# Specification, Part 1

# System Specification

Lesson 2

System Engineering

ENGINEERING COLLEGE
OF AARHUS

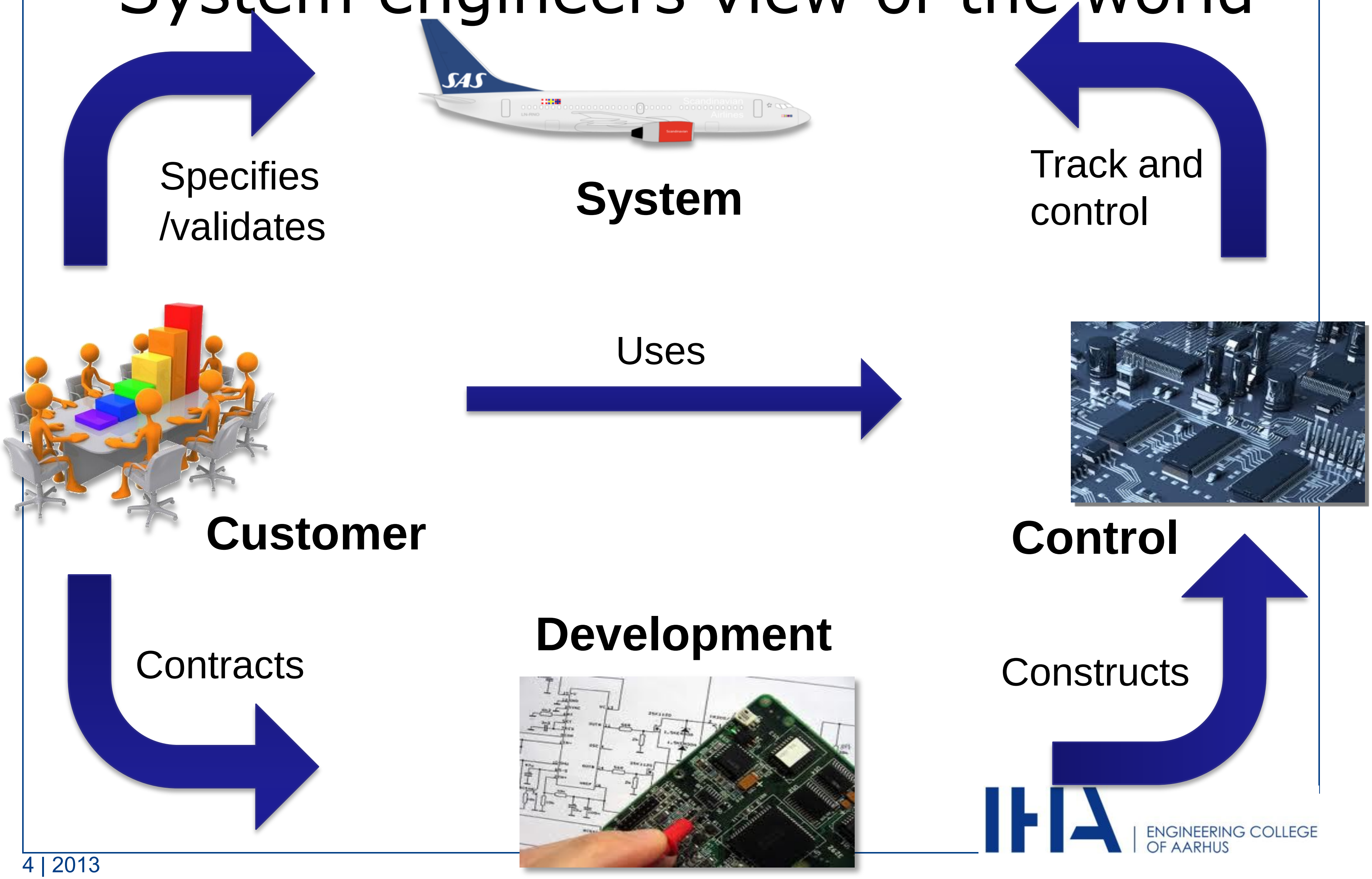
# Agenda

- System Specification
- Requirements Specification
- Types of requirements
- Specifying requirements
- Finding requirements (Elicitation)
- Good / Bad Requirements
- Traceability

# Agenda

- System Specification
- Requirements Specification
- Types of requirements
- Specifying requirements
- Finding requirements (Elicitation)
- Good / Bad Requirements
- Traceability

