CS3500 Software Engineering

2017/2018

Dept. Computer Science
Dr. Klaas-Jan Stol

University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Welcome to CS3500

Moodle

- No account?
  → email help@cs.ucc.ie

- All slides, assignments, and papers made available here.

- All lectures and labs also scheduled in Moodle calendar.

Reminder: Schedule Adjustment

- Next lecture as usual: Monday 25th Sep

- No lecture on:
  - Wednesday 27th September
  - Monday 2nd October
  - Wednesday 4th October

- No labs on:
  - Thursday 28th September
  - Thursday 5th October

- These will be rescheduled later.

Lab session start tomorrow (20^{th} Sep)

- Presence tomorrow mandatory
- We will make teams for graded tasks
- I will present first task.
- First task due October 16
  - Submission & details follow soon on Moodle.

Requirements Engineering (RE)

Part I

After studying this material and associated papers, you should be able to:

- Define what a system is, what a requirement is, and what requirements engineering is.
- Define what stakeholders are, and describe the different stakeholders involved in a system and their concerns.
- Be able to classify requirements and explain what SMART requirements are.
- Describe and use techniques for identifying, documenting, and prioritizing requirements.

Contents

This Lecture

1. Definitions

2. Classifying requirements

3. SMART requirements

4. Identifying requirements

5. Requirement specification

6. Prioritizing requirements

SECTION I

Definitions

1. System

2. Requirement

3. Stakeholder

4. Requirements Engineering

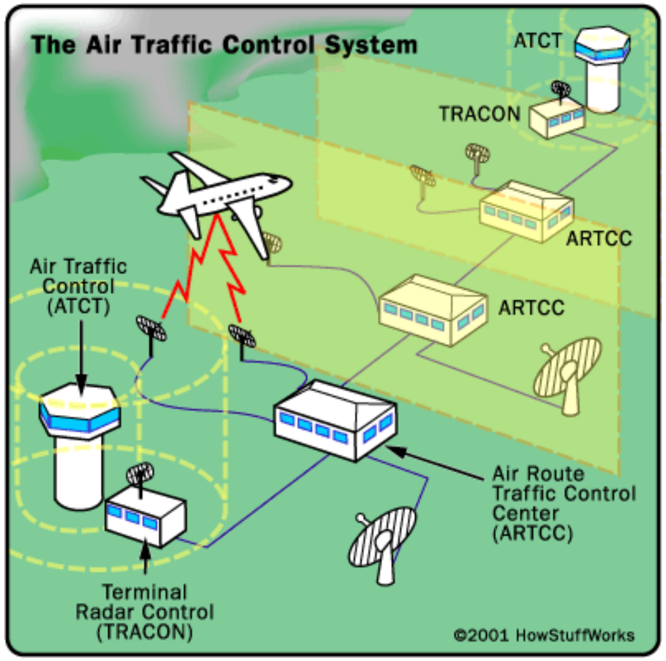
Definitions

System:

A collection of components (machine, software, human) which co-operate in an organised way to achieve some desired result—the requirements.

EXAMPLE

Air Traffic Control System



Left:
Simplified overview of components of an ATC

Below:
User-view of an Air Traffic Controller.

Definitions

Requirement:

