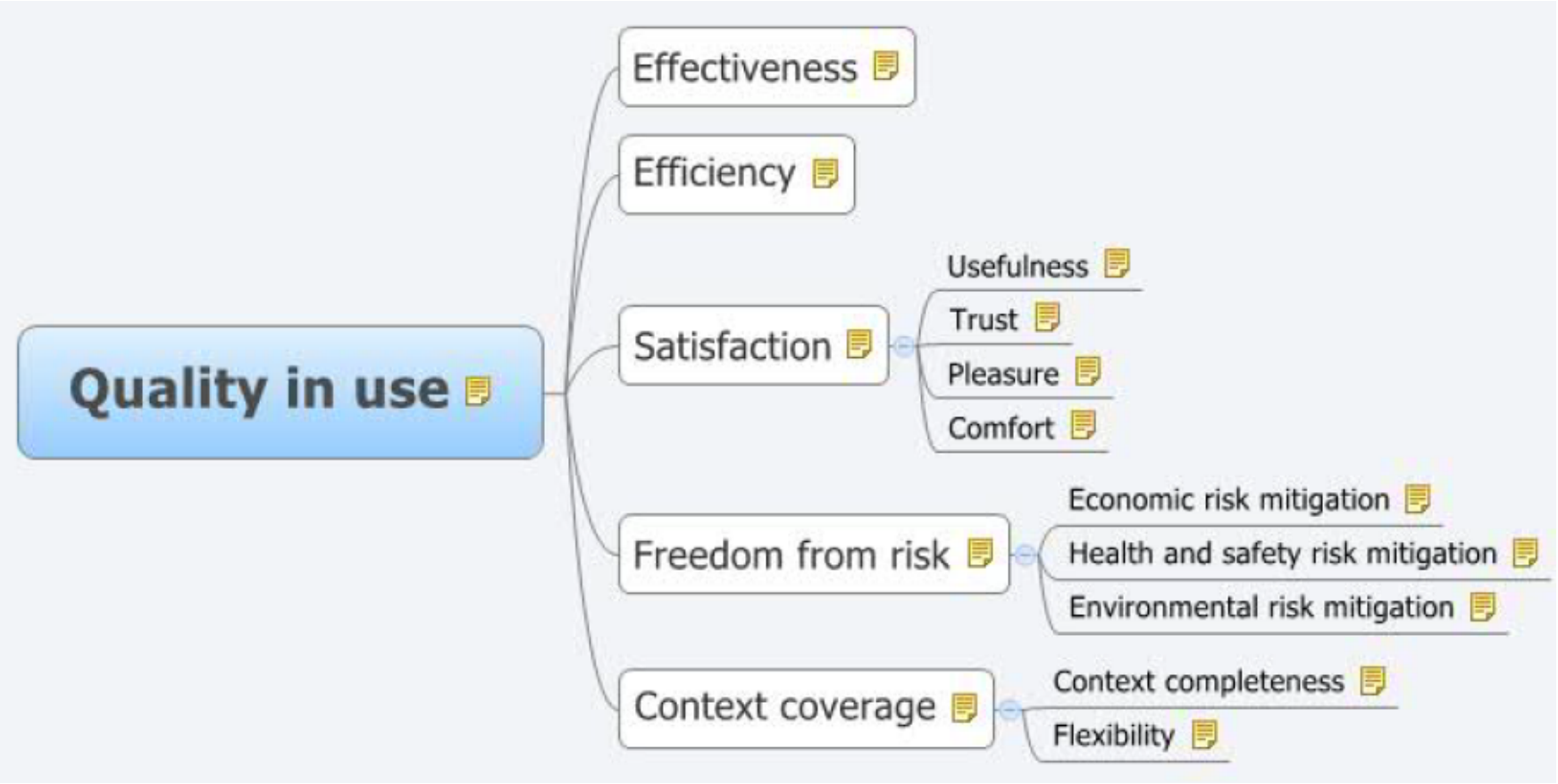


User's view of the quality of the software product when used in a specific environment and context. It measures the extent to which users can achieve their goals in that environment, rather than measuring the properties of the software itself.

Totality of characteristics of the software product from an external view during its execution.

Totality of characteristics of the software product from an internal view during its development or maintenance.

ISO/IEC 25010: Quality in use characteristics and sub-characteristics



ISO/IEC 25010: Product quality characteristics and sub-characteristics

*ISO/IEC 25010: Product quality characteristics

- Functionality suitability - degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions
- Performance efficiency - performance relative to the amount of resources used under stated conditions
- Reliability - degree to which a system, product or component performs specified functions under specified conditions for a specified period of time
- Usability - degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use
- Compatibility - degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment
- Maintainability - degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers
- Portability - degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another
- Security - degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization

http://pef.czu.cz/~papik/doc/MHJS/pdf/ISOIEC_FDIS25010_(E).pdf

REQUIREMENTS ENGINEERING IN SOME WELL-KNOWN PROCESSES

Waterfall

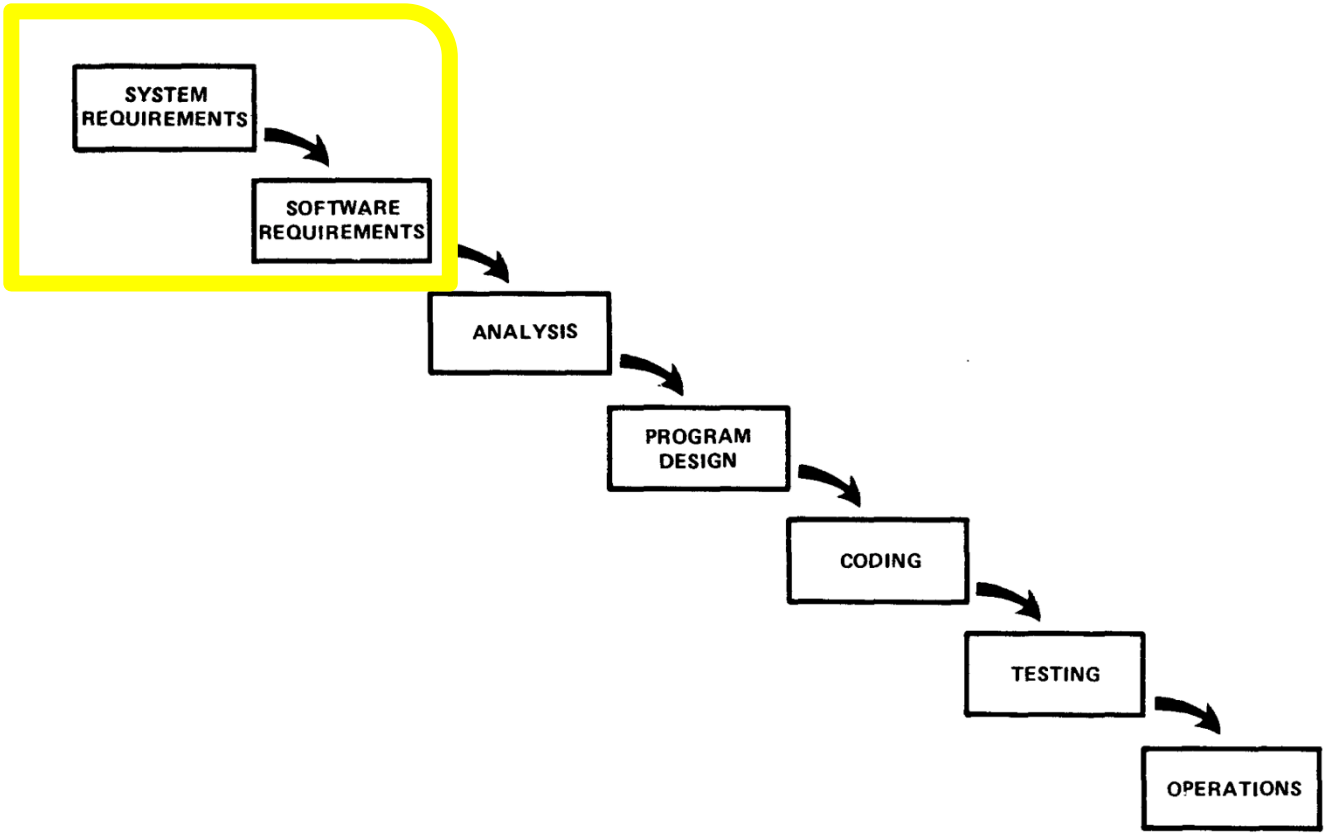
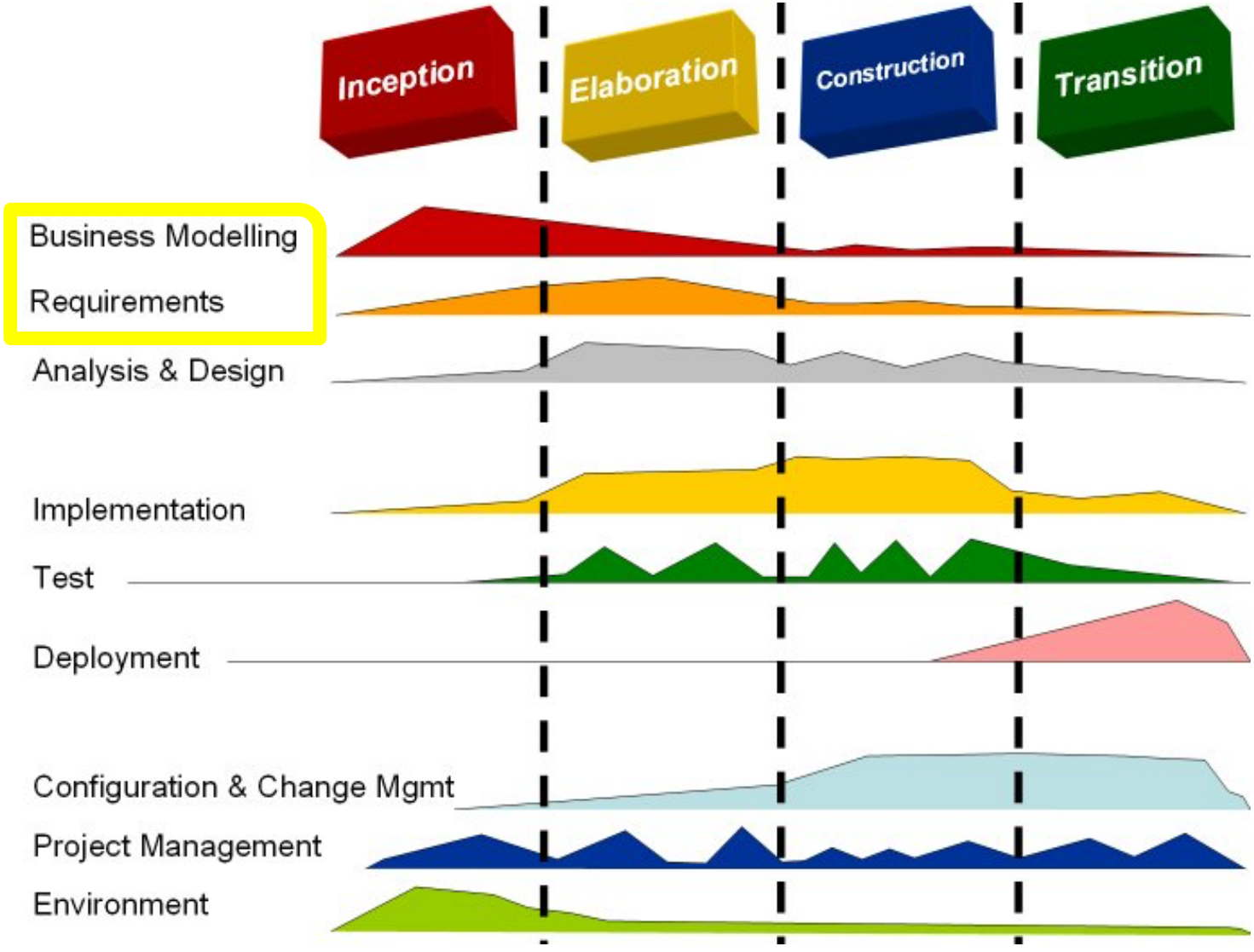
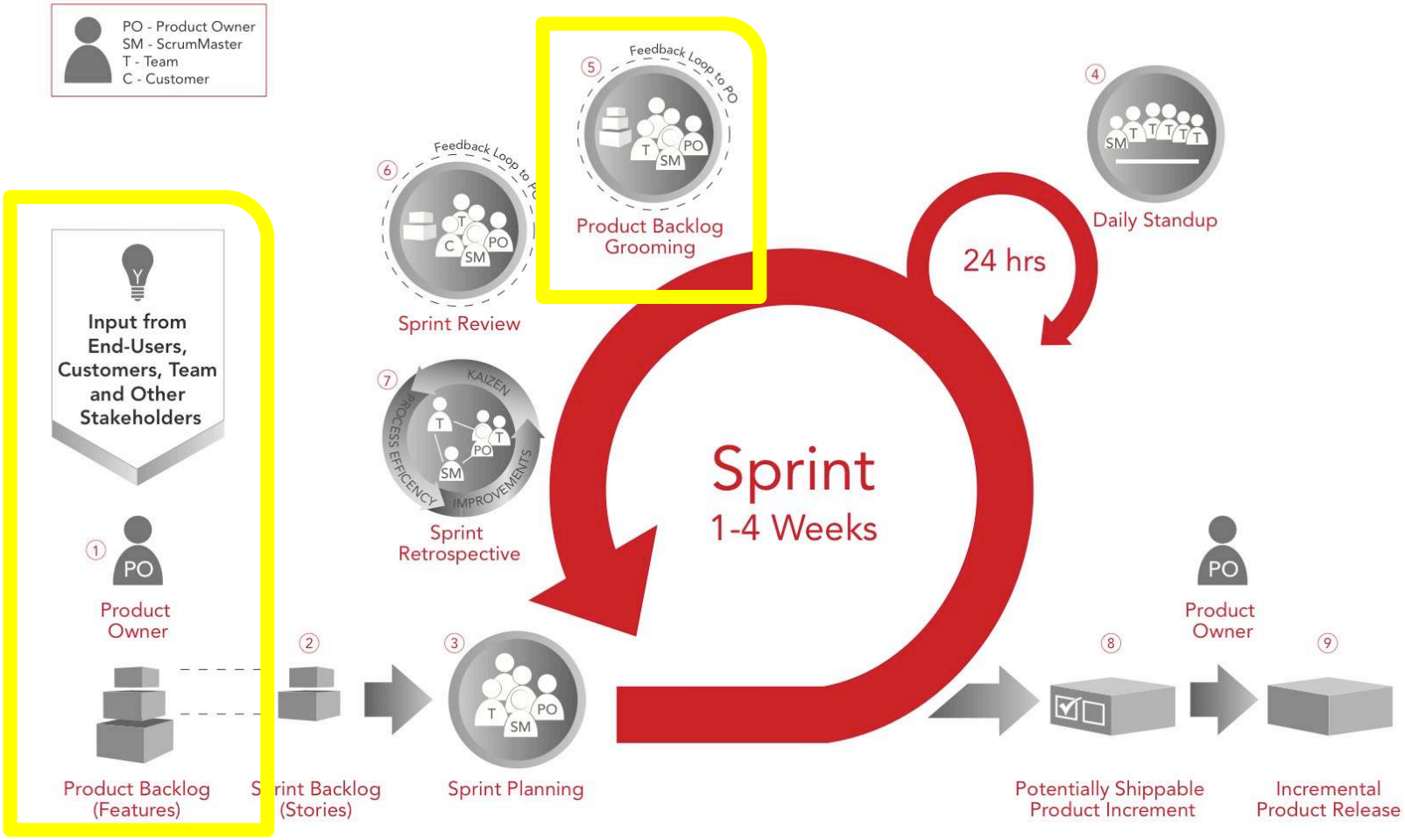


