Requirements Engineering

L.EIC-ES-2021-22

J. Pascoal Faria, Ademar Aguiar

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

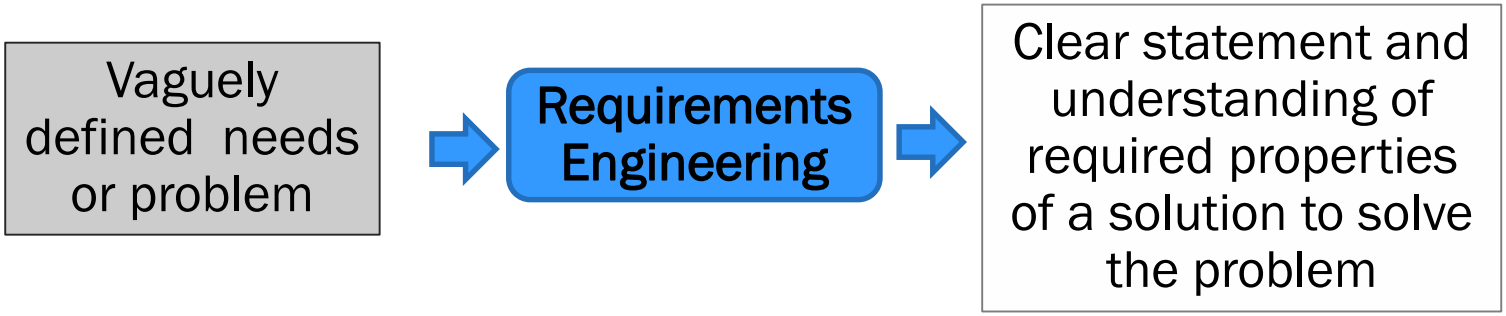
Contents

- Scope, importance & challenges of requirements engineering
- Types and sources of requirements
- Requirements engineering process
- User stories writing
- Acceptance tests
- User interface prototyping
- Supplementary materials (informative)
  - Requirements engineering in some well-known processes
  - Definitions of quality characteristics
  - Requirements engineering activities
  - 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

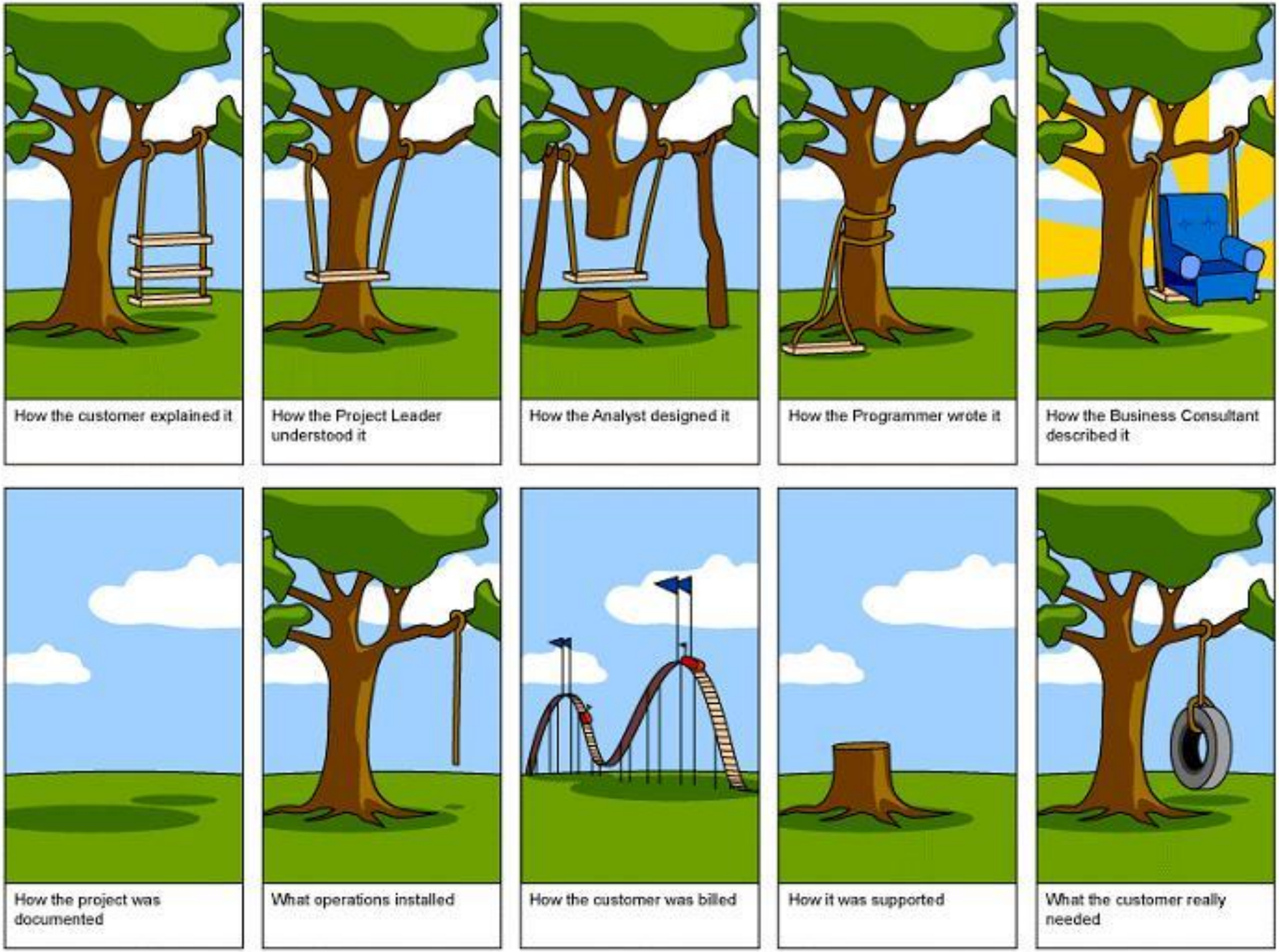
1	User Involvement	16 %
2	Executive Management Support	14 %
3	Clear Statement of Requirements	13 %
4	Proper Planning	10 %
5	Realistic Expectations (scope, budget, schedule,...)	8 %
6	Smaller Project Milestones	8 %
7	Competent Staff	7 %
8	Ownership	5 %
9	Clear Vision & Objectives	3 %
10	Hard-working,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

(source: Standish group CHAOS report)

Main challenges of RE

- Ensure requirements communication & understanding
  - Challenging because of different backgrounds, implicit knowledge, etc.
- Manage 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