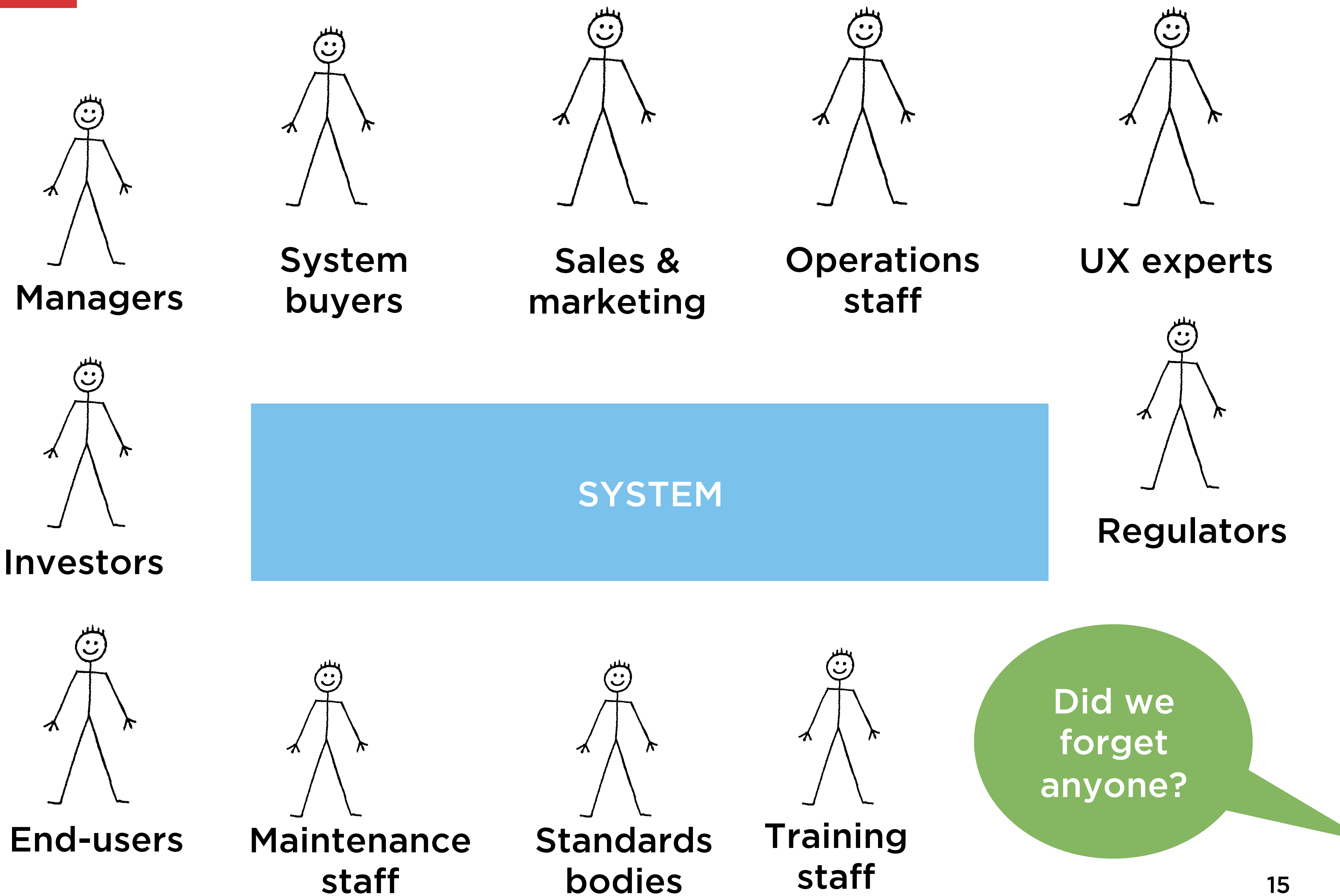
A statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines).

Definitions

Stakeholder:

An individual, group of people, organisation or other entity that has a direct or indirect interest (or stake) in a system.

Stakeholders



Stakeholders

Stakeholders and concerns

Stakeholder	Example Concerns
Managers	• Responsible for budget and progress
Investors	• Invested money in the project
End-users	• Usability • System capability
Maintenance and service staff	• Keep system running (no changes please!)
Training staff	• Usability • Documentation
System buyers	• Price • System capability
Sales & marketing	• System capability • Competitors
User Experience (UX) experts	• Usability
Regulators (e.g. FDA)	• Standards & regulation compliance
Developers	• Technical feasibility • Project schedule

Definitions

Requirements engineering:

The subset of systems engineering concerned with discovering, developing, tracing, analyzing, qualifying, communicating and managing requirements that define the system at successive levels of abstraction.

The importance of requirements

A Software Requirement Specification (SRS) serves different purposes:

Created by jai
from Noun Project

1. Provides a basis for standardizing process and products.
2. Provides a standard for measuring progress.
3. Provides guidance to developers to decide what to do next, and when they are finished.

The importance of requirements

- Requirements should specify WHAT the software should do, not HOW.
  - Understand and specify the problem, not the solution
- Problem space vs Solution space
  - Problem space defines WHAT
  - Solution space defines HOW

However: Impossible to separate specification from implementation

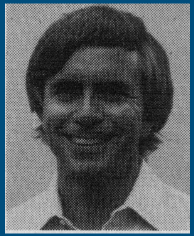
1. Limitations of implementation technology may force a specification change.
   1. Problem: implement a “stack” data structure (LIFO)
   2. Solution: use an array (fixed capacity) or linked list (flexible capacity).
   3. Choice of implementation affects specification.

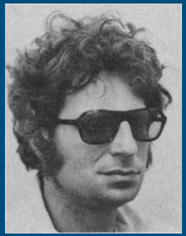
2. Implementation choices (solution space) may augment original specification.
   - Problem: provide a pattern-match routine
   - Solution: use COTS pattern-match component that also allows wildcards (not originally specified)
   - Choice of implementation (COTS) affects specification.

Reading Assignment

“On the Inevitable Intertwining of Specification and Implementation”

By: William Swartout and Robert Balzer





Originally published in:
Communications of the ACM, volume 25, number 7, July 1982, pages 438-440.