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

Rational Unified Process

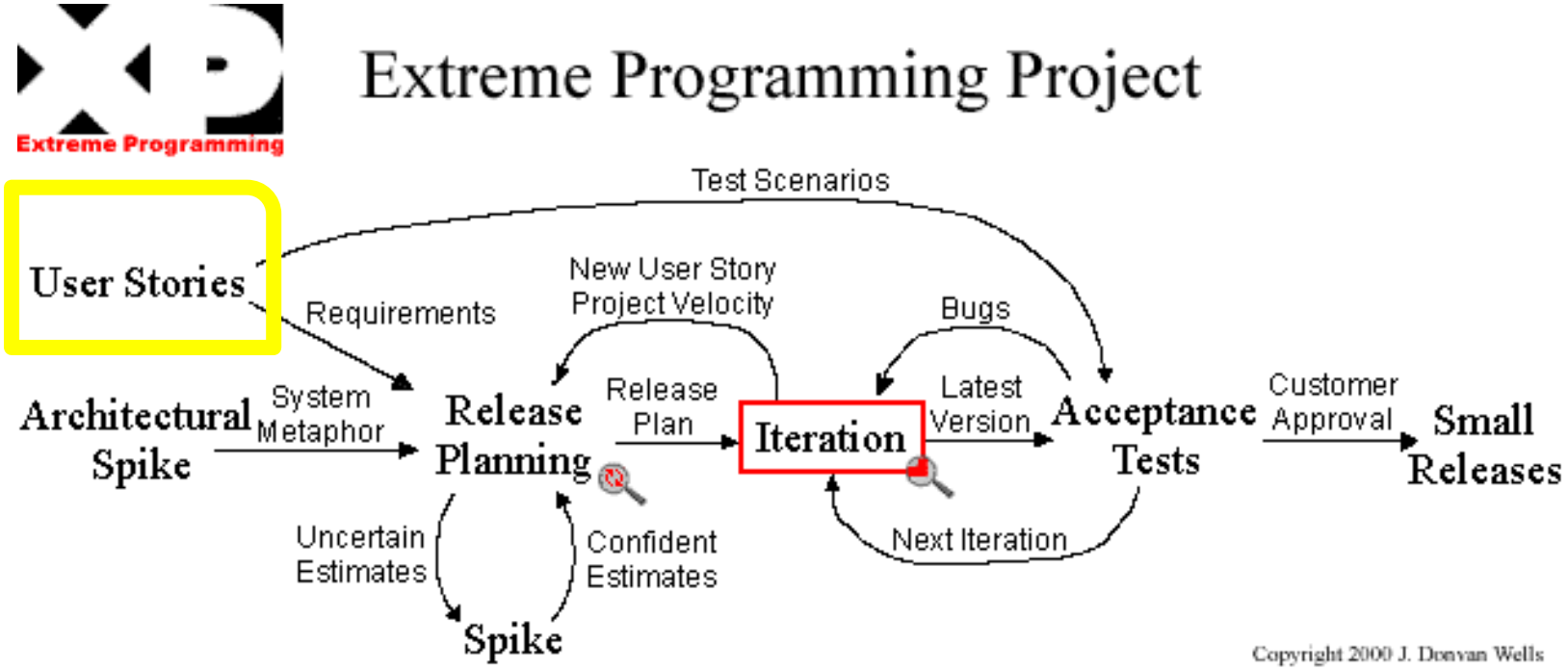


Scrum



scruminc.

XP


Extreme Programming
Extreme Programming Project
Test Scenarios
User Stories
Requirements
New User Story
Project Velocity
Bugs
Architectural Spike
System Metaphor
Release Planning
Release Plan
Iteration
Latest Version
Acceptance Tests
Customer Approval
Small Releases
Uncertain Estimates
Confident Estimates
Spike
Next Iteration
Copyright 2000 J. Donvan Wells

Agile methods and requirements

- Many agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly.
- Requirements documents are therefore always out of date.
- Agile methods usually use incremental requirements engineering and may express requirements as user stories.
- This is practical for business systems but may be problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

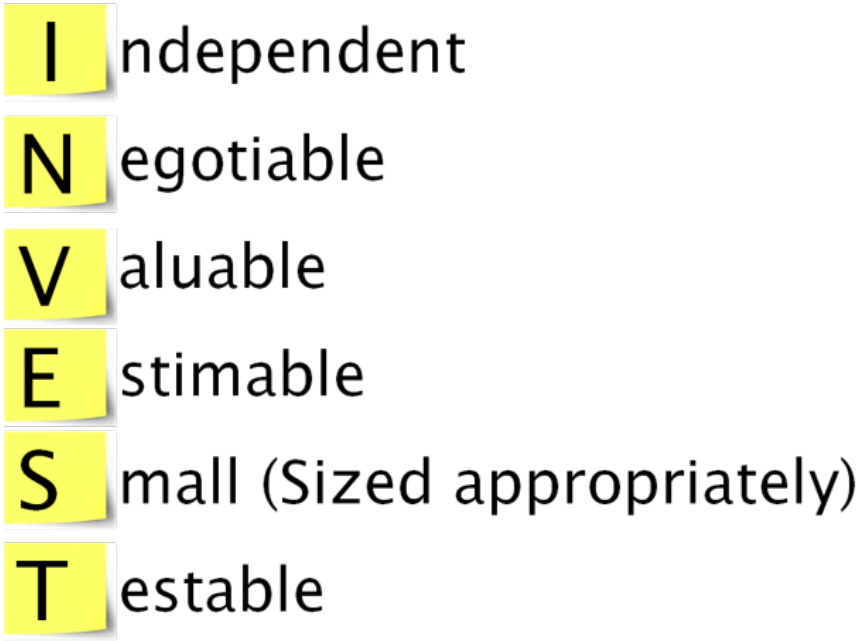
User Stories

"*A user story is a promise for a conversation*"

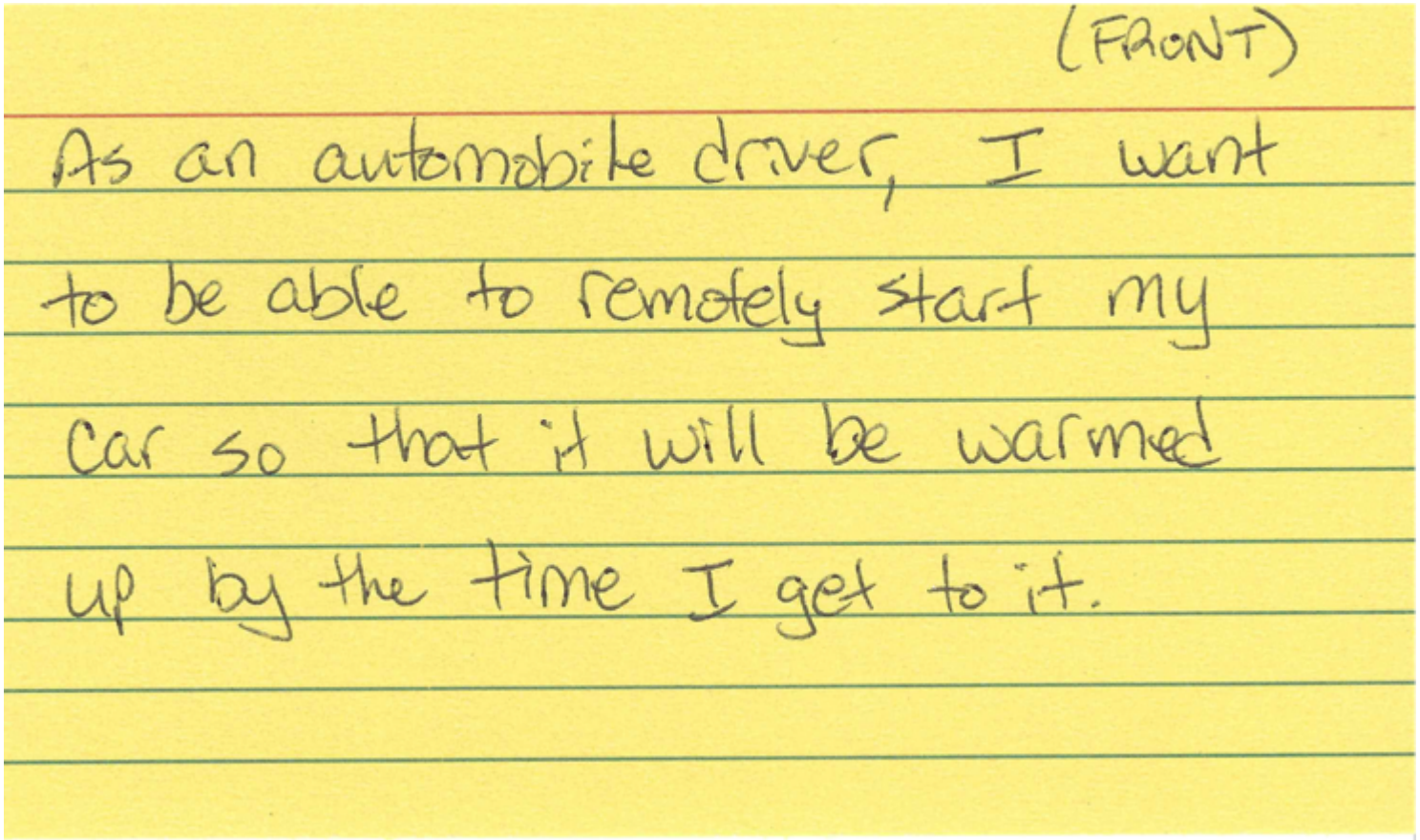
Alistair Cockburn

- Lightweight way to record a software need, with just enough information (for prioritization, planning, and initiating conversations).
- Should include: ***Who***, ***What*** and ***Why***
- It's good to **INVEST...** (next slide)

User Stories - INVEST

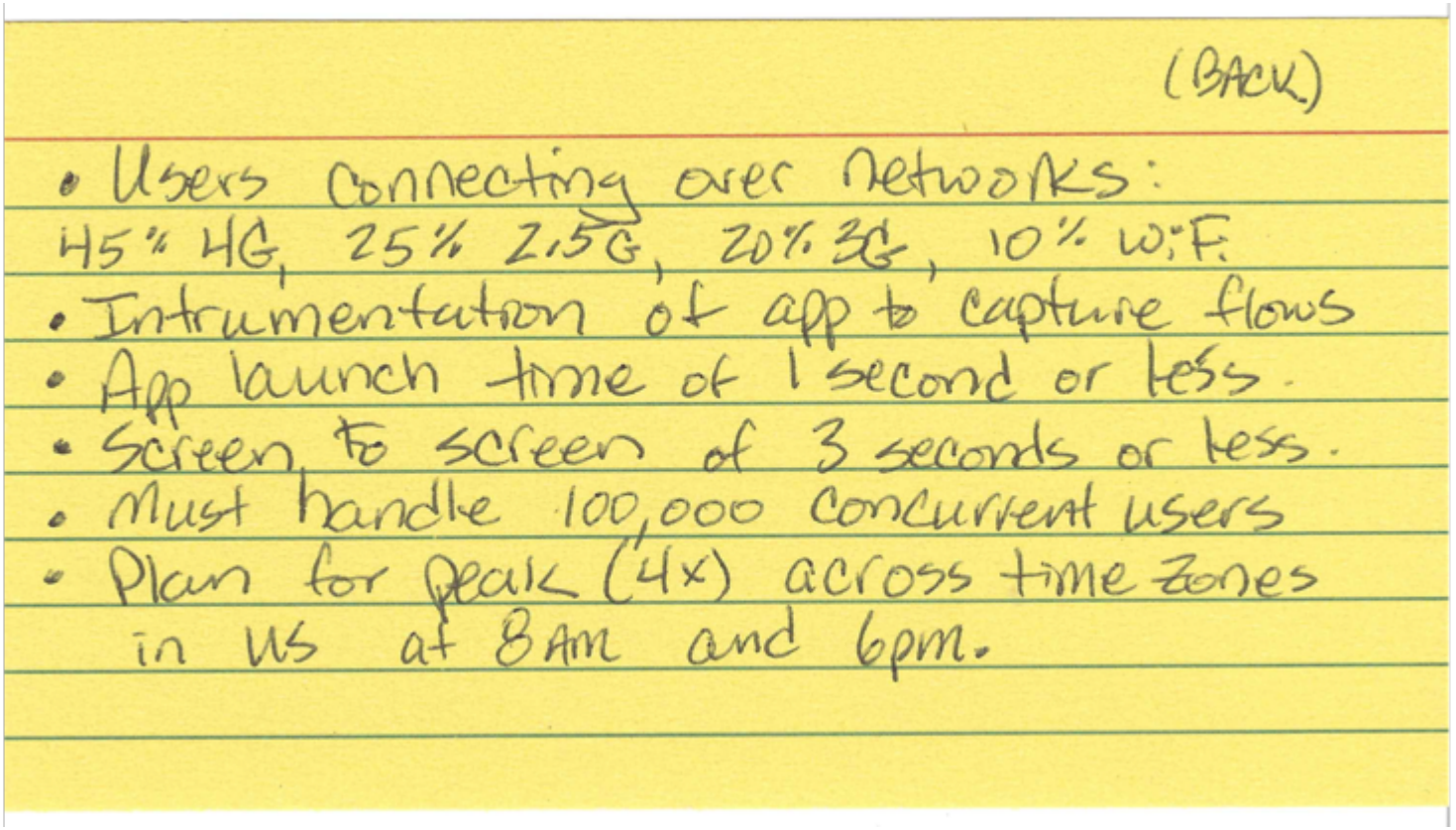


User Stories – Example



Adapted from https://techbeacon.com/how-build-performance-your-user-stories

User Stories – Example



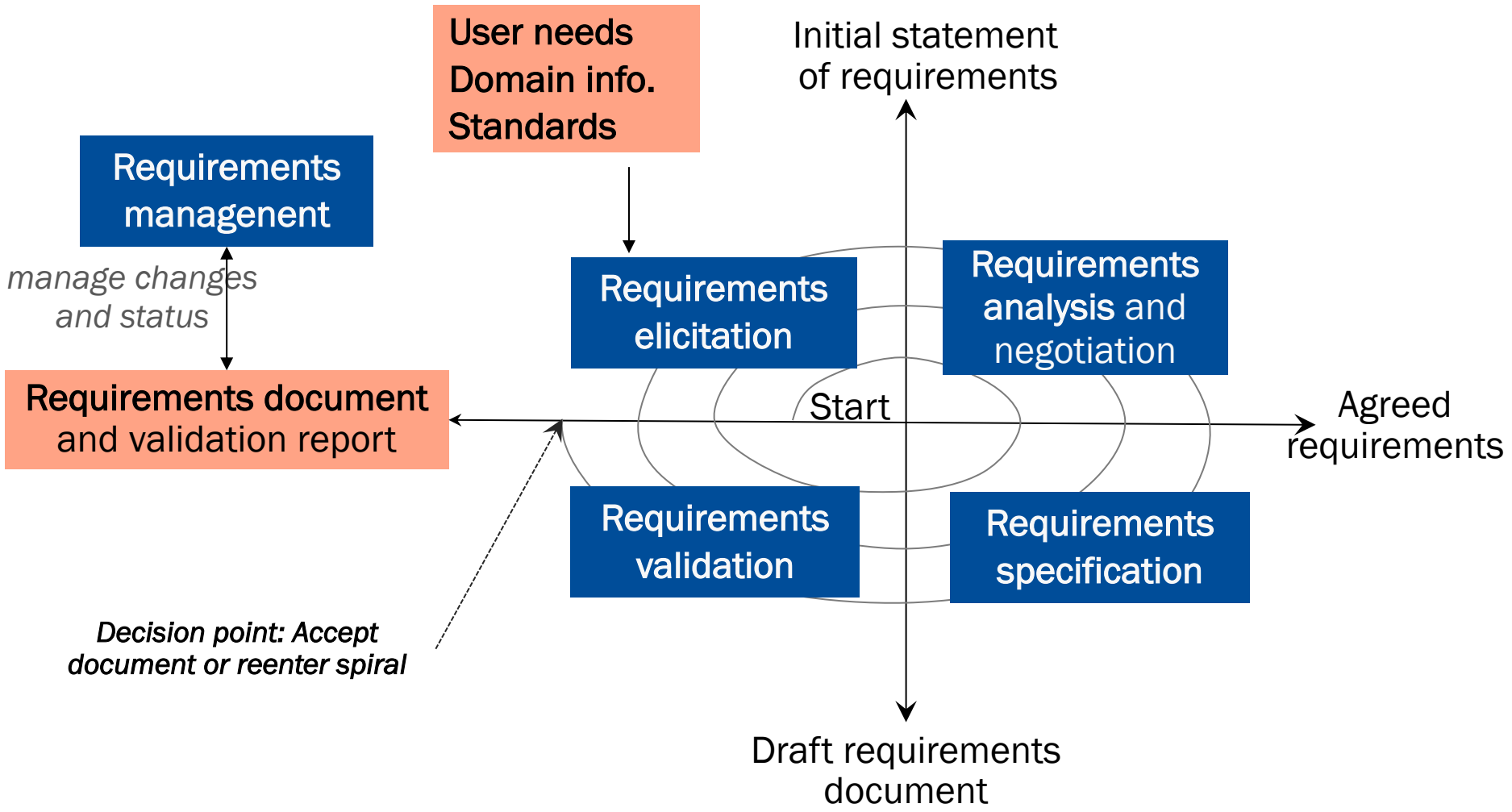
Adapted from https://techbeacon.com/how-build-performance-your-user-stories

References and further reading

- Software Engineering, Ian Sommerville, 9th Edition (chap.6-7)
- Guide to the Software Engineering Body of Knowledge (SWEBOK), 2004 edition, IEEE Computer Society
- IEEE Std 610.12: 1990 - Standard Glossary of Software Engineering Terminology
- ISO/IEC 12207 - Information technology - Software life cycle processes
- ISO/IEC 25000 family of standards - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)
- “Requirements Engineering as a Success Factor in Software Projects”, Hubert F. Hofmann, Franz Lehner, IEEE Software 2001
- Extreme Programming Explained”, Kent Beck and Cynthia Andres, 2nd ed. Addison Wesley, 2004

REQUIREMENTS ENGINEERING PROCESS

Requirements enginering process



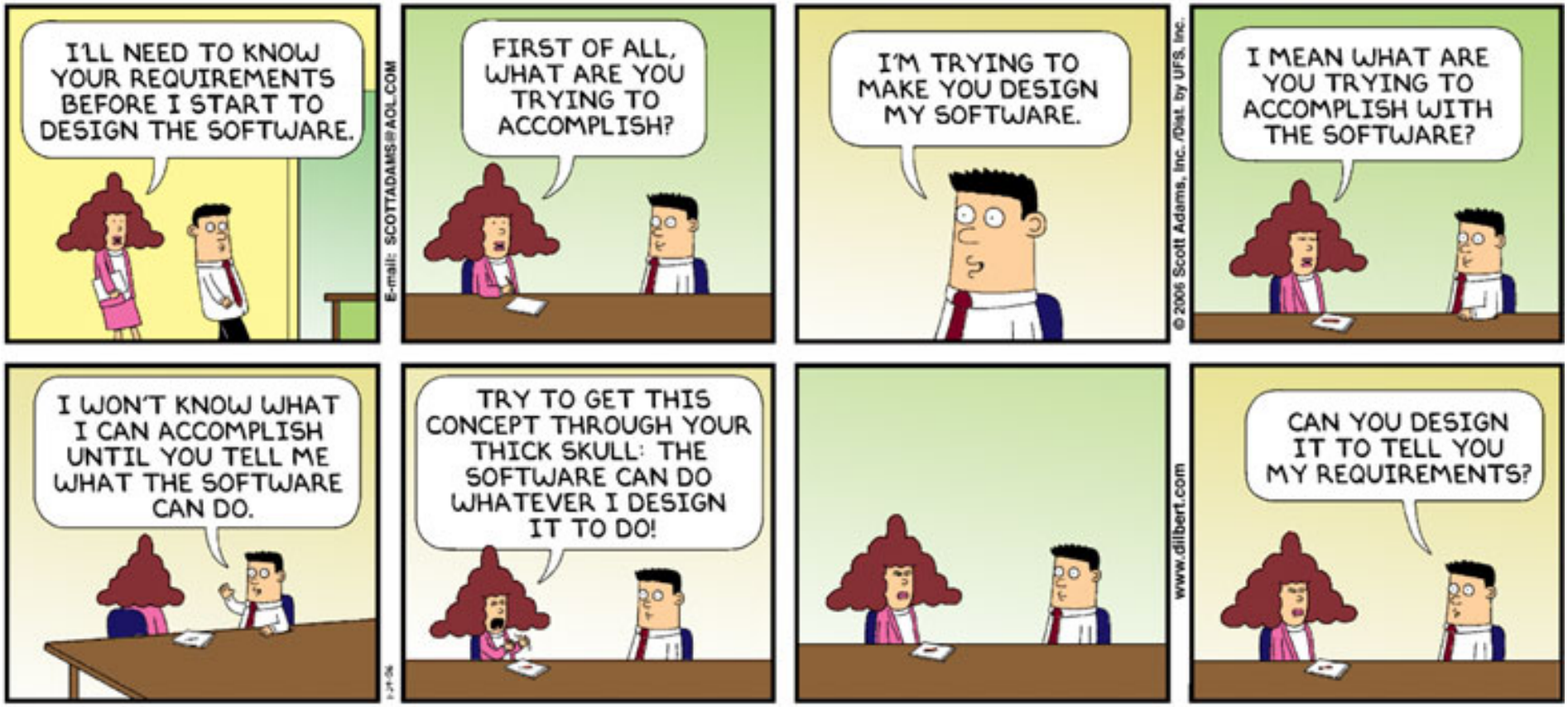
Requirements elicitation (or discovery)