# System engineers view of the world



Specifies /validates

System

Track and control

Uses

Customer

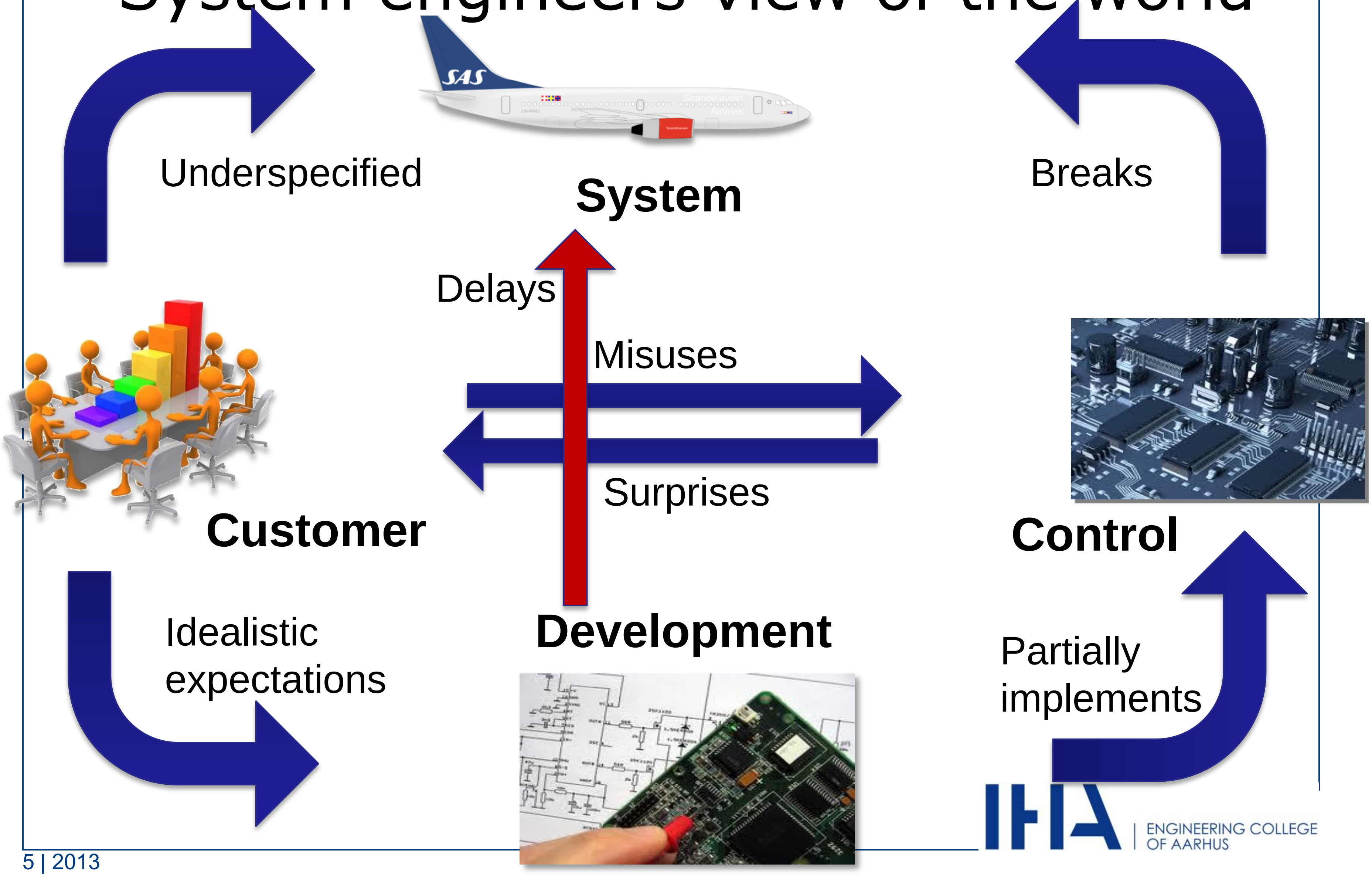
Control

Contracts

Development

Constructs

# System engineers view of the world



System

Underspecified

Breaks

Customer

Control

Development

Delays

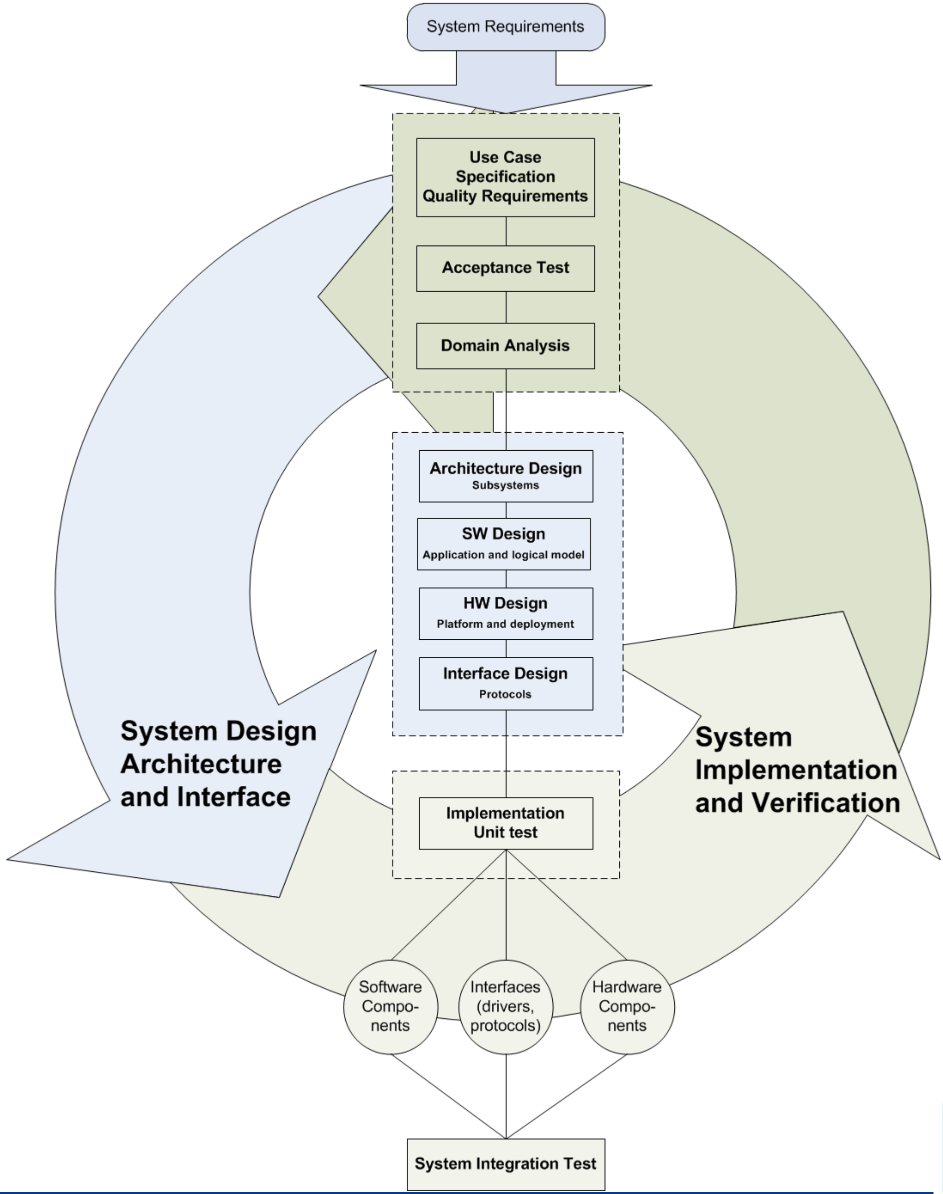
Misuses

Surprises

Idealistic expectations

Partially implements

## The System Engineering Process



# What is a system specification?

- “a statement that identifies a capability, characteristic, or quality factor of a system in order for it to have value by a user or a customer to solve a problem or achieve an objective”

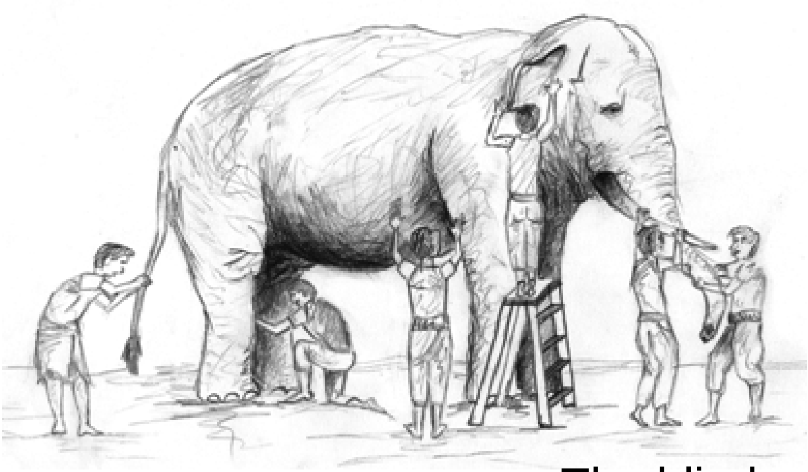
  Ralph Young, *Requirements Engineering Handbook, 2004*

- Do the right thing
- The ***what*** and not the ***how***

# What is a system specification?

- The Stakeholders description of a desire functionality or behavior for a system to achieve an objective.
- Functional requirements state what the system is required to do.
- The primary input to the design process
- The baseline against which acceptance tests are carried out.