Types and sources of requirements

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.*
  - May also be expressed from the user perspective, as a user story: *As a customer, I want to receive an e-mail notification when the items I ordered are dispatched, so that I can prepare for reception,*
- **Nonfunctional requirements** are the ones that act to constrain the solution
  - Most of them refer to product quality (sub)characteristics or "ilities" (see next slide)
    - Example: *The maximum system down-time should be 8 hours per year (availability requirement).*
  - Can also include development process requirements
    - Example: *The product should be developed in Java (requirement source: maintenance company).*

Quality characteristics and sub-characteristics [ISO/IEC 25010 standard]

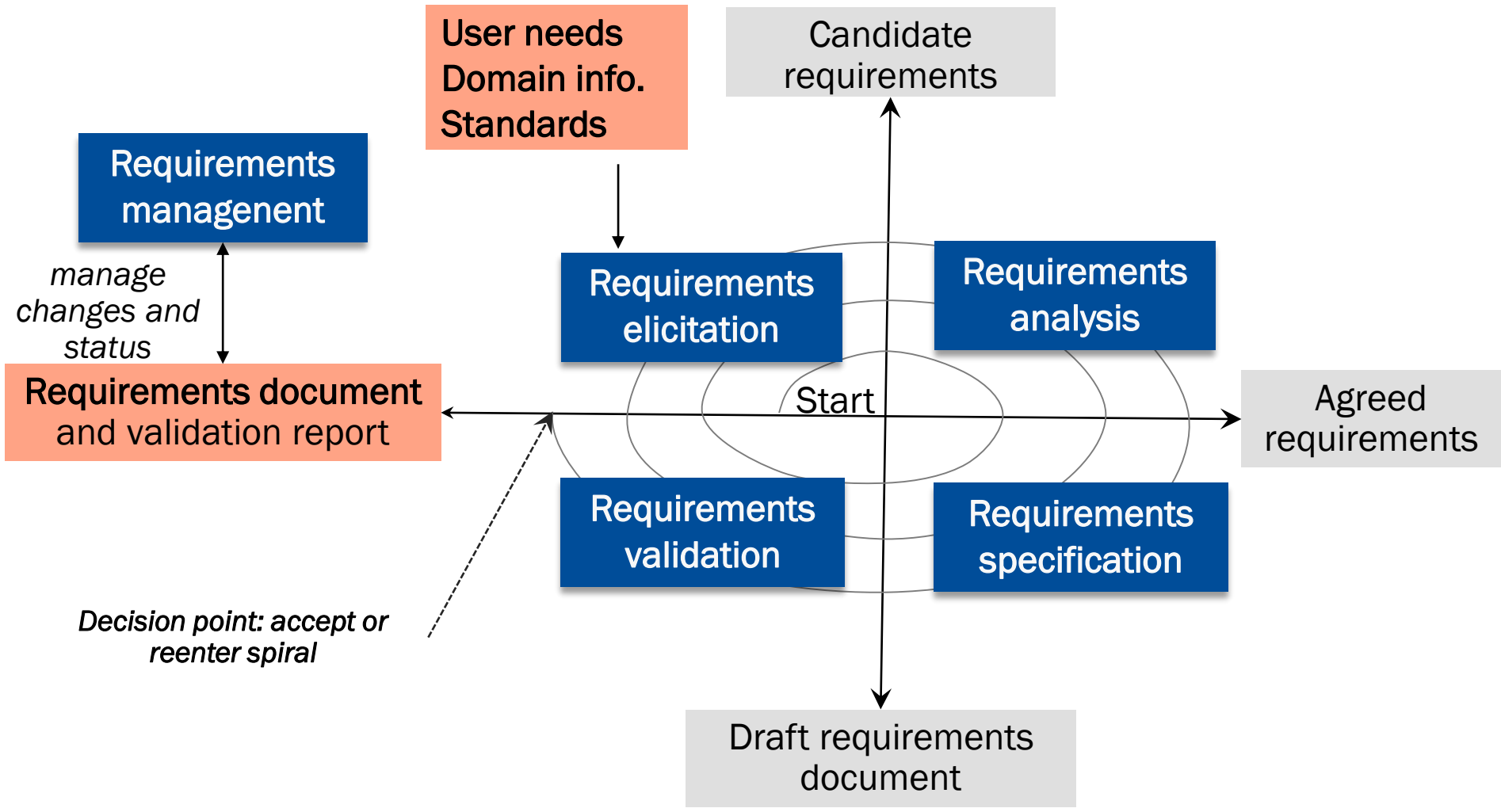
Use as a checklist for discovering nonfunctional requirements!

Sources of requirements: stakeholders

- **Stakeholders** are the main sources of requirements: people who will be affected by the system and who have a direct or indirect influence on the elaboration of requirements
  - Customers ("pay" for the system)
  - End users (use the system)
  - 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 engineering process

Requirements enginering process



Requirements engineering activities

- **Elicitation**: interact with stakeholders and other sources (through interviews, etc.) to discover their needs and requirements
- **Analysis**: organize and assess the collected information (for completeness, consistency, clarity, etc.), in order to arrive at a prioritized list of agreed requirements
- **Specification**: produce requirements documentation with an appropriate level of detail, depending on the context
- **Validation**: make sure (through reviews, etc.) that the documented requirements allow achieveing the project's business objectives