- Goals:
  - Interact with stakeholders and other sources (documents, existing systems, etc.) to collect/discover their requirements
- Techniques:
  - Interviews
  - Facilitated meetings (brainstorming, focus groups, etc.)
  - Questionnaires
  - Goal analyses
  - Social observation and analysis (of how people actually work)
  - (User-interface) Prototypes
  - Scenarios, user stories, use cases (real-life usage examples)
  - Social media analysis (including customer reviews)

Stakeholders

- People who will be affected by the system and who have a direct or indirect influence on the elaboration of requirements:
  - Customers
  - End users
  - Managers and others involved in organizational processes influenced by the system
  - People responsible for maintaining the system
  - Clients of the organization that may use the system
  - Regulatory and certification bodies, etc.
- Example in an automatic railway signaling system:
  - system operators, train conductors, managers, passengers, installation and maintenance engineers, certification and security authorities

Requirements elicitation



Requirements analysis (& negotiation)

- Goals:
  - Detect and resolve problems with the requirements (conflicts, omissions, ambiguity, etc.)
  - Group related requirements and organize thei clusters
  - Arrive at a list of agreed requirements

- Techniques
  - Checklists – helps discovering recurring probl
  - Modeling – formalization helps discovering or inconsistencies, etc.
  - Requirements classification and prioritization

Requirements analysis checklist

Characteristic	What to consider
Completeness	Is anything missing or forgotten? Is it thorough? Does it include everything necessary to make it stand alone?
Consistency	Are there any requirements conflicts?
Unambiguity	Is there a single interpretation? Or is it vague?
Verifiability	Is enough information provided to create acceptance tests to check requirements implementation?
Necessity	Is the feature within the system scope? Is it traceable to an original customer need? Does the specification stick with defining the product and not the underlying design?
Feasibility (Realism)	Can the features be implemented with the available technology & budget?

Requirements specification

- Produce a Software Requirements Specification (SRS) document
- Often accompanied by other artifacts, such as:
  - Other documents
    - Preliminary user manual (sometimes created instead of the SRS!)
    - Glossary (business and technical terms)
  - Tables and matrices
    - Requirements attributes tables (with priority, source, etc.)
    - Traceability matrices (requirements to user needs, req.s to test cases, etc.)
  - Prototypes
    - User-interface prototypes (may be embedded in preliminary user manual)
  - Models
    - Use case models + Domain models (in UML), or
    - Data-flow diagrams (DFD's) + Entity-relationship diagrams (ERD's), or
    - Formal models (for critical systems)

Requirements specification template (1)

Table of Contents

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Definitions, acronyms, and abbreviations
   - 1.4 References
   - 1.5 Overview
2. Overall description
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User characteristics
   - 2.4 Constraints
   - 2.5 Assumptions and dependencies
3. Specific requirements (See 5.3.1 through 5.3.8 for explanations of possible specific requirements. See also Annex A for several different ways of organizing this section of the SRS.)

Appendixes

Index

Figure 1—Prototype SRS outline

Source: IEEE Std 830-1998 - Recommended Practice for Software Requirements Specifications

Requirements specification template (2)

May be use cases →

3. Specific requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 System features
  - 3.2.1 System Feature 1
    - 3.2.1.1 Introduction/Purpose of feature
    - 3.2.1.2 Stimulus/Response sequence
    - 3.2.1.3 Associated functional requirements
      - 3.2.1.3.1 Functional requirement 1
      - .
      - .
      - .
      - 3.2.1.3.*n* Functional requirement *n*
  - 3.2.2 System feature 2
  - . . .
- 3.3 Performance requirements
- 3.4 Design constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Note: This template is useful for large systems engineering projects.

Example template of SRS Section 3 organized by features

Requirements validation

- Goals
  - Demonstrate that the requirements define the system that the customer really wants
- Motivation
  - Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error
- Techniques
  - Requirements reviews and inspections
  - Prototyping
  - Acceptance test case generation
  - Model validation