# Why have a specification?



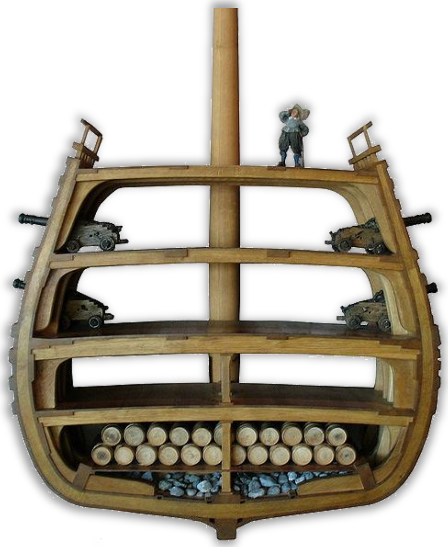
The blind men and the elephant

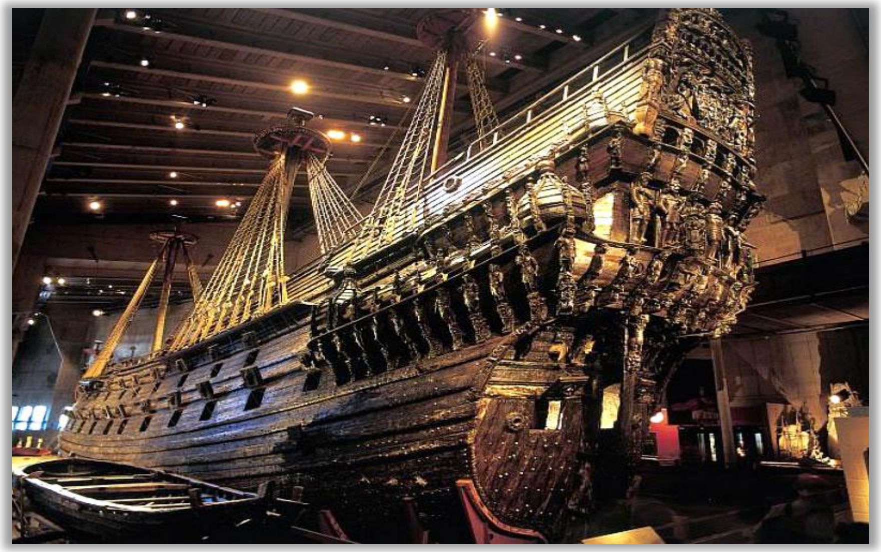
- Define requirements through exact statements determining the essential characteristics or particular needs to be satisfied.
- Establish a common language to facilitate communication
- Avoid misunderstandings, establish coherence
- Basis for cost and schedule estimates

# Why have focus on specification?

- Time, Effort, Price
- Correcting or changing functionality following specification
  - 3 x as much during the design phase
  - 5-10 x as much during implementation
  - 10-100 x as much after release

- Vasa, *10. aug. 1628 – †10. aug. 1628
- L: 69m H: 52m B: 11,7m D: 5m
- Requirements creep
- Inexperience
- Bad test conditions
- Schedule Pressure





[Why the Vasa Sank: 10 Problems and Some Antidotes for Software Projects, *IEEE Software*, Fairley03]

# DABHS

- Automated Baggage Handling
- Extremely advanced
- Change requests
- Test conditions
- Physical Constraints

[A Case Narrative of the Project Problems with the DABHS, Donaldson, *SFC, 2002*]

# LASCAD

- 7 million people, 600 square miles, 2500 calls daily, 1 year delivery time
- Reuse of hardware
- No training
- No focus on QA
- No defined ownership
- No load test
- Requirements document described the *how* instead of the *what.*
- [A Comedy of Errors: the London Ambulance Service case study, Finkelstein, 2002]

- One requirement was:
  **“the nearest available ambulance shall be dispatched to the incident.”**

- Did not respect rescue stations or precinct.

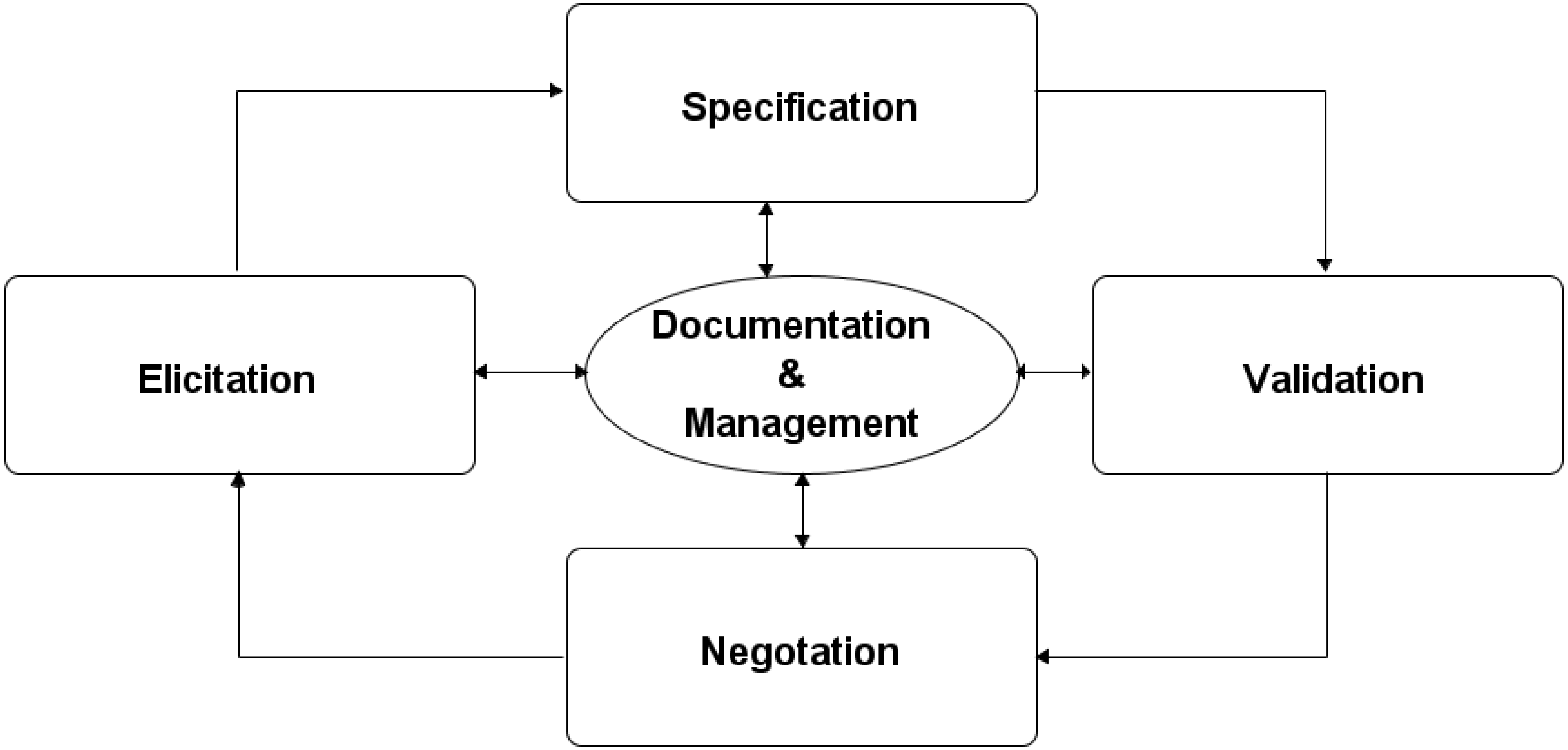
# What to derive from these cases?