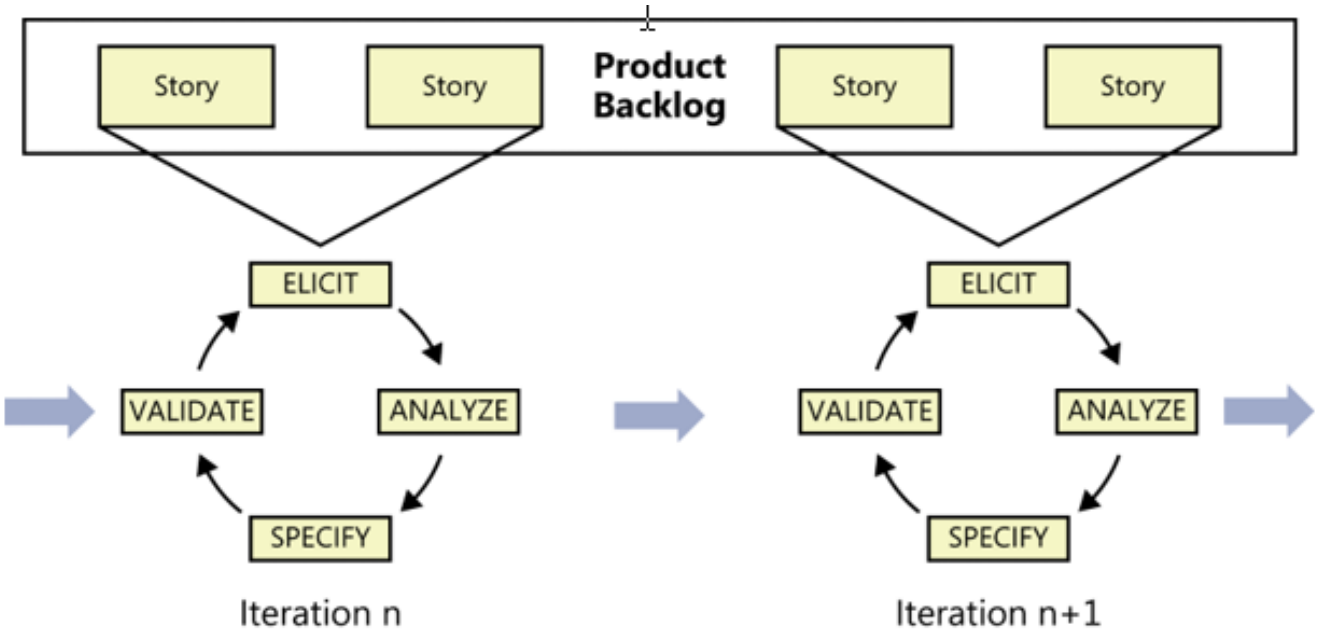
Requirements engineering artifacts

- List of requirements (mandatory)
  - Possible attributes: priority, type, source, status, complexity
  - May be organized in clusters of related requirements
- System models (e.g., use case and domain models)
  - Help to remove ambiguity in natural language descriptions, find problems in requirements, tackle complexity through abstraction, and guide subsequent development.
- User interface prototypes or mockups
  - Help to address areas of uncertainty, validate requirements and discover new ones, and guide subsequent development
- Acceptance tests
  - Early derivation of acceptance tests from requirements helps to clarify and find problems in requirements, and support TDD

All required in practical projects for learning purposes!

Requirements engineering in agile methods

- To better cope with changing requirements & uncertainty, agile methods favor incremental, lightweight, requirements engineering.
- An evolving **product backlog** contains a prioritized list of requirements, possibly expressed as **user stories.**



User stories writing

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

I ndependent

N egotiable

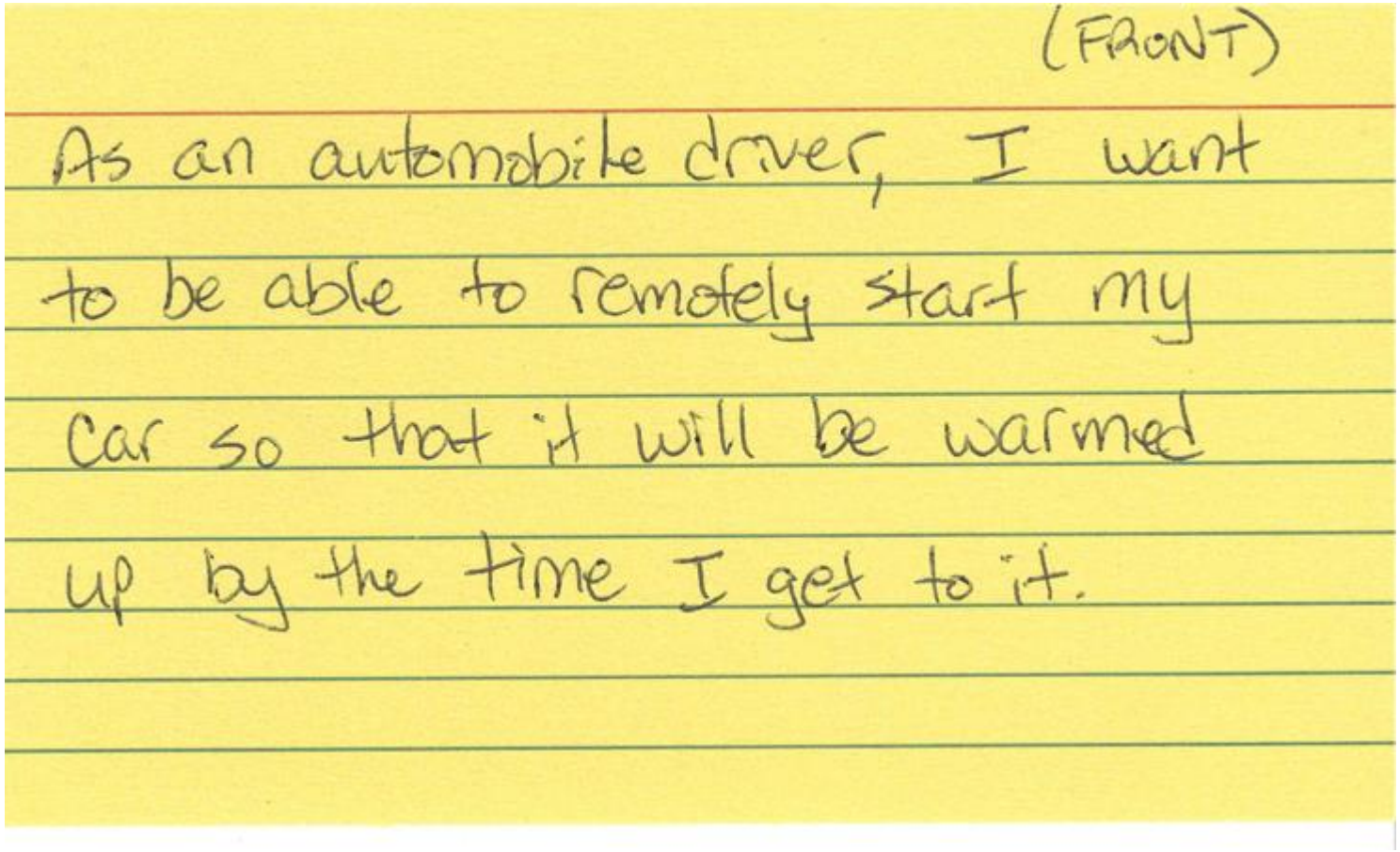
V aluable

E stimable

S mall (Sized appropriately)

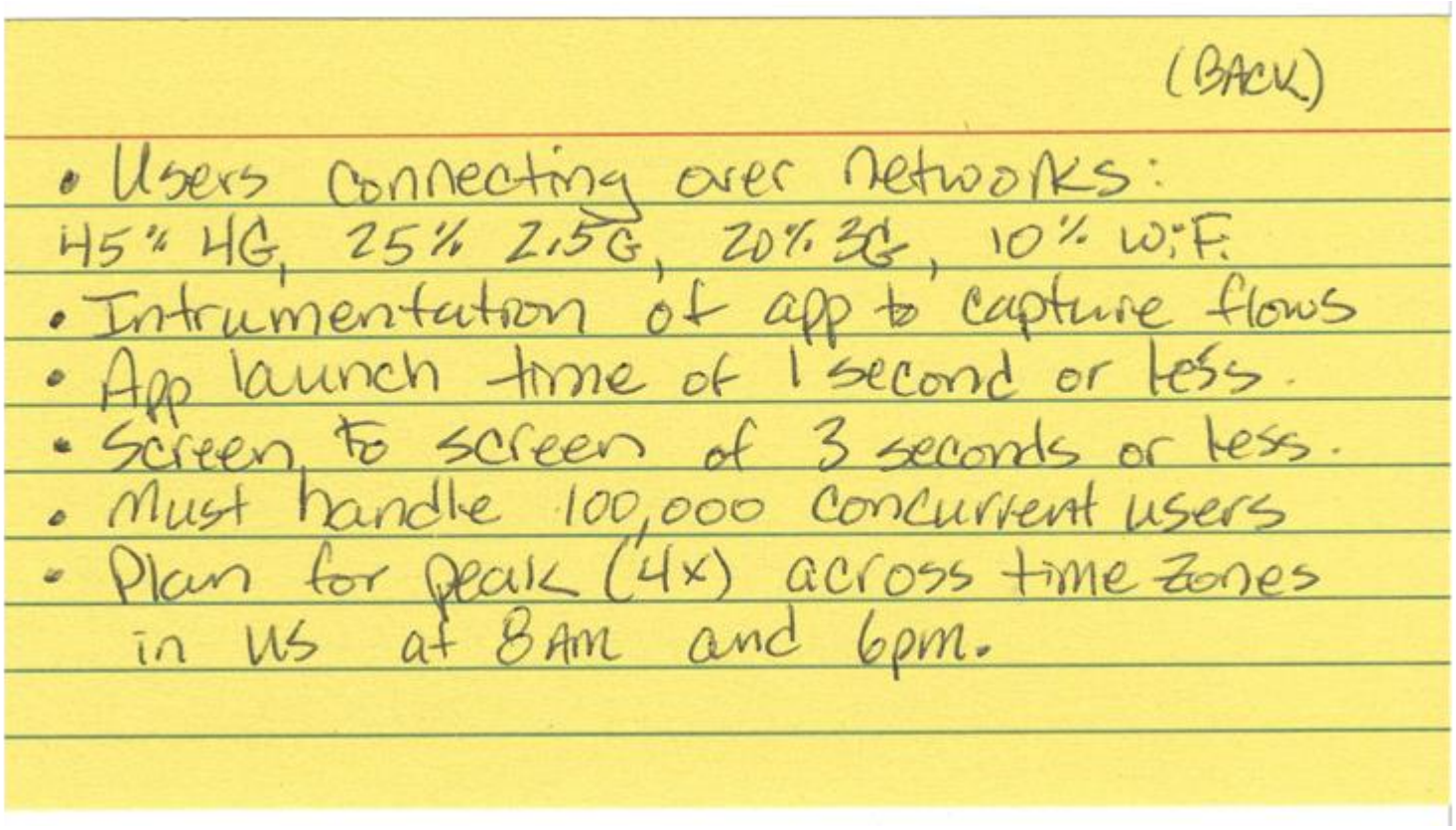
T estable

User Stories – Example (1/2)



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

User Stories – Example (2/2)



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

Acceptance tests

Acceptance tests

- **Acceptance tests** are test cases defined for customers to decide if a system or feature implementation can be accepted (i.e., satisfies requirements and expectations).
- In agile processes, one or more acceptance tests may be defined for each user story (as the **acceptance criteria**)
- A common format is that of **Behavior-Driven-Development (BDD)**, i.e., test scenarios specifying expected system behaviors according to the template

  Given [initial context or preconditions]
  When [event(s) occur(s)]
  Then [ensure some outcomes or postconditions]

- Such scenarios can be written in a language such as **Gherkin** and automated with tools such as **Cucumber**

Example

```
Feature: Write comments
  As a blog reader
  I want to be able to write a comment
  So that I tell the author my opinion / feedback

  Background: Login with author user
    Given that the author adds a new post with title "Post to comment"


  Scenario: Leave a comment with all info filled in
    Given that I select the post
    When I add a new comment with name, email  and  body
    Then I will see the comment on the blog


  Scenario: Leave a comment with name field not filled in
    Given that I select the post
    When I add a new comment with email and body
    Then I will see the message "ERROR: please fill the required fields (name, email)."
```

User interface prototyping

User interface prototyping

- A prototype is an initial/primitive version of a system
  - Cheaper, easier & faster to develop than the real system
  - With 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

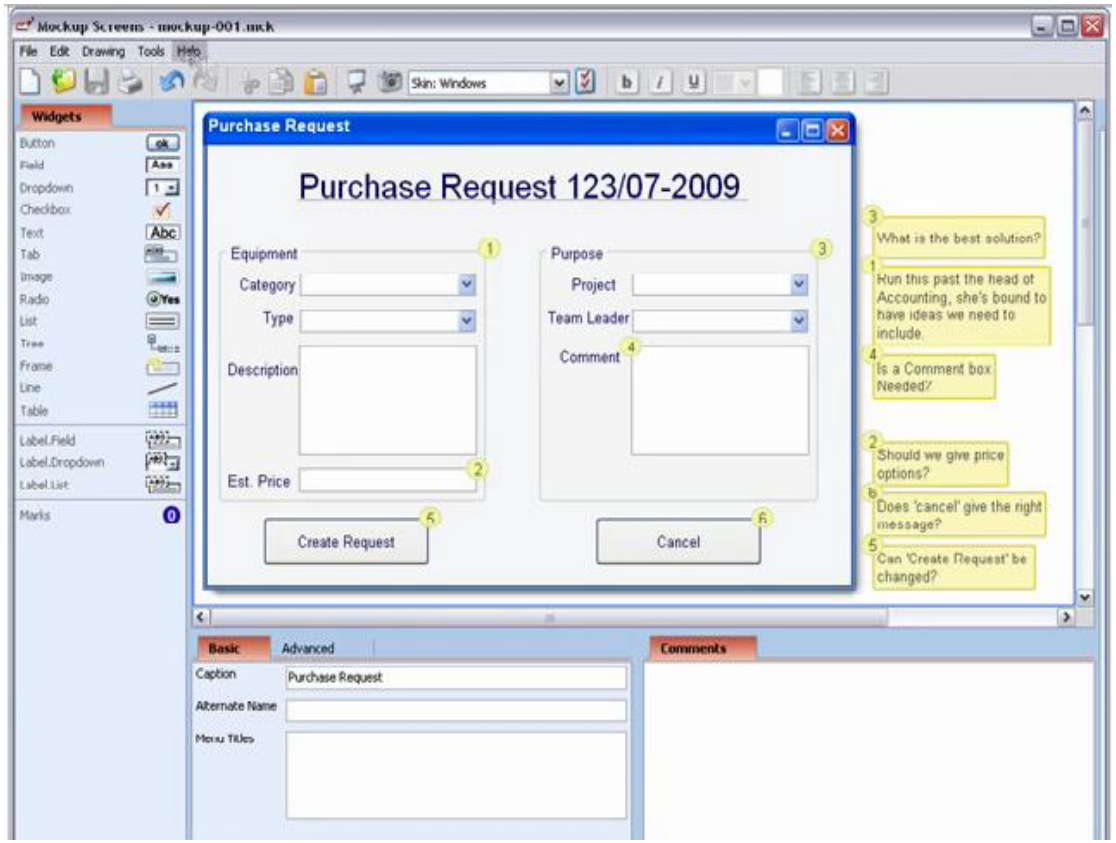
Paper prototyping

- 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



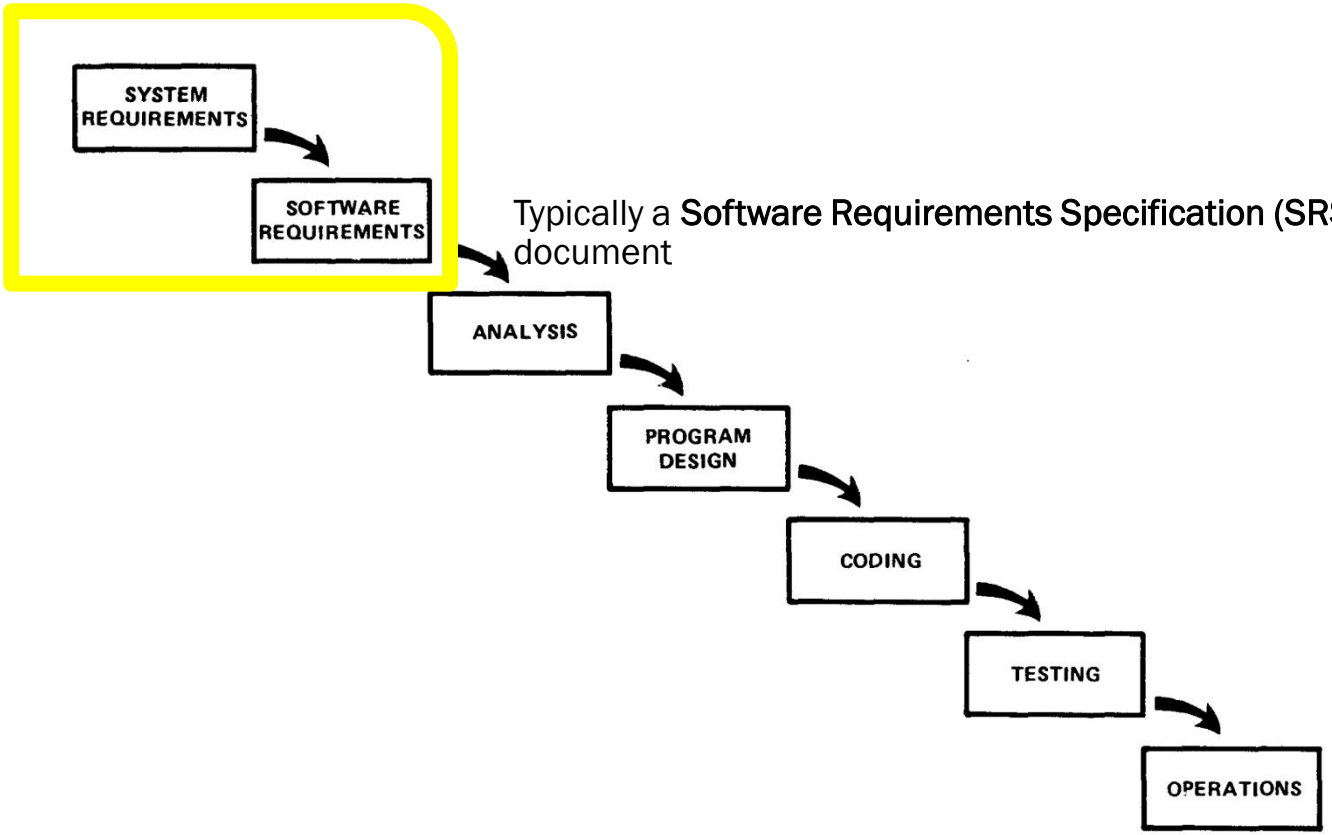
http://www.mockupscreens.com

References and further reading

- Software Engineering, Ian Sommerville, 10th Edition (chap.4-5)
- 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 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

