

Specifications of IT systems

Jens Bennedsen, Peter Gorm Larsen and Joey Coleman
(jbb@iha.dk), (pgl@iha.dk) and (jwc@iha.dk)

Aarhus University,
School of Engineering/Department of Engineering

Agenda

- Administrative information about the course
 - Specifications and different kinds of systems
 - Structure of a Software Requirement Specification
 - Different levels of requirements
 - Requirements Elicitation
 - The Tendering Process

Who are the teachers ?

- **Professor Jens Bennedsen**; MSc, PhD
- 20+ years of professional experience
 - 12 years with “Datamatikeruddannelsen”
 - 2 years with LEC
 - 6 years with Aarhus University/IT University West
 - 4 years with the Aarhus University, School of Engineering
- Object-oriented languages, techniques and teaching



Who are the teachers?

- **Professor Peter Gorm Larsen; MSc, PhD**
- 20+ years of professional experience
 - ½ year with Technical University of Denmark
 - 13 years with IFAD
 - 3,5 years with Systematic
 - 8 ½ years with IHA/Aarhus University
- Consultant for most large defence contractors on large complex projects (e.g. Joint Strike Fighter)
- Relations to industry and academia all over the world
- Has written books and articles mostly about VDM
- See <http://pglconsult.dk/private/peter.htm> for details



Who is the teacher (one week)?

- **Joey Coleman**; MPhil, PhD
- Undergrad study at Ryerson in Toronto, Canada
- 2 years in industry (including a 'dotcom' startup)
- 8 years in Newcastle, UK
 - MPhil — Research-based degree
 - PhD — Reasoning about concurrent systems
- COMPASS project — Systems of Systems modelling
- Programming Language Semantics, Formal Modelling



Contacting Details

- The most convenient way – email
- jbb@iha.dk and pgl@eng.au.dk
- Or see us in our offices.
- Our offices are:
 - JBB: Dalgas Avenue 2, Room 335
 - PGL: Edison, Room 308

The Instructor

- Peter W. V. Jørgensen
- PhD student
- Department of Engineering
- Edison Room 325
- Email: pvj@eng.au.dk



Teaching Material (1/2)

- Books
 - Søren Lauesen: *Software Requirements: Styles and Techniques* pages: 1 - 40, [41-70], 71-152, [153-216], 331 – 438
 - Søren Lauesen: *Guide to requirements SL-07 - template with examples*. **OR** Søren Lauesen: *Vejledning til kravskabelon SL-07 - fra behov til løsning*.
- Extract from a book
 - Modelling Systems by Fitzgerald and Larsen
- Examples
 - Exemplary Requirements Specification(s)

Teaching Material (2/2)

- Software for Dependable Systems: Sufficient Evidence? Daniel Jackson, Martyn Thomