UNIVERSITY OF LEICESTER

Managing Requirements

CO3095, CO7095, CO7508

José Miguel Rojas – *material provided by Neil Walkinshaw*

Requirements Elicitation

*"The hardest single part of building a software system is **deciding precisely what to build**.*

*No other part of the conceptual work is **as difficult as establishing the detailed technical requirements**...*

*No other part of the work so **cripples the resulting system if done wrong**. No other part is **as difficult to rectify later**"*

Fred Brooks, No Silver Bullet: Essence and Accidents in Software Engineering, 1987

What is a “Requirement”

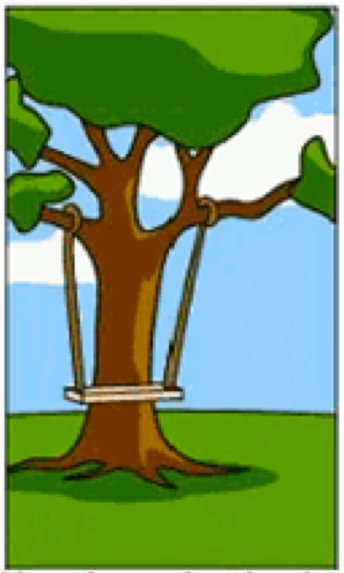
IEEE Std 610.12-1990:

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 document.
3. A documented representation of a condition or capability as in definition 1 or 2.

I would like a wash basin that will allow me to wash my hands.

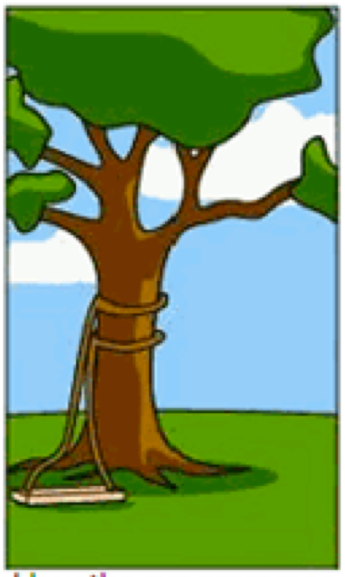
NO NO NO! I want WARM water, not a choice between freezing cold and boiling hot!

How the customer explained it

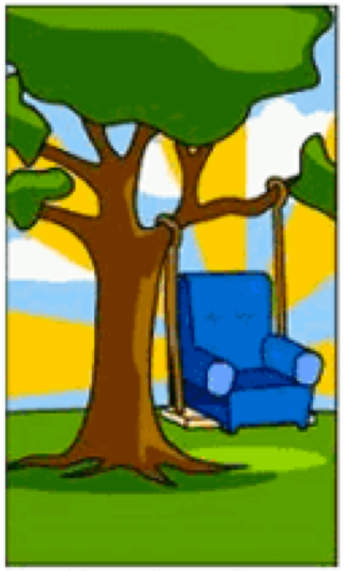


How the project leader understood it

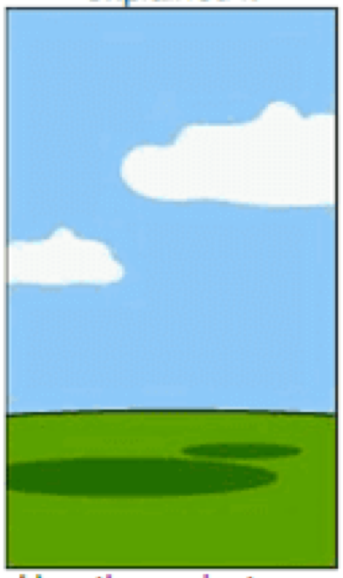
How the engineer designed it



How the programmer wrote it



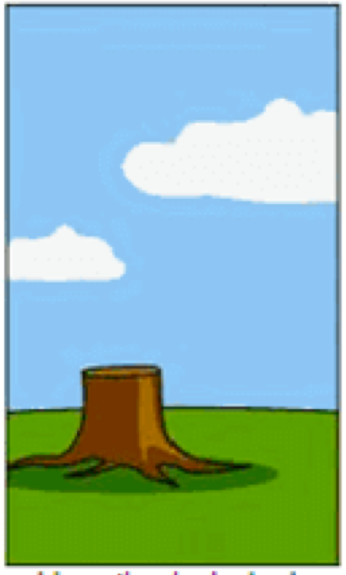
How the sales executive described it



How the project was documented

What operations installed

How the customer was billed



How the helpdesk supported it

What the customer really needed

Eliciting Appropriate Requirements is Difficult

Disagreement between stakeholders

Different priorities can lead to conflict

Indecision

Stakeholders may not be sure about what they want

Level of detail

Lack of detail can lead to ambiguities, too much detail becomes time-consuming and cumbersome

Scope

Scope of requirements may not be evident from the start

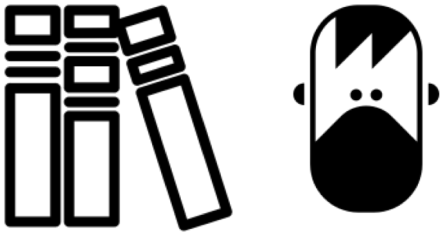
UNIVERSITY OF LEICESTER

Requirements Elicitation Steps

1. Understand the application domain

   Safety / business critical?
   How many users and use-cases?
   Subject to certification / standards?
   Vulnerable to changing requirements?

2. Identify sources of requirements

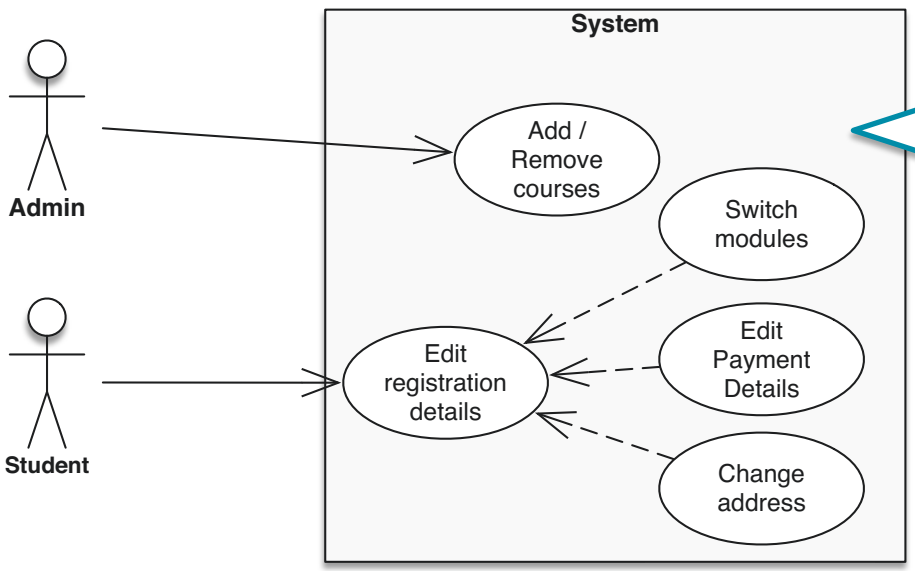


   Who are the obvious stakeholders?
   What is the workflow within which the system has to fit?
   Any legal requirements?

3. Analyse Stakeholders

   Focus groups
   Interviews
   Identify conflicts

Documenting Requirements as Use Cases



Use case diagram captures key use cases (and their relationships)

An accompanying textual description.

1. **Title:** Edit Registration Details
2. **Description:** User is able to edit their details on the database.
   a. Logs in
   b. Selects "edit registration details"
   c. Selects option.
   d. . . .
3. **Extensions**
   a. Switch modules
      i. User selects "Switch Modules"
      ii. User selects module they wish to switch.
      iii. . . .
   b. . . .

Software Requirements Specifications

A more comprehensive document that is structured as follows:

1. Introduction
   - Purpose
   - Scope
   - Definitions, acronyms, abbreviations
   - References
   - Overview
2. Overall description
   - Product perspective
     - System Interfaces
     - User Interfaces
     - Hardware interfaces
     - Software interfaces
     - Communication Interfaces
     - Memory Constraints
     - Operations
     - Site Adaptation Requirements
   - Product functions
   - User characteristics
   - Constraints
   - Assumptions and dependencies
   - Apportioning of requirements
3. Specific requirements
   - External interfaces
   - Functions
   - Performance
   - Logical database
   - Design constraints
   - Software System attributes
     - Reliability
     - Availability
     - Security
     - Maintainability
     - Portability
     - others

What do these remind you of?