Estimated reading time:
3 pages, 45 min.

SECTION II

Classifying Requirements

1\. Functional vs non-functional

2\. Behavioral vs developmental quality

3\. Architectural Significant Requirements

Functional vs Non-functional

Classification 1:

- **Functional Requirements:**
  what a system must do

- **Non-Functional Requirements (NFR):**
  all other constraints, such as performance, reliability, modularity, safety, and other –ilities.

?

Problem: can be ambiguous. Is performance of a video compression algorithm ‘functional’ or ‘non-functional’ if it hampers smooth playback?

Metrics for selected NFRs

Property	Measure
Performance	• Transactions per second • User/Event response time • Screen refresh time
Size	• Megabytes • Number of ROM chips
Ease of use	• Training time • Number of help frames
Reliability	• Mean time to failure (MTTF) • Probability of unavailability • Rate of failure • Availability
Robustness	• Time to restart after failure • Percentage of events causing failure • Probability of data corruption on failure
Portability	• Percentage of target dependent statements • Number of target systems

Adapted from I. Sommerville, "Software Engineering", 9th Ed.

Behavioral requirements vs Developmental quality attributes

Classification 2:

- Behavioral requirements:
  All information needed to determine if the run-time behavior of a solution is acceptable, incl.
  - Performance
  - Security

- Developmental quality attributes:
  include any constraints on the attributes of the system's static construction, incl.
  - Testability
  - Changeability
  - Maintainability
  - Reuseability

Architectural Significant Requirements vs Non-ASR

Classification 3:

- Architectural Significant Requirements (ASR)
- Non-ASR

This means:
Achieving them has implications for the architecture design of the system.

More on ASRs later!

SECTION III
SMART Requirements

Writing good requirements is hard. Let's be SMART.

Requirements should be SMART

- Specific
- Measurable
- Attainable
- Realistic
- Traceable

SMART: S for Specific

A requirement must state exactly what is required

1. Clear & precise: no ambiguity
2. Consistent: same terminology
3. Simple: split up statements if needed
4. Sufficient detail

EXAMPLE

SMART: S for Specific

The mission planning system shall support several planning environments for generating the mission plan

Not very specific:

- How many are “several”?
- What is a “planning environment”?
- What is a “mission plan”?

The system shall support 50 simultaneous users

SMART: M for Measurable

Once a system is constructed, is it possible to measure (verify) that a requirement was satisfied?

- Some requirements cannot be measured without special instruments/tools
  —e.g. memory leaks
- Some requirements cannot be measured
  —e.g. no specific answer can be provided.

EXAMPLE

SMART: M for Measurable

The system shall produce a plan optimised for time.

Not measurable:

- What is an optimal time plan?
- How do you know whether it's optimized or not without knowing the optimum?

The system shall produce a production plan that can be performed within 1 hour.

SMART: A for Attainable (Achievable)

Requirements must be technically attainable.

- Some requirements cannot be achieved due to
  - technical constraints
  - project resource constraints

EXAMPLE

SMART: A for Attainable (Achievable)

The system shall be 100% reliable and 100% available.

Not attainable:

- Simply not possible in practice.

The system shall be available at least 99.9%.

SMART: R for Realistic (Realisable)

Requirements must be realisable, realistic, relevant, worthwhile, sensible

Note difference realistic v. achievable:

- Achievable means possible given existing constraints.
- Realisable means something that is realistic and sensible.

EXAMPLE

SMART: R for Realistic (Realisable)

The system shall store a copy of Google’s search index.

Not realistic:

While technically achievable (because Google’s doing it), it’s not realistic or sensible.

The system shall store the top 100 search results based on Google search.

SMART: T for Traceable

Ability to trace a requirement from statement to design, implementation, and test.

- To understand rationale for requirement
- To support verification
- To support modification

SMART: T for Traceable

Traceability not inherent in requirement itself, but requires additional information.
For example:

1. Originators of requirements (who requires it?)
2. Assumptions (what is left implicit?)
3. Business justifications
4. Relationship to other requirements
   (e.g. dependency, implications)
5. Criticality (priority)