- A specification is more than functional requirements
- The specification should attempt to address:
  - Requirements creep / Requirements Change
  - Test conditions
  - Training
  - Realistic Schedule
  - External Constraints (physical, legal)
  - Ownership and Stakeholders
  - Existing systems/hardware

# Who uses a specification

| Stakeholder | Use of specification |
|---|---|
| Customer | Fullfillment of business goal<br>Contract |
| Manager | Scheduling<br>Progress measuring |
| System Engineer | Design<br>Functionality<br>Constraints |
| Test Engineer and QA personnel | Test planning<br>Verification<br>Validation |

# Specification Process



# Requirements Specification

- Documentation of the specification
- Principally the outcome of elicitation
- Complete description of the behavior and constraints of the system to be developed
- Baseline for communication between stakeholders
- Often has high demands for versioning and traceability

# Kinds of Requirement Specifications

- Ranges from high-level abstract statements to detailed mathematical specifications
  - Textual Specification
  - Structured Natural Language
  - Graphical Specification (e.g. SysML)
  - Formal Specification

# Textual Requirement Specifications

- Defined in word processor
  - Word, OpenOffice, NeoOffice, IWork, etc.
- Textual + diagrams and illustrations
  - Use Cases (Mainly For Functional Requirements)
  - UML / SysML
- MoSCoW Method (prioritisation technique)

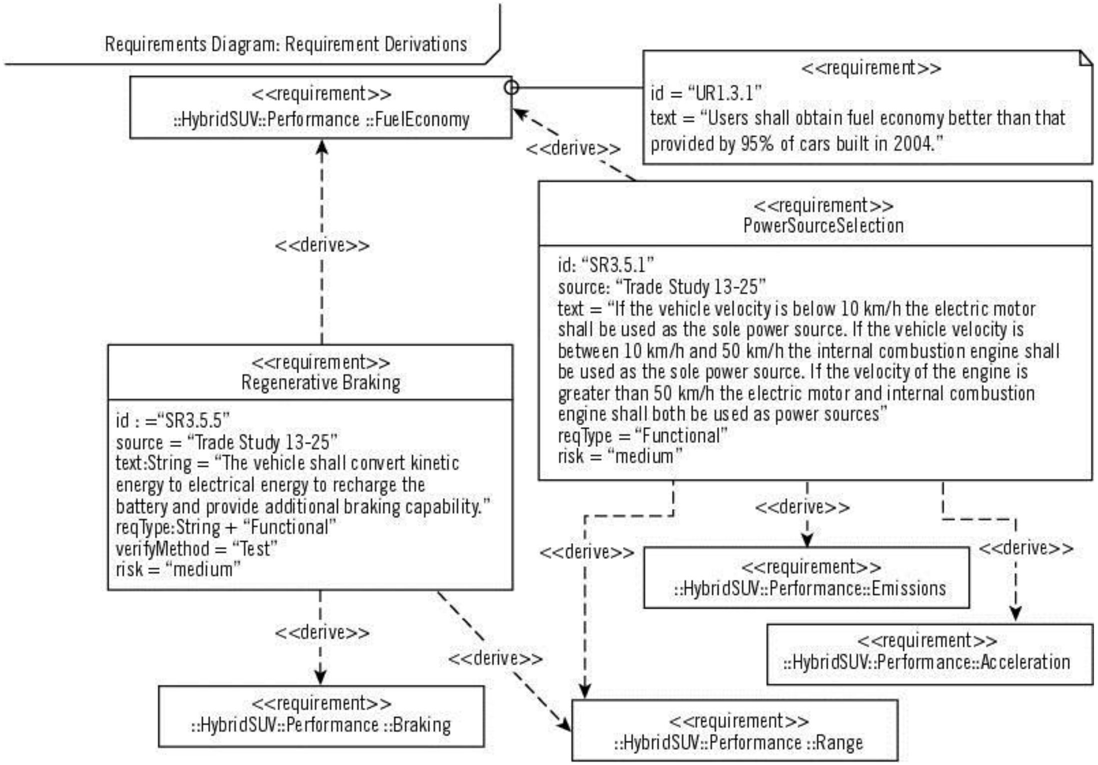
  **M** - MUST have this.
  **S** - SHOULD have this if possible,
  **C** - COULD have this if it does not affect anything.
  **W** - WON'T have this time, but WOULD like in the future

# Structured Natural Language

- Forms/templates are used to establish rigour and structure
- Defining standard forms or templates to express the requirements
  - Input/Output
  - Pre/Post condition

# Graphical Specifications

- SysML Requirement Diagram
  - Identifies requirements hierarchies and derivation



Requirements Diagram: Requirement Derivations

<<requirement>>
::HybridSUV::Performance ::FuelEconomy

<<requirement>>
id = "UR1.3.1"
text = "Users shall obtain fuel economy better than that provided by 95% of cars built in 2004."