Security Requirements

- Thinking about security requirements can be challenging
- There is no implementation yet
- Key design decisions have yet to be taken
- C-I-A: security model with three key principles:

Confidentiality

Private data should not be accessible

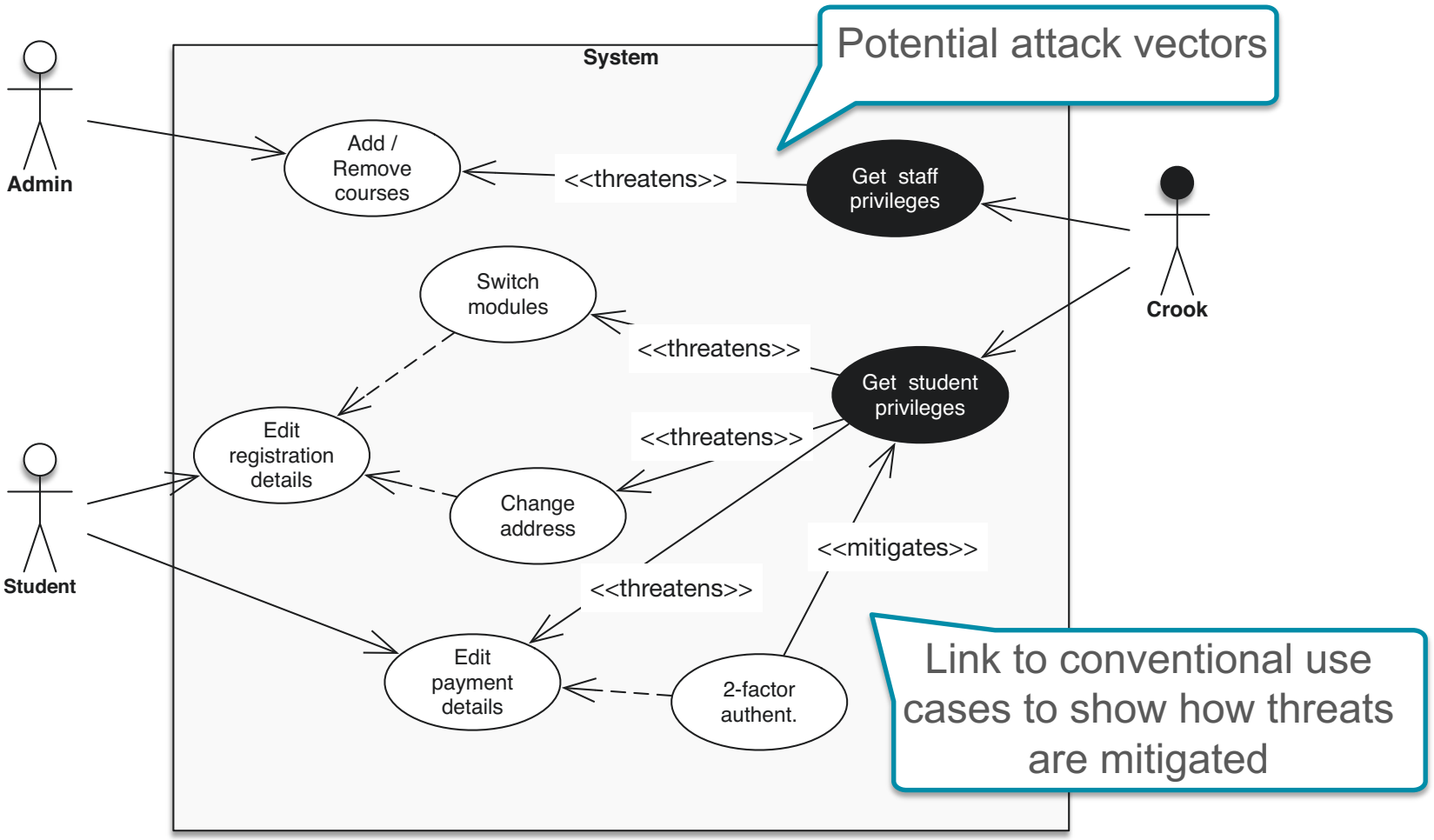
Integrity

Data should not be corruptible / manipulatable

Availability

System should remain impervious to, e.g., (D)DoS attacks

Security Requirements with Misuse Cases



Managing Requirements and Handling Change

Requirements are Prone to Change

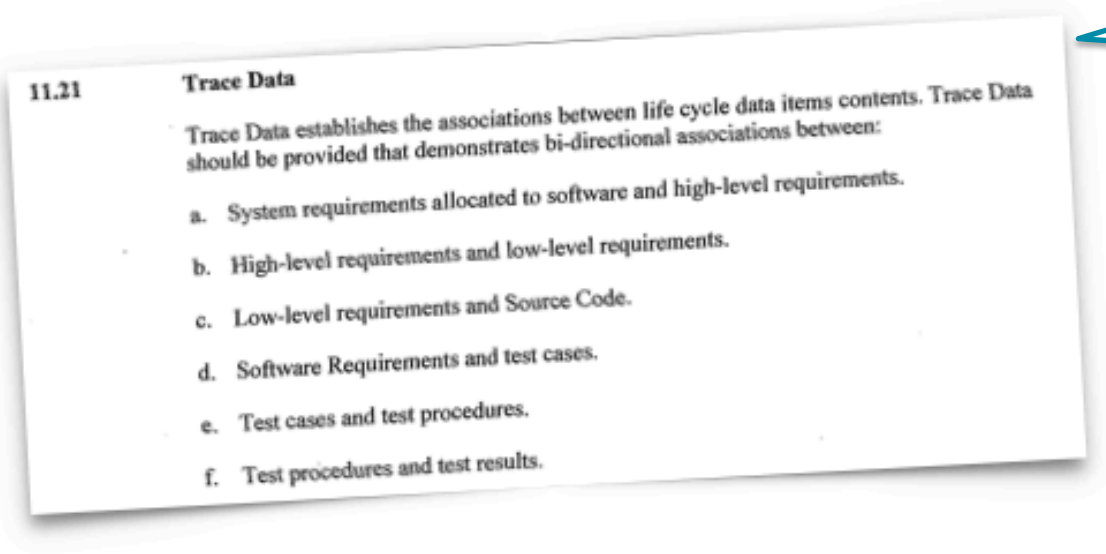
- Changes to requirements can be expensive to address
- They affect every stage of development and testing
- Need to be accommodated in development process
- Rationale for Agile techniques

Tom Scott
Youtube: *The Problem with Time & Timezones – Computerphile*

Tracing Requirements

- Linking requirements to development artefacts
  - What code modules and tests are relevant?
  - If a requirement changes, which parts of the system are affected?
- A key factor for certification

DO178-B/C



11.21 Trace Data

Trace Data establishes the associations between life cycle data items contents. Trace Data should be provided that demonstrates bi-directional associations between:

a. System requirements allocated to software and high-level requirements.

b. High-level requirements and low-level requirements.

c. Low-level requirements and Source Code.

d. Software Requirements and test cases.

e. Test cases and test procedures.

f. Test procedures and test results.