Requirements management

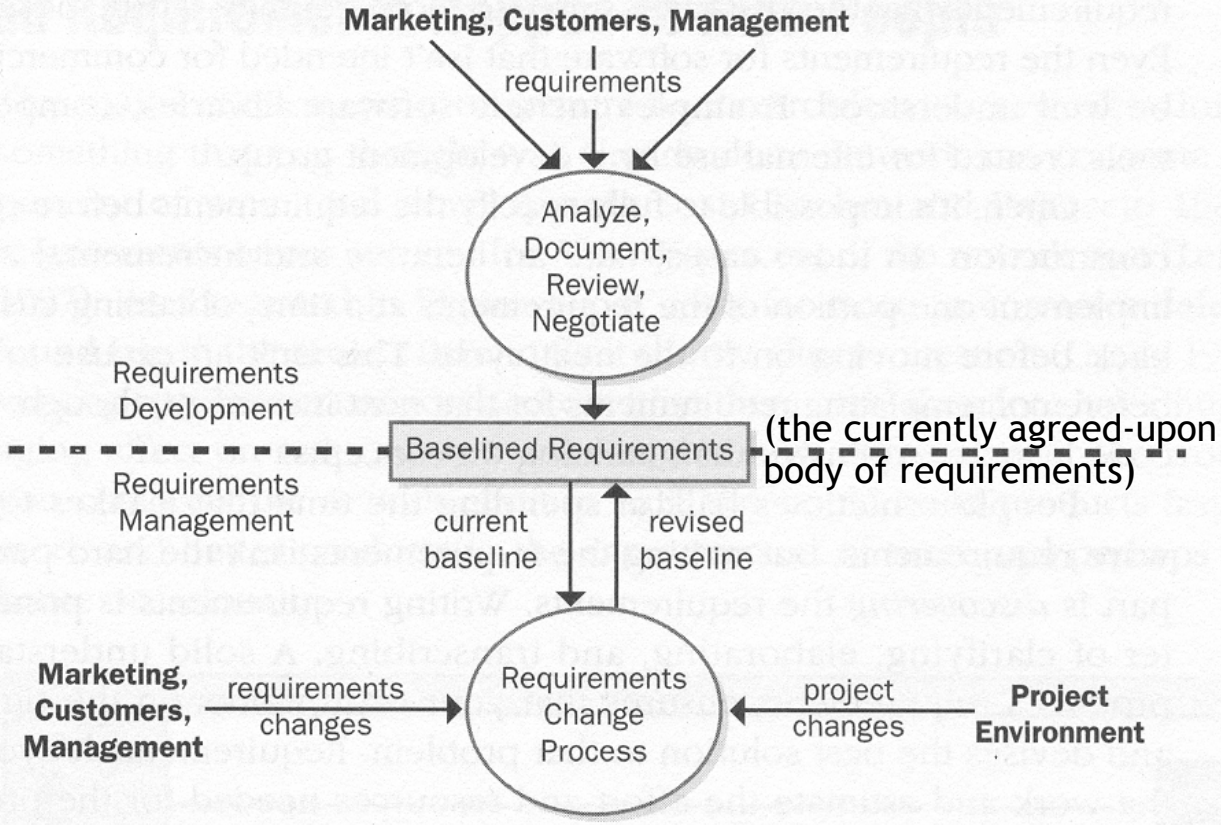


Figure 1-3 The boundary between requirements development and requirements management.

(Software Requirements, K.E.Wiegers)

REQUIREMENTS ELICITATION TECHNIQUES

Interviews

- Most widely used requirements elicitation technique
- Types of interviews:
  - Open/unstructured - various issues are explored with stakeholders
    - Better for initial exploration and for developing new/innovative requirements
  - Closed/structured - based on a pre-determined list of questions
    - Better for filling knowledge gaps (requires more preparation/background)
  - Mixed – most often in practice
- Both individual or group interviews are possible
- Activities involved:
  - Preparation – goals, participants, location, questions, background info
  - Execution – opening, questions, finalisation
  - Follow-up – analyse results, ask interviewees to confirm results

Brainstorming

- Useful to elicit new and innovative requirements
- Participants in requirements brainstorming sessions:
  - Moderator (usually a requirements analyst)
  - 4-8 people with different/multiple perspectives on the product
- Phases in brainstorming sessions:
  - Idea generation - participants are encouraged to come up with as many ideas as possible, without discussion of the merits of the ideas. Rules:
    - Quantity over quality: as many ideas as possible
    - Free association and visionary thinking are explicitly desired
    - Take on and combine expressed ideas
    - Do not criticize!
    - Questions for clarification of ideas
    - Do not abort the brainstorming at the first deadlock, make a short break
    - Let the brainstorming come to a natural end
  - Consolidation - ideas are discussed, revised, and organized.

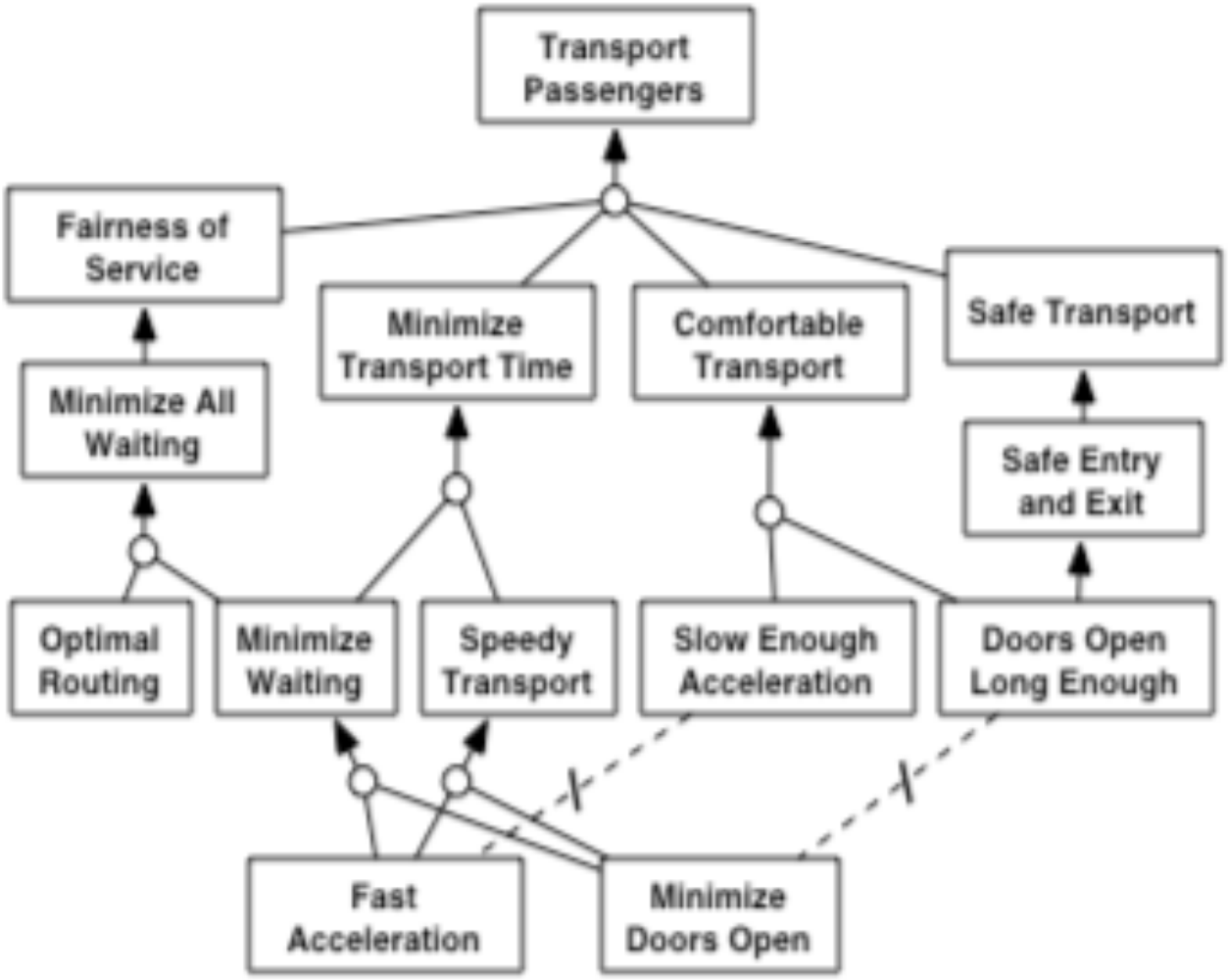
Questionnaires (surveys)