<<requirement>>
PowerSourceSelection
id: "SR3.5.1"
source: "Trade Study 13-25"
text = "If the vehicle velocity is below 10 km/h the electric motor shall be used as the sole power source. If the vehicle velocity is between 10 km/h and 50 km/h the internal combustion engine shall be used as the sole power source. If the velocity of the engine is greater than 50 km/h the electric motor and internal combustion engine shall both be used as power sources"
reqType = "Functional"
risk = "medium"

<<requirement>>
Regenerative Braking
id : ="SR3.5.5"
source = "Trade Study 13-25"
text:String = "The vehicle shall convert kinetic energy to electrical energy to recharge the battery and provide additional braking capability."
reqType:String + "Functional"
verifyMethod = "Test"
risk = "medium"

<<requirement>>
::HybridSUV::Performance::Emissions

<<requirement>>
::HybridSUV::Performance::Acceleration

<<requirement>>
::HybridSUV::Performance ::Braking

<<requirement>>
::HybridSUV::Performance ::Range

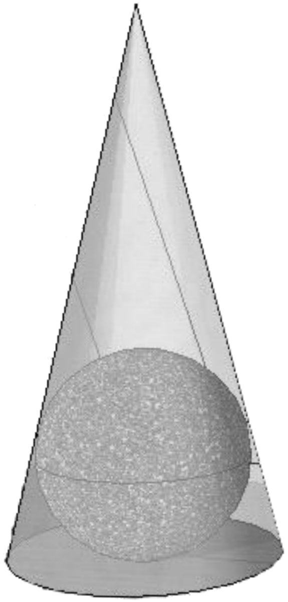
# Formal Specification (1/2)

- Written language is contradictory and ambiguous
  - Ambiguous sentences
    - He gave her cat food
  - Contradictory sentences

    The sentence below is true
    The sentence above is false

# Formal Specification (2/2)

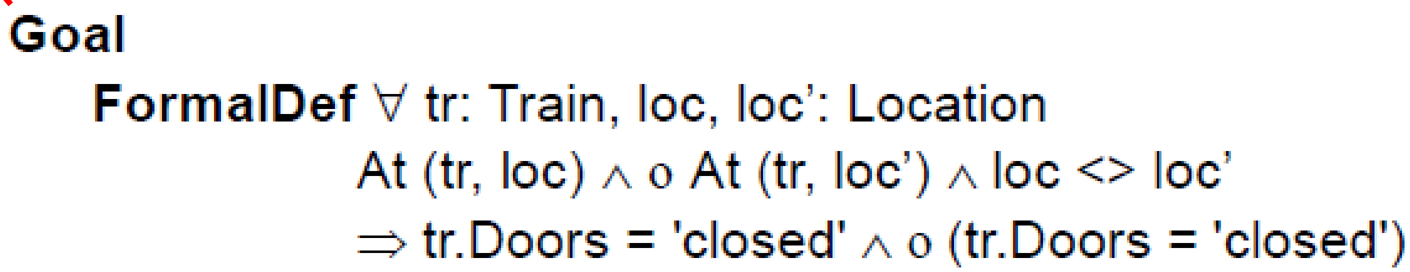
> Any sphere is equal to four times the cone which has its base equal to the greatest circle in the sphere and its height equal to the radius of the sphere *Archimedes*



- $V = \frac{4}{3}\pi r^3$
- Formal specifications are precise and unambiguous
  - Mathematically based
  - Can be proved by mathematical analysis (CASL, Z, VDM, LTL)

# Formal Specification - Example

Not curriculum



**Goal**

**FormalDef** $\forall$ tr: Train, loc, loc': Location

$\text{At (tr, loc)} \wedge \circ\ \text{At (tr, loc')} \wedge \text{loc <> loc'}$

$\Rightarrow \text{tr.Doors = 'closed'} \wedge \circ\ \text{(tr.Doors = 'closed')}$

For all trains it must hold that for each train which; is at a location and has a next location and these locations are not the same, it implies, that the train doors are closed and stay closed.

# Types of requirements

- Functional
  - What the system should do (behaviours)
- Non-functional
  - Qualities or criteria of the system, rather than specific behaviour
- Other categorizations exists:
  - Domain Requirements (Business Rules)

# Functional Requirements

- Functional
  - Describes system services
  - Requirement for each input and output
  - Behavioural requirements
  - Use Cases

# Exercise

- Who would the stakeholders be ?
- What type of specification would you use?

# Quality Demands/Non-functional Requirements

- Qualities or Constraints on the services or functions offered by the system

> Qualities are properties or characteristics of the system that its stakeholders care about and hence will affect their degree of satisfaction with the system.
> *[Defining Non-Functional Requirements,* Malan01]

- NFRs should satisfy two attributes
  - Should be objective
  - Must be verifiable (measurable metrics)

# Types of requirements

- FURPS+ (Robert Grady, Hewlett-Packard)

  **F**unctionality

  **U**sability

  **R**eliability

  **P**erformance

  **S**upportability

  + (Design and Physical constraints, Interfaces, Legal, Test, **Reuse, Economic constraints, Aesthetics, Comprehensibility, Technology tradeoffs)**

- Others are:

  McCall, Boehm, Dromey

# Functionality

- Characteristics
  - Behaviour
  - Security

# Usability

- Characteristics
  - Operability
  - Accessibility
  - User Interface (If any)
  - Documentation
- Metrics
  - ~~The system should be easy to use~~
  - Max. number of errors made by users for a specific task over a time period.
  - Time to learn a certain functionality

# Reliability (1/2)

- Characteristics
  - Reliability
    - is the probability of a system to perform a required function under stated conditions
  - Availability
    - is a function of how often failures occur, repair time and maintenance interval.
  - Maintainability
    - is the ability of a system to or restored to a specified condition

# Reliability (2/2)

- Metrics
  - Reliability
    - Mean time between failure (MTBF)
  - Availability
  - Maintainability
    - Mean time to restore (MTTR)

# Performance

- Characteristics
  - Throughput
  - Response time
  - Start-up time
  - Capacity & Efficiency Constraints
- Metrics
  - Time
  - Specifics is out of scope for this course
  - *95% of the transactions shall be processed in less than 1 second at 80 % load*