Traceability Matrix

Dependencies

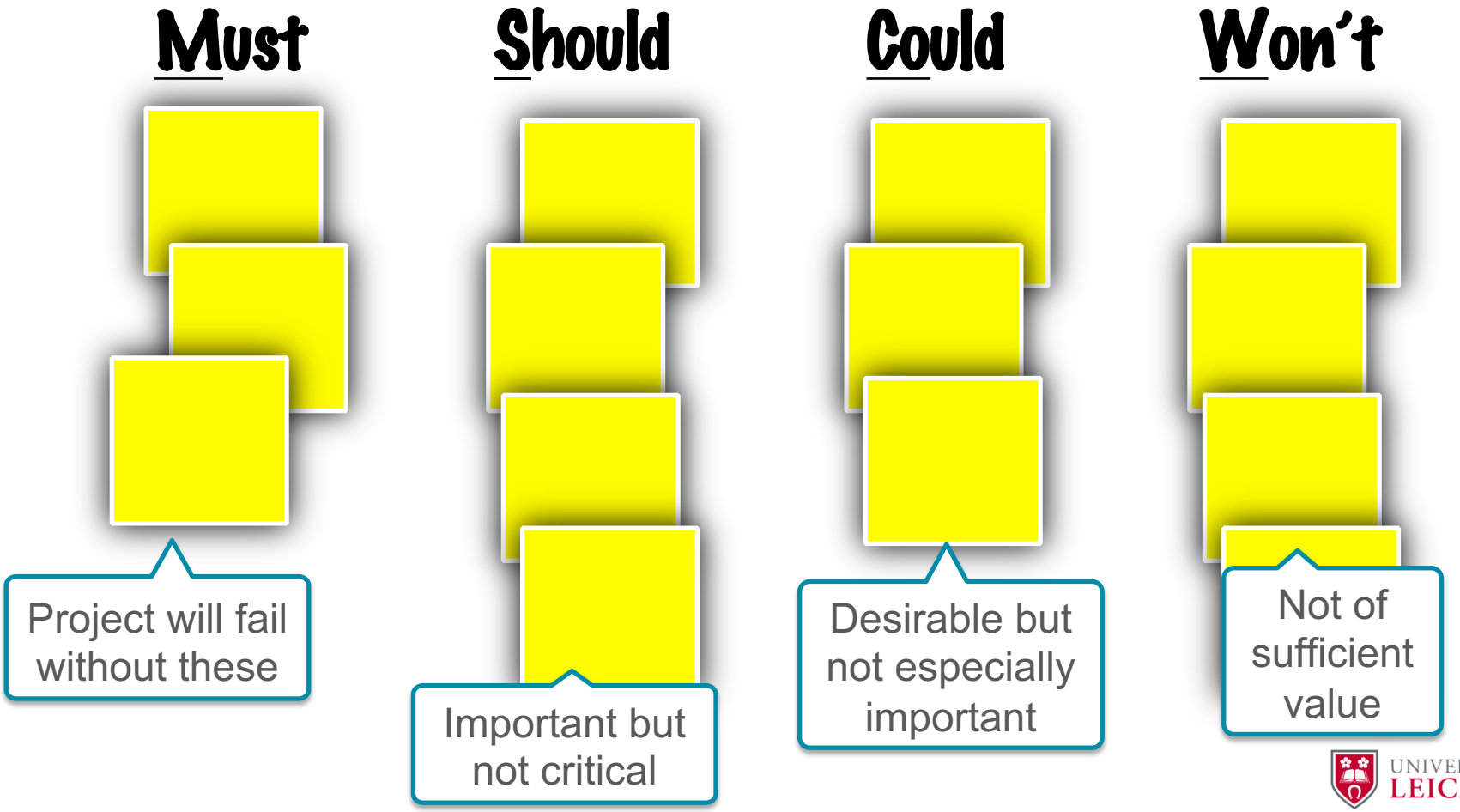
Requirement ID	Design Artefacts	Source code	Tests	Deps.
A.0.1: Develop storage format	Scenarios save file and load file	Package org.processor.xml	WriterTest.testSave, ReaderTest.testLoad	D 0.2
B.2.1: Develop file reader	load file scenario	Reader.java	ReaderTest.testLoad	A0.1
C.2.0: Develop file writer	save file scenario	Writer.java	WriterTest.testSave	A0.1
D.0.2: Develop core data structure	Scenarios save file and load file	Package org.processor.model	WriterTest.testSave, ReaderTest.testLoad	

Less common in Agile projects, where overhead of maintaining traceability links can become obstructive