Supplementary materials: Requirements engineering in some well-known processes

Waterfall

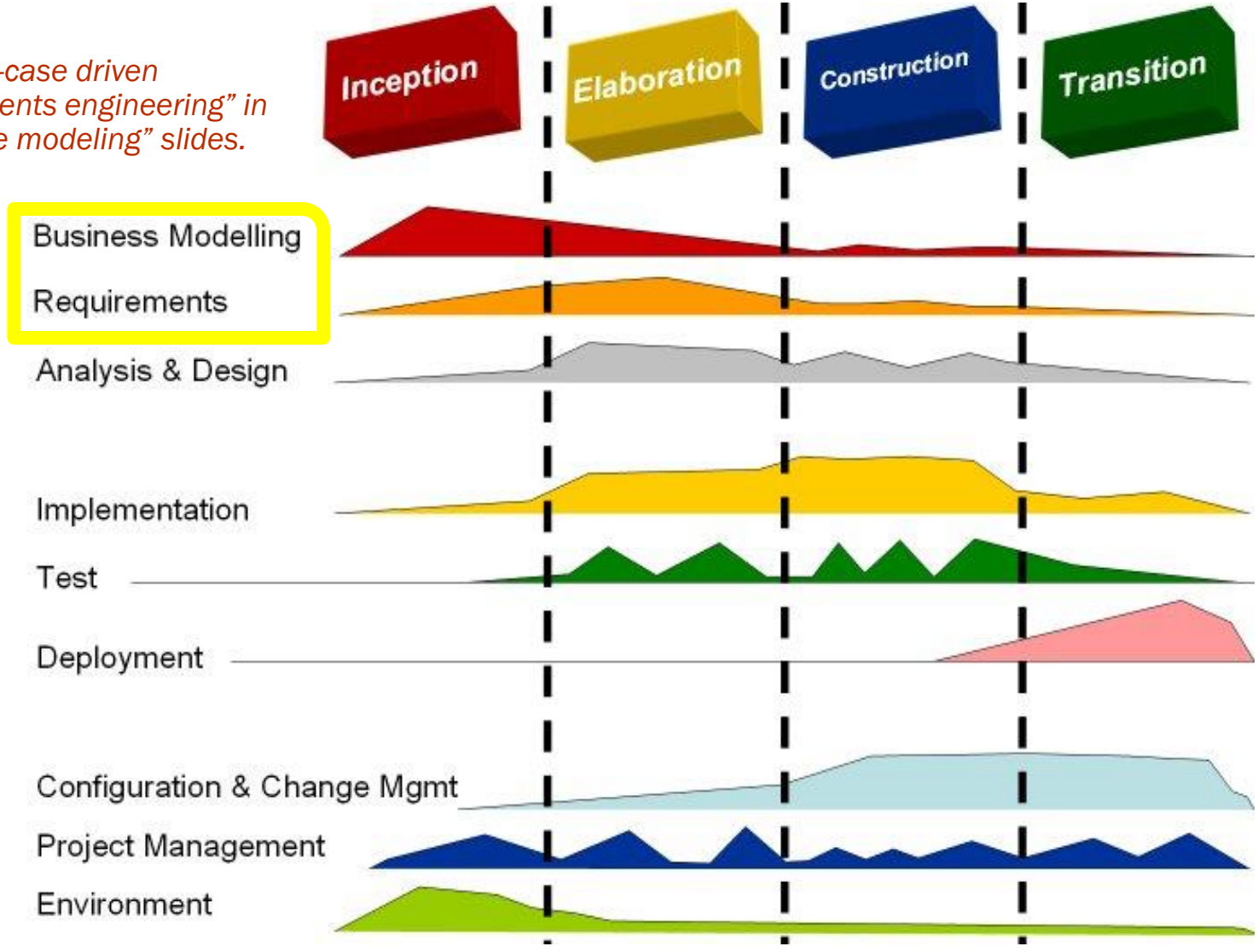


Typically a **Software Requirements Specification (SRS)** document

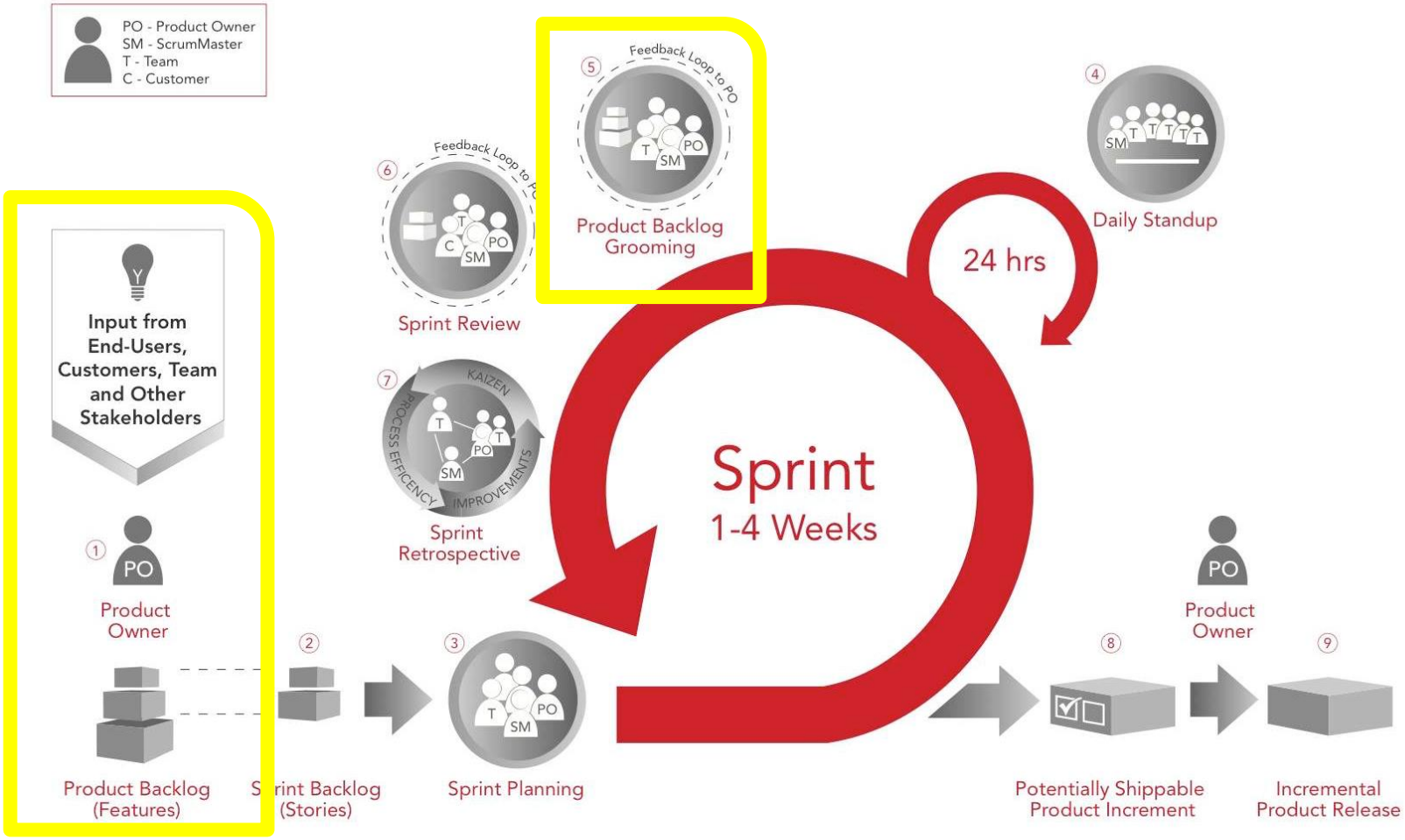
Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

Rational Unified Process

See "use-case driven requirements engineering" in "use case modeling" slides.