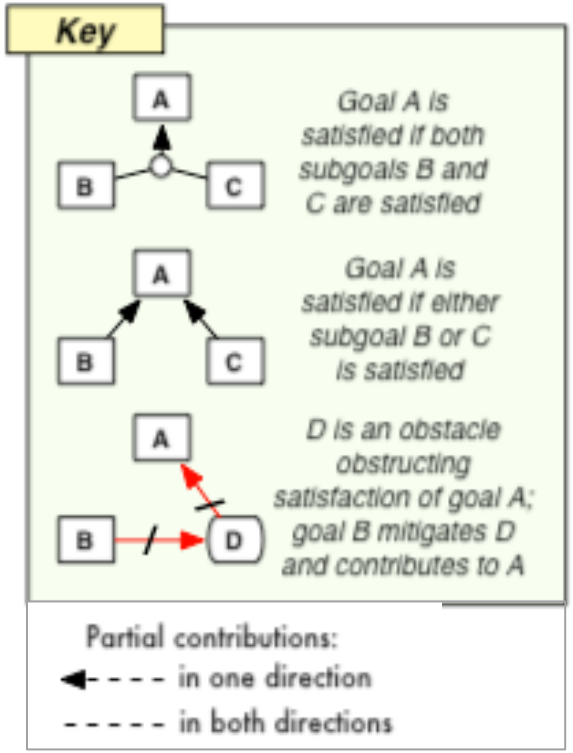
- Well-suited for confirming/prioritizing previously identified candidate requirements
- A set of questions are sent to a (potentially large) number of stakeholders
- Very limited suitability for developing new and innovative requirements
- Steps:
  - Preparation: select questions & target participants; prepare (Web) form
  - Execution: contact participants, remind deadlines, thank answers
  - Follow-up: check data quality, compute statistics, inform participants about the results

Goal analysis

- Hierarchical decomposition of stakeholder goals to derive system requirements
- Goal versus Requirement
  - Goal - a desired state (e.g., increase web sales by 10% in 2 years)
  - Requirement - a desired property of a system (for reaching a goal)
- Benefits of focusing on the notion of goals in RE:
  - Helping identifying requirements (ask why, how several times)
  - Helping justifying the presence of requirements
  - Helping detecting and resolving requirements conflicts

Goal analysis: example (elevator)





Social Observation and Analysis

- Requirements can be derived from the external observation of the routine way and tactics of work
- Many systems are developed to support people work
- People often find it difficult to tell how they perform routine tasks and work with others.
  - When tasks become routine and people don't think much about them consciously, it is hard to verbalize how the work is done
  - Example: Try to explain how to tie your shoelaces

https://www.youtube.com/watch?v=wMuNjnNyaiA

Prototyping

- A prototype is an initial/primitive version of a system
  - Cheaper, easier & faster to develop than the real system
  - Limited functionality
- User interface prototypes give an early preview of what the final system will look and work like, and are used in RE to:
  - Address areas of higher uncertainty and risks of misunderstandings
  - Validate previously identified requirements and identify new ones
- Types:
  - Throw-away prototypes (paper or computer based) - Ensures focus on requirements rather than implementation constraints
  - Evolutionary prototypes - Appropriate for rapid, iterative, application development with strong end user involvement

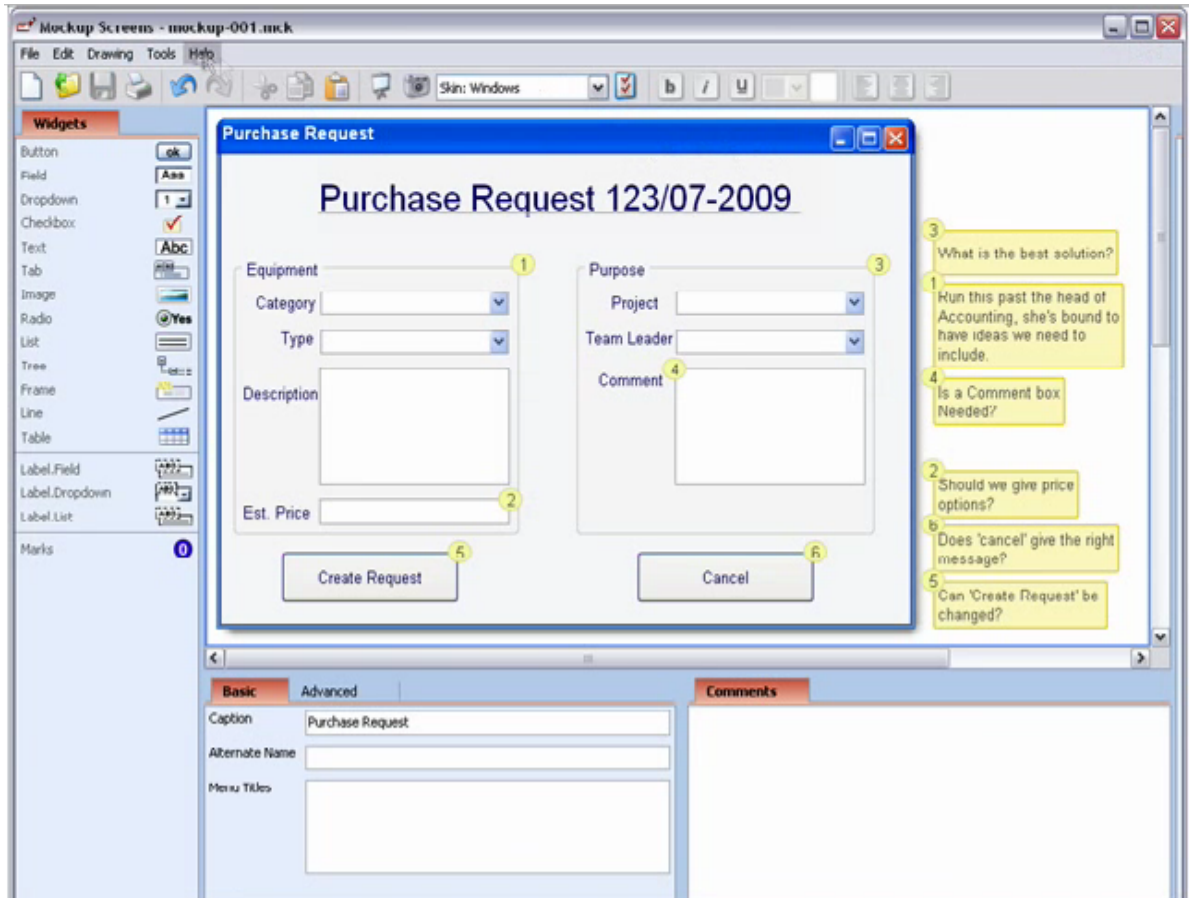
Paper Prototyping

- Quick, easy and cheap to develop
- Low fidelity
- Usually the preferred approach for requirements elicitation

http://www.youtube.com/watch?v=5Ch3VsautWQ

Computer-Based Prototypes

- More time, skills and cost to develop
- Higher fidelity
- Functional, evolutionary prototype
- Or non-functional, throwaway drawings and mockups



http://www.mockupscreens.com/index.php?page=Screen-Prototypes
http://www.youtube.com/watch?v=B8zNyEkCiGo&feature=related

nnovation Lab: Sunglass iPad App Case Study

NORDSTROM

INNOVATION LAB

0:04 / 6:42

References and further reading

- Software Engineering, Ian Sommerville, 9th Edition (chap.6-7)
- Software Requirements, 2nd Edition, Karl E. Wiegers, Microsoft Press, 2003
- Guide to the Software Engineering Body of Knowledge (SWEBOK), 2004 edition, IEEE Computer Society
- IEEE Std 830-1998 - Recommended Practice for Software Requirements Specifications
- IEEE Std 610.12: 1990 - Standard Glossary of Software Engineering Terminology
- ISO/IEC 12207 - Information technology - Software life cycle processes
- ISO/IEC 25000 family of standards - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)
- “Requirements Engineering as a Success Factor in Software Projects”, Hubert F. Hofmann, Franz Lehner, IEEE Software 2001

ademar.aguiar@fe.up.pt