# Supportability

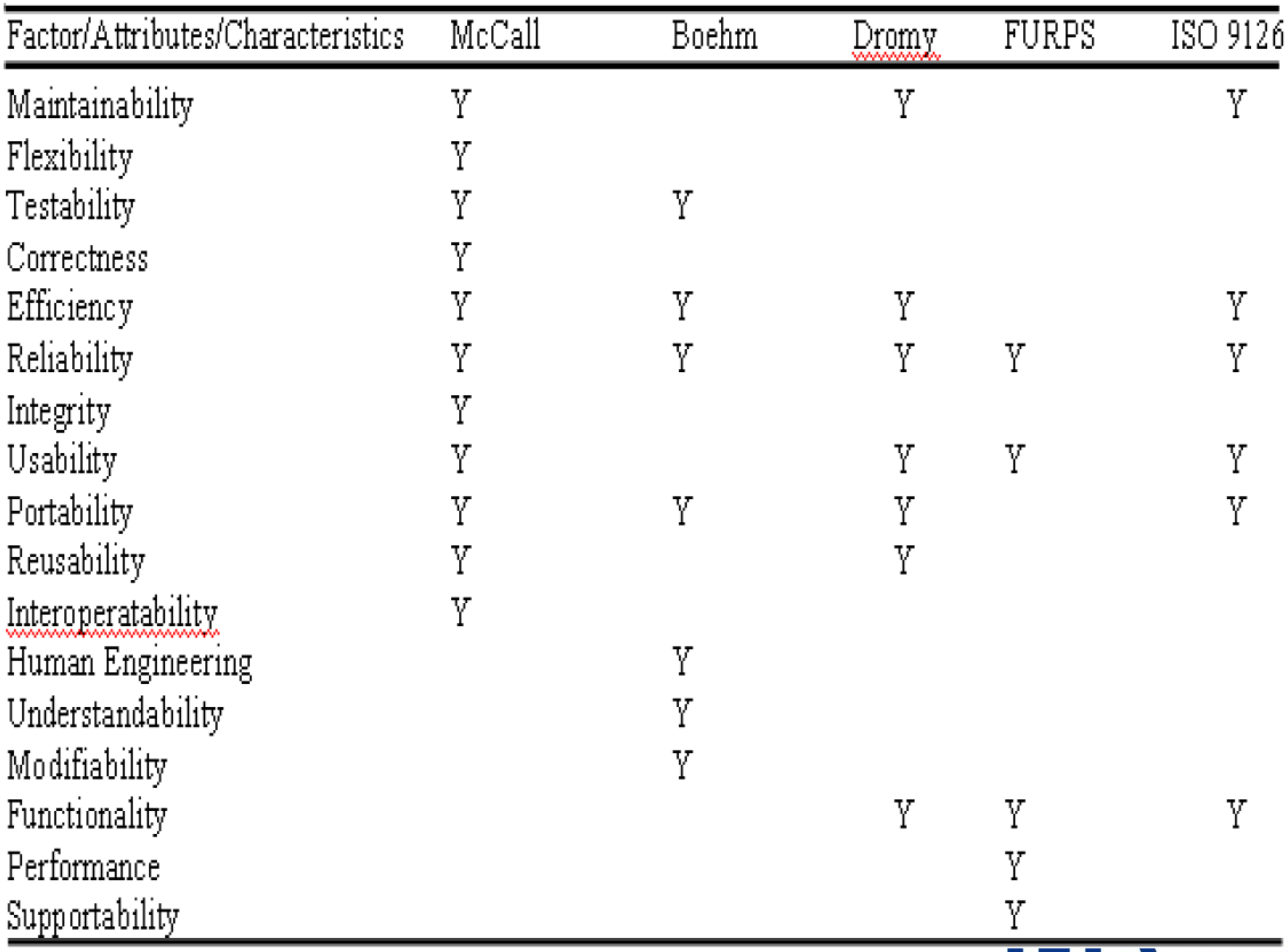
- Characteristics
  - Compatibility,
  - Installability,
  - Localizability,
  - Maintainability
- Metrics
  - Same as with Usability
  - Measure of success under specified scenarios

ENGINEERING COLLEGE
OF AARHUS

+

- Legal
  - Data Protection Act, Health and Safety act,
- Technology trade off
  - Balance between two incompatible features
- Test
  - Conditions, Environment, Access
- Implementation
  - Platforms, Hardware, Software
- Reuse
  - Existing systems, parts, modules
- Environmental
  - RoHS, WEEE, etc.

# Types of requirements



| Factor/Attributes/Characteristics | McCall | Boehm | Dromy | FURPS | ISO 9126 |
|---|---|---|---|---|---|
| Maintainability | Y | | Y | | Y |
| Flexibility | Y | | | | |
| Testability | Y | Y | | | |
| Correctness | Y | | | | |
| Efficiency | Y | Y | Y | | Y |
| Reliability | Y | Y | Y | Y | Y |
| Integrity | Y | | | | |
| Usability | Y | | Y | Y | Y |
| Portability | Y | Y | Y | | Y |
| Reusability | Y | | Y | | |
| Interoperatability | Y | | | | |
| Human Engineering | | Y | | | |
| Understandability | | Y | | | |
| Modifiability | | Y | | | |
| Functionality | | | Y | Y | Y |
| Performance | | | | Y | |
| Supportability | | | | Y | |

# Exercise 2

- Find NFR / Quality demands for this coffee machine using (F)URPS+

  **U**sability

  **R**eliability

  **P**erformance

  **S**upportability

# Does this fit in FURPS+ ?

- For each boat, the elapsed time is defined as the difference, in seconds, between the race start time and the boat’s finish time.
- Personal information about the students can be accessed only by those who need that information in student administration.
- The system must be developed using the XYZ suite of CASE tools.

# Level of requirements

- Goal-level requirements
- Domain-level requirements
- Product-level requirements
- Design-level requirements

# Goal-level requirements

- Specified overall goals of the system
- Often called “Business goal”
- Example:

  “The product shall ensure that pre-calculations match actual costs within a standard deviation of 5%.”

- Can easily be verified – late in development process
- Impossible to implement solution based on business goals only

# Domain-level requirements

- How the system should support the environment/domain
- Example:

  “The product shall support the cost registration task, including recording of experience data.”
- High level of domain knowledge is needed to implement domain-level requirements

# Product-level requirements

- Typical functional requirements
- Example:
  “The product shall have a function for recording experience data and associated keywords.”
- Easy to implement with limited domain knowledge

# Design-level requirements

- Often used to precisely specify system interfaces
- Only how the interface should look – not how it should be implemented
- Example:

  “The product shall provide the screen pictures shown in app. X.”
  “RS-232 must be supported for legacy”
- To precise descriptions limits the possible solutions
- Very easy to test design-level requirements

# Documentation approaches

- Documents
- Requirements management tools
  - IBM Rational DOORS
  - Borland CaliberRM
- Specification Languages

# Example outline of SRS (1/6)

(By C.Kruegel, UCSB, Based on IEEE Recommended Practice)

Table of Contents

1. Introduction

    1.1 Purpose

        Purpose of the SRS

        Intended audience of the SRS

    1.2 Scope

    1.3 Definitions, acronyms, abbreviations

    1.4 References

    1.5 Overview

# Example outline of SRS (2/6)

2. Overall description
    2.1 Product perspective

    2.2 Product functions

# Example outline of SRS (3/6)

3. Specific requirements (these are the detailed requirements)

   3.1 External interface requirements

   System interfaces, user interfaces, hardware interfaces, software interfaces, communications interfaces, etc.