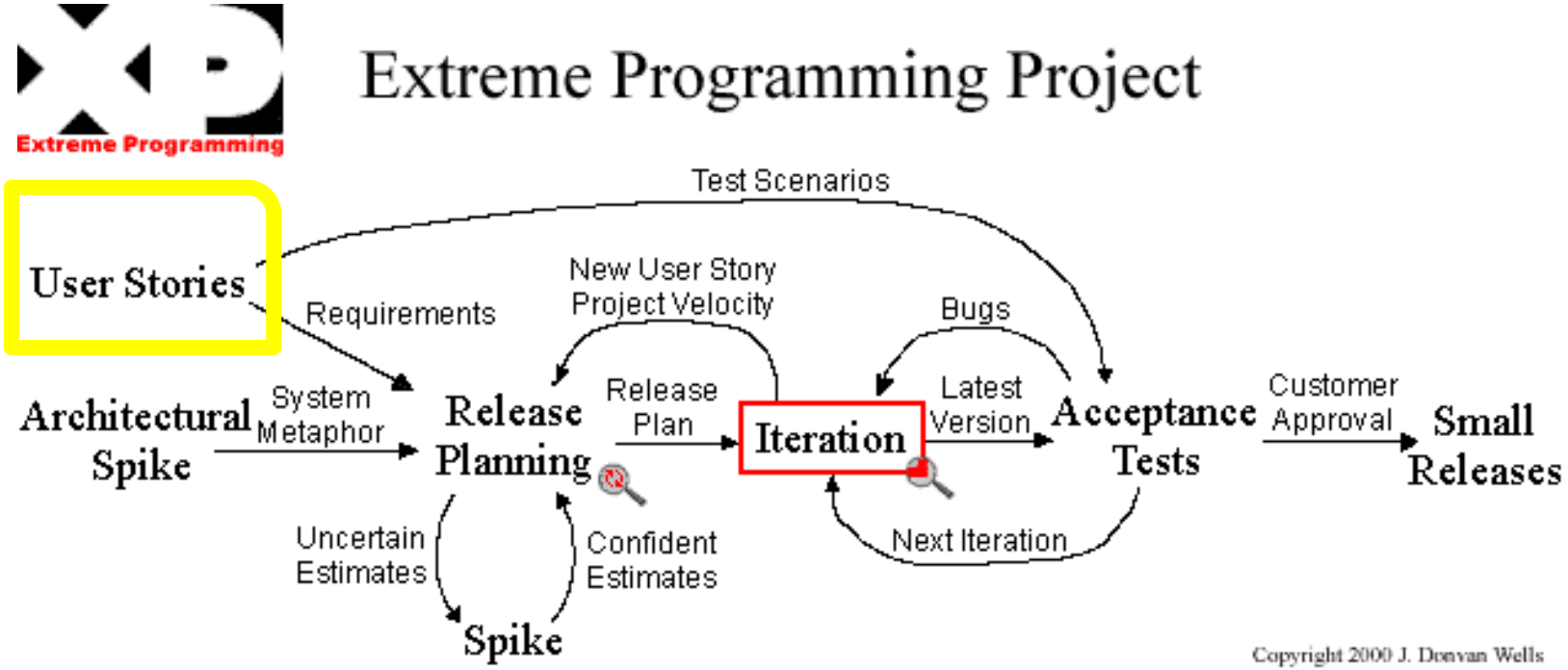


Scrum



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

Supplementary materials: Definitions of quality characteristics

Definitions of 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

Supplementary materials: Requirements engineering activities

Requirements elicitation

- Goals:
  - Interact with **stakeholders** and other sources (documents, existing systems, etc.) to elicit/discover their requirements
- Techniques:
  - Interviews (most commonly used technique)
  - Facilitated meetings (brainstorming, focus groups, etc.)
  - Questionnaires (surveys)
  - Goal analyses (goal-driven RE)
  - Social observation and analysis (of how people actually work)
  - Social media analysis (including customer reviews)
  - User-interface prototyping (to validate & discover new requirements)
  - Scenarios, user stories, use cases (real-life usage examples, idem)

Requirements elicitation

Requirements analysis

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

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

Requirements specification

- Of varying complexity, depending on the context:
  - A Software Requirements Specification (SRS) document, following a template such as the one defined by IEEE Std 830-1998
  - A simple Product Backlog, with a prioritized list of requirements, expressed as user stories or product features
- Often accompanied by other artifacts, such as:
  - User-interface prototypes
  - Models
    - E.g., use case and domain models in UML
  - Other documents
    - Preliminary user manual
    - Glossary (business and technical terms)
  - Tables and matrices
    - Requirements attributes tables (with priority, source, etc.)
    - Traceability matrices (requirements to user needs, test cases, etc.)

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

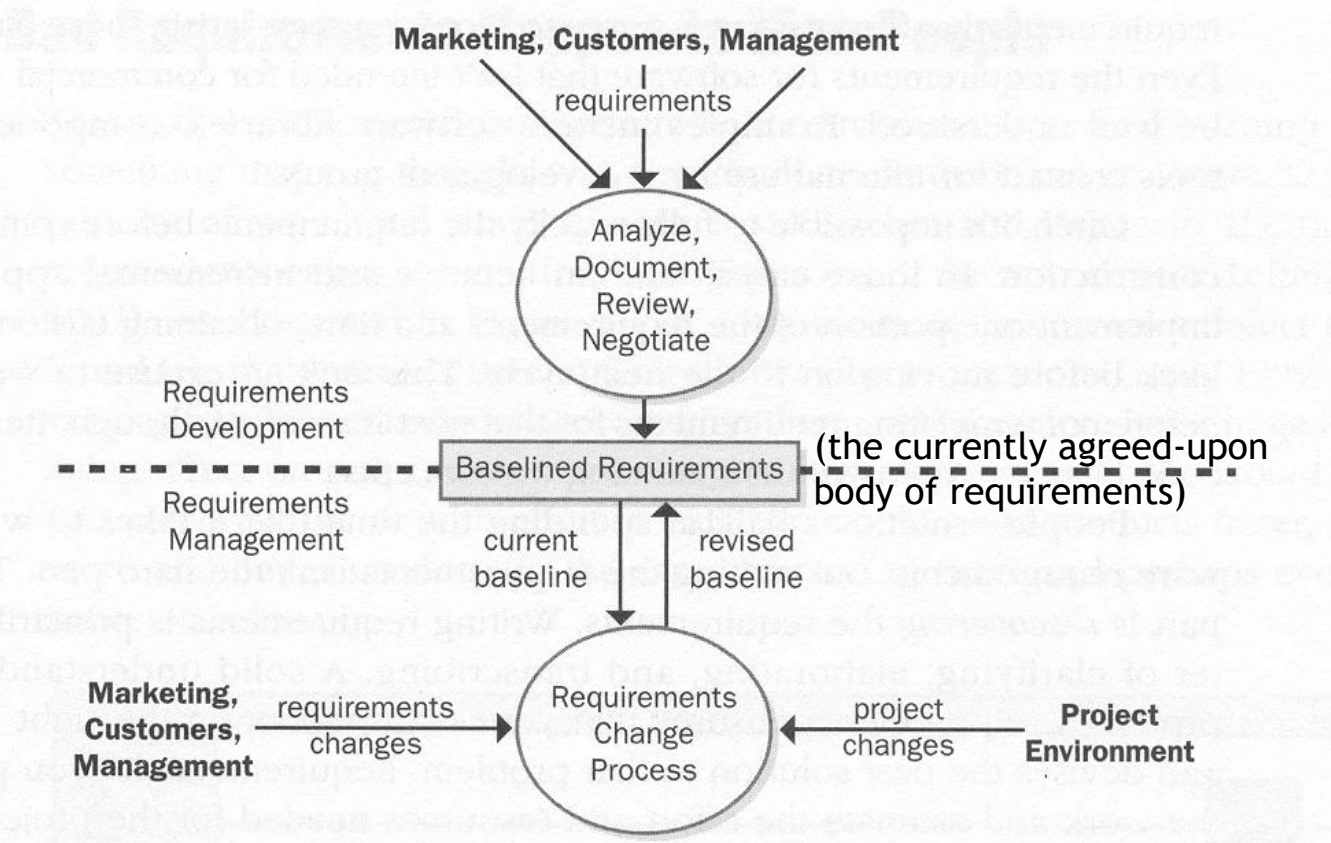


Figure 1-3 The boundary between requirements development and requirements management.