Prioritisation

MoSCoW

Arrange requirements into four categories



The Kano Model

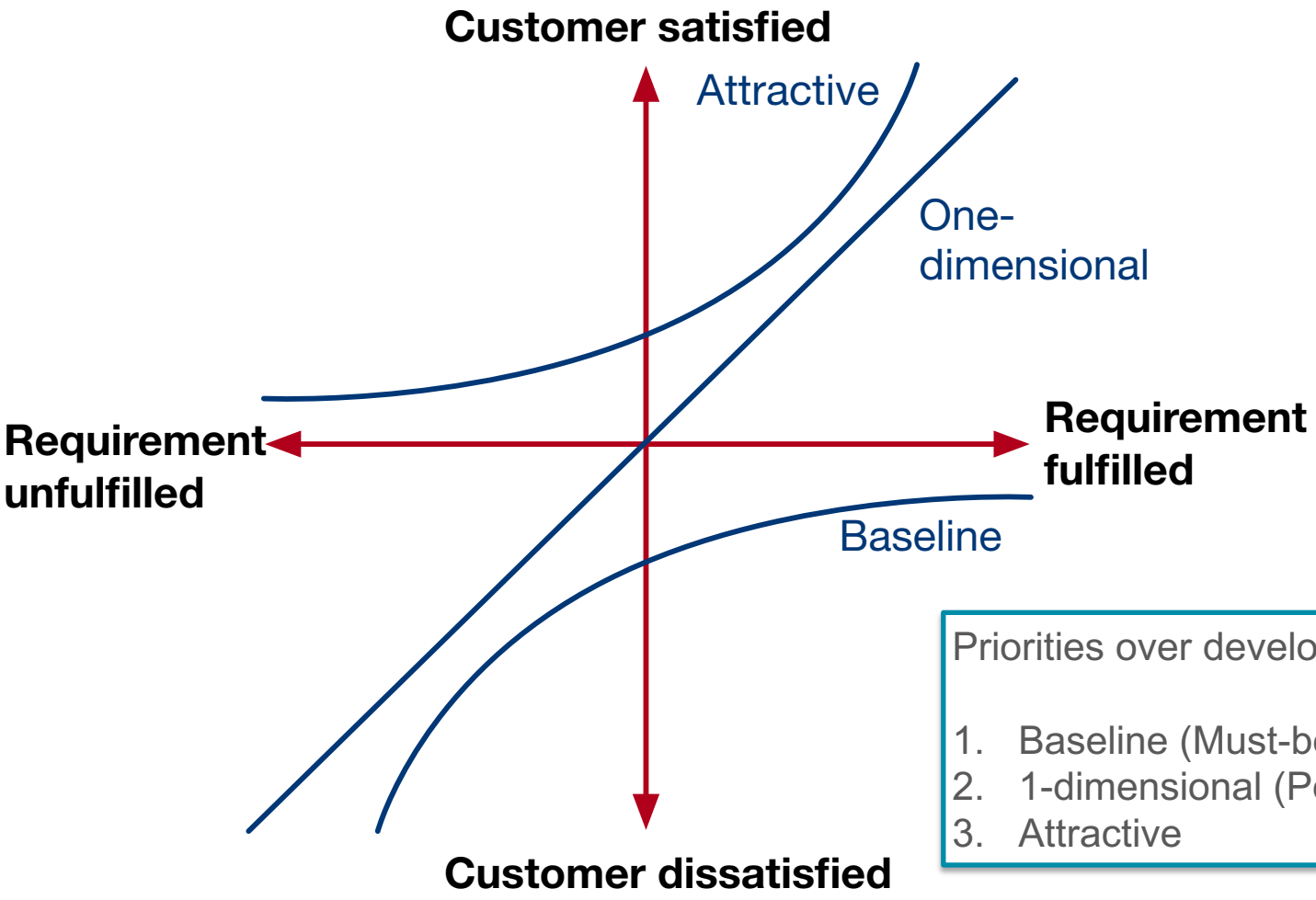
Requirements are divided into categories:

Baseline: If not fulfilled, will cause the user to be extremely dissatisfied. Since taken for granted, their fulfilment will however not significantly increase satisfaction.

One-dimensional: Requirements where extent to which they are fulfilled has a direct effect on satisfaction. If not implemented, user will be dissatisfied. If implemented, user will be very satisfied.

Attractive: Neither expected nor expressed by the customer, but have a strong impact on satisfaction if implemented.

Model of Customer Satisfaction



Priorities over development:

1. Baseline (Must-be)
2. 1-dimensional (Performance)
3. Attractive

Summary

- Development revolves around ability to capture requirements.
- Intrinsically difficult to express and capture.
- Use cases are most popular format, but can also be expressed in more detailed forms (e.g. SRS).
- Misuse cases can be used to express security-related use-cases.
- Potentially security issues can be flagged up by C-I-A.
- Traceability is generally important to link requirements to source code.
- Prioritisation can be achieved by MoSCoW and KANO.