   A detailed description of all inputs and outputs from the software system should be given.

# Example outline of SRS (4/6)

## 3.1.1 User interfaces

Screen formats, page or window layouts, error messages, etc. Some sample screen dumps can be used here to explain the interface

## 3.1.2 Hardware interfaces

Interface between hardware and software product, which devices are supported

## 3.1.3 Software interfaces

Specify use of other software products and interfaces with other application systems

## 3.1.4 Communication interfaces

Interfaces to communications such as local network protocols, etc.

# Example outline of SRS (5/6)

3.2 Functional requirements

Use cases

***This section is very important.***
***Ensure requirements are unambiguous, complete and consistent***

# Example outline of SRS (6/6)

3.3 Performance requirements
Speed, availability, response time

3.4 Design constraints
Required standards, implementation language restrictions, resource limits, operating environment(s) etc.

3.5 Software system attributes
Attributes such as security, portability, reliability

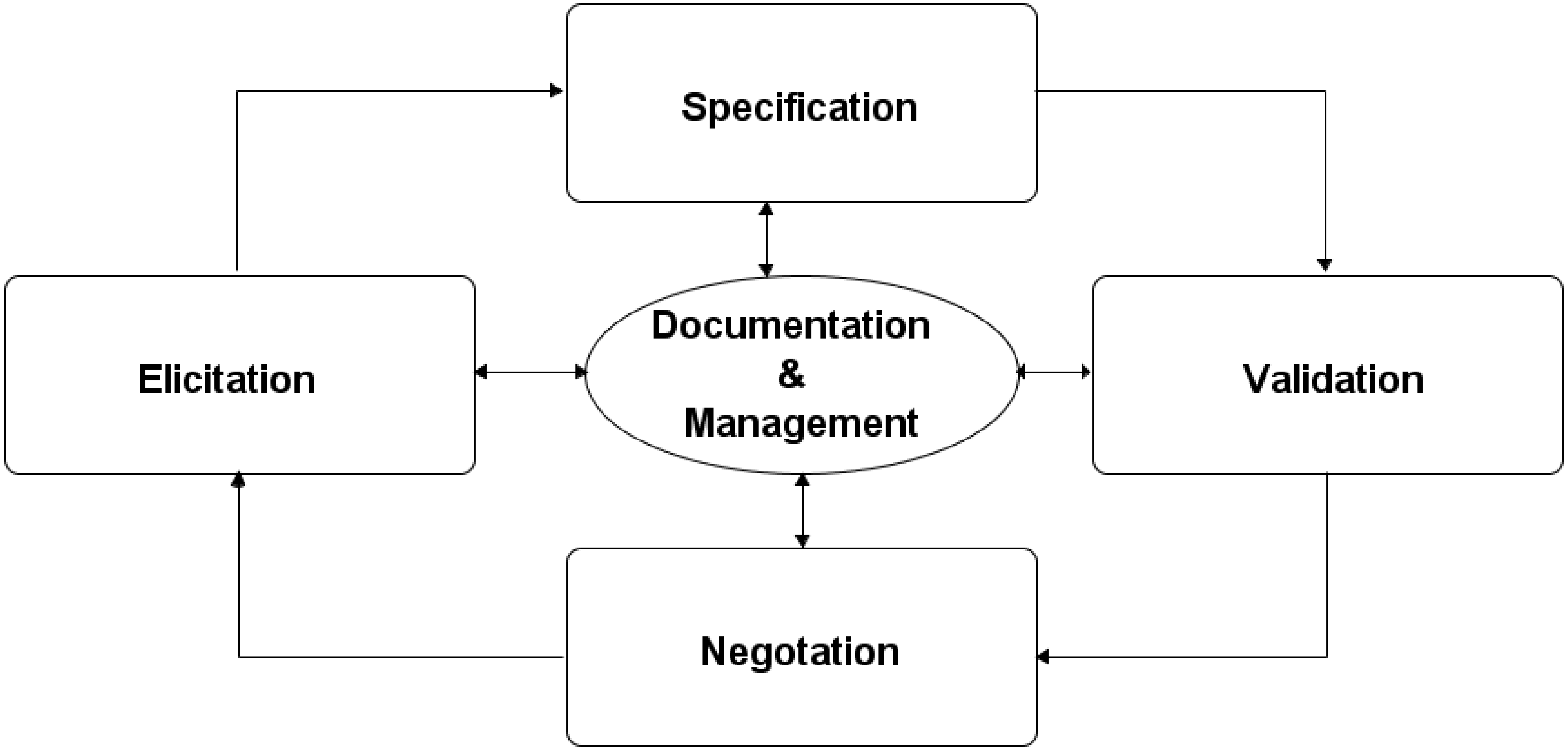
3.6 Domain requirements
Explain the application domain and constraints on the application domain

4 Appendices
Any other important material

# Finding Requirements (Elicitation)

- Formulating overall goal of the system (mission statement)
- Describing current work process and problems
- Not sequential process – but iterative
- Elicitation is concerned with where requirements come from and how they can be collected

# Specification Process

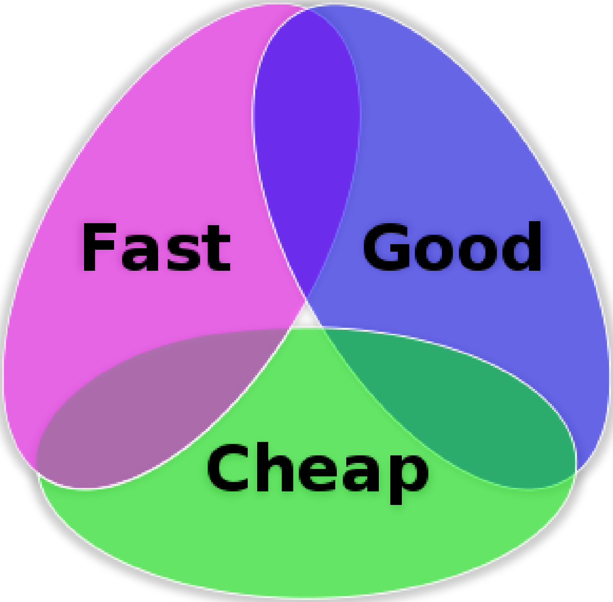


# Challenges in Elicitation

- Users do not know what they want, or they know what they want but cannot articulate it
- Stakeholders come up with solution – not demands
- Different stakeholders have conflicting views
- To many “nice to have” requirements are specified
- The “Yes, But” syndrome

# Challenges in Elicitation

- The larger the project, the more difficult it is to grasp
- Users’ expectations are often unrealistic, and compromises must be made between economy and features



# Approaches in Elicitation

- Setting goals
  - Decide how to analyze data once collected
- Relationship with participants
  - Clear and professional
- Triangulation
  - Use more than one approach

# Requirements Elicitation Techniques

- Techniques
  - Interview
  - Stakeholder analysis
  - Brainstorm
  - Prototype
  - Requirements Workshop
  - Task Demonstration
  - Role Playing

# Interview

- Simple direct technique
- Context-free questions can help achieve bias-free interviews
- Convergence on some common needs will initiate a “requirements repository” for use during the project.
- A questionnaire is no substitute for an interview.