(Software Requirements, K.E.Wiegers)

Supplementary materials: 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.

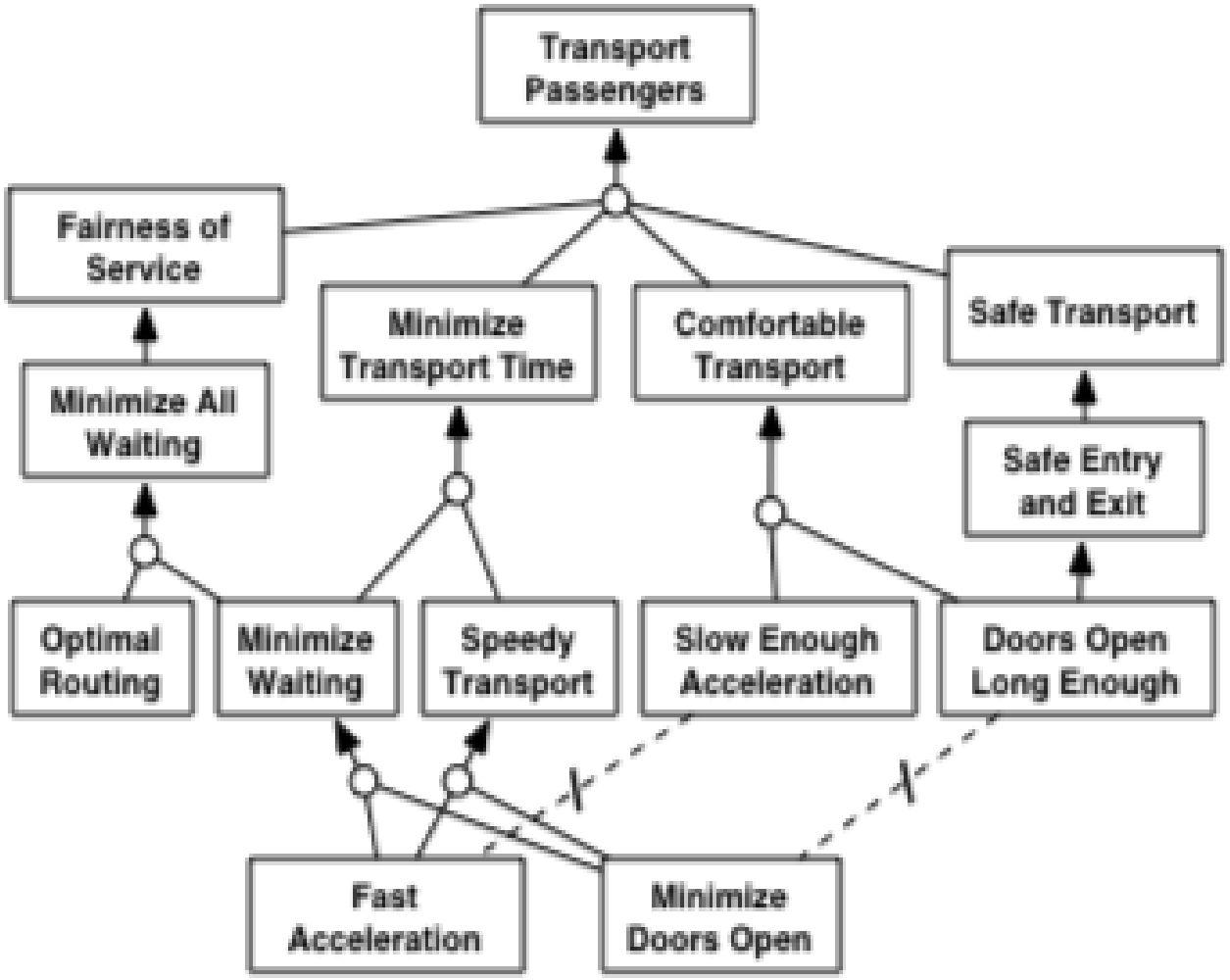
Questionnaires (surveys)

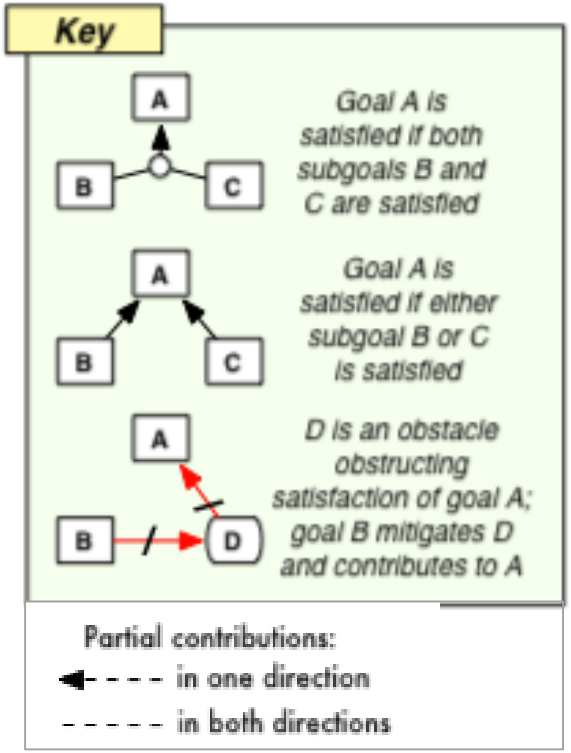
- Well-suited for confirming/prioritizing previously identified candidate requirements
- A set of questions are sent to a (potentially large) number of stakeholders
- Very limited suitability for developing new and innovative requirements
- Steps:
  - Preparation: select questions & target participants; 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

nnovation Lab: Sunglass iPad App Case Study

NORDSTROM

INNOVATION LAB

0:04 / 6:42

jpf@fe.up.pt, ademar.aguiar@fe.up.pt