“3C” Goals of RE

1. **Comprehension:**
   Understand what the software must do

2. **Communication:**
   Communicate this to all stakeholders

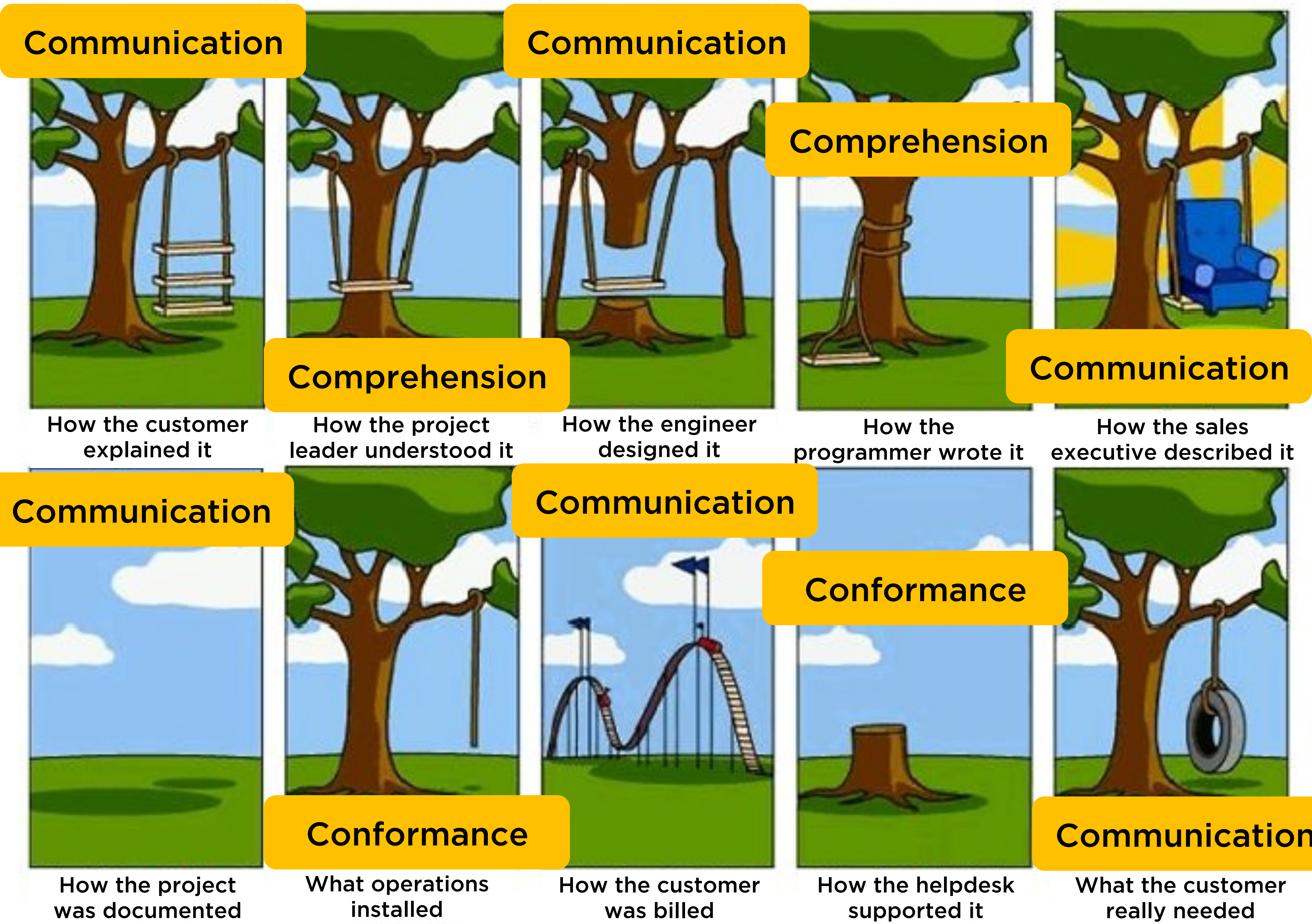
3. **Conformance:**
   Ensure that final system satisfies the requirements

“3C” Problems in RE

1. **Comprehension (understanding):**
   Problem: people do not know what they want or change their minds.

2. **Communication:**
   Problem: communicating requirements is hard because software is “thought stuff” and different stakeholders have different expertise.

3. **Conformance:**
   Problem: because requirements often change and might be conflicting (contradictory)


Communication
How the customer explained it
Comprehension
How the project leader understood it
Communication
How the engineer designed it
Comprehension
How the programmer wrote it
Communication
How the sales executive described it
Communication
How the project was documented
Conformance
What operations installed
Communication
How the customer was billed
Conformance
How the helpdesk supported it
Communication
What the customer really needed

Summary

- Definitions of System, Stakeholder, Requirement, Requirements Engineering

- Classifying requirements
  - Functional vs. Non-Functional
  - Behavioral vs Development quality
  - Architectural Significant vs Non-ASR

- Requirements should be SMART
  - Specific
  - Measurable
  - Attainable
  - Realistic
  - Traceable

- 3C Problems in Requirements Engineering
  - Comprehension
  - Communication
  - Conformance

Requirements Engineering (RE)

End of Part I

Thank you
for your attention

Questions & suggestions can be sent to:

k.stol@cs.ucc.ie