# Stakeholder analysis

- Who are the stakeholders?
- What are their goals?
- Which risks and costs do they see?

# Brainstorm

- Brainstorming involves both idea generation and idea reduction.
- The most creative, innovative ideas often result from combining, seemingly unrelated ideas.
- Various voting techniques may be used to prioritize the ideas created.

# Prototype session

- Preliminary model built for demonstration purposes
- The customer may be more likely to view the prototype and react to it, than to read the SRS and react to it.
- The prototype provides quick feedback.
- Prototype displays unanticipated aspects of the systems behavior

# Requirements Workshop

- Users and developers cooperate to analyse and design
- Mixture of brainstorm and prototype sessions
- Do not “attack” other members.
  Do not get on a soap box.

# Task demonstration

- Difficult to understand workflows/processes otherwise
- Task specific observation
  - Think-out-loud
  - Measure time used
  - Measure errors made
  - Count number of keystrokes

ENGINEERING COLLEGE OF AARHUS

# Role Playing

- Role playing allows stakeholders to experience the user’s world from the user’s perspective

# Good/Bad Requirements

- Foundation of the Project
- Usually the problem lies in a lack of communication and comprehension between the customer and the developer

# Good Requirements

- Correct – specifying something actually needed
- Unambiguous – only one interpretation
- Complete – includes all significant requirements
- Consistent – no requirements conflict
- Verifiable – all requirements can be proven by test
- Modifiable – changes can easily be made to the requirements
- Traceable – the origin of each requirement is clear

# Bad Requirements

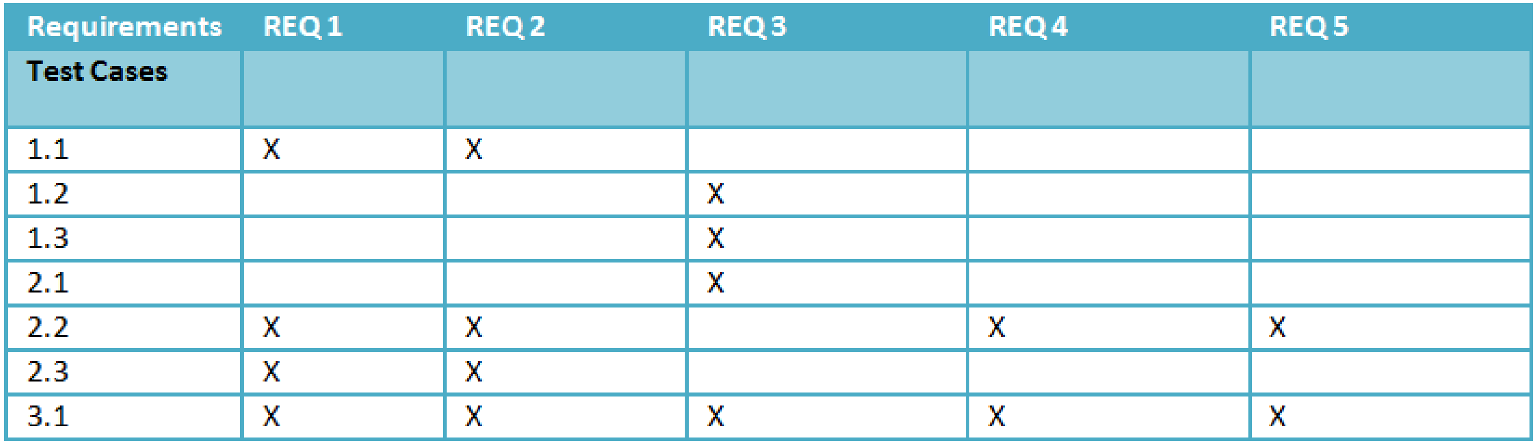
- Incorrect – specifying something not needed
- Ambiguous – multiple interpretations
- Incomplete – excludes significant requirements
- Inconsistent – requirements are contradicting
- Unverifiable – requirements cannot be proven
- Unmodifiable – requirements are carved in stone
- Untraceable – requirement has no origin

# Traceability

- Traceability is concerned with the relationships between requirements, their source plus the system design and implementation
- Requirements have unique IDs which can be referenced
  - Forward traceability
    - references requirements to products/features/test
  - Backward traceability
    - references products/features/test to requirements

# Requirements Traceability Matrix

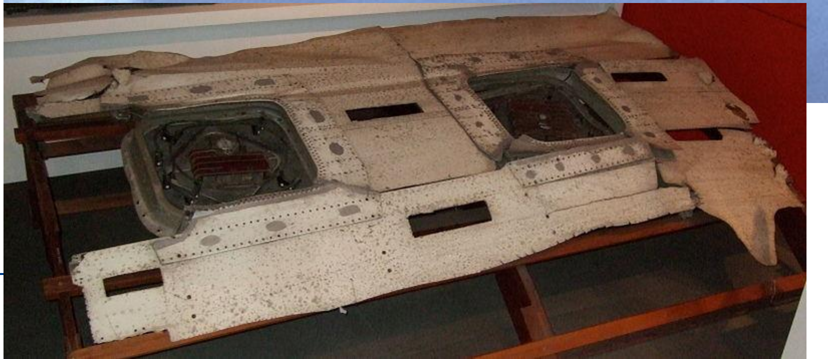
- Determine the two-way mapping between Requirements and Features/Test
- Are all features mapped to a requirement? And are each requirements fulfilled by a feature?



| Requirements | REQ 1 | REQ 2 | REQ 3 | REQ 4 | REQ 5 |
|---|---|---|---|---|---|
| Test Cases | | | | | |
| 1.1 | X | X | | | |
| 1.2 | | | X | | |
| 1.3 | | | X | | |
| 2.1 | | | X | | |
| 2.2 | X | X | | X | X |
| 2.3 | X | X | | | |
| 3.1 | X | X | X | X | X |

# de Havilland Comet

- 1951, First commercial jet
- 16,000 pressure cycles
- Crash 1952, 53, 2 x 54
- Test equivalent to 3000 flights
- Metal fatigue



A specification that will not fit on one page of 8.5x11 inch paper cannot be understood.

*Mark Ardis*

Professor
Rochester Institute of Technology

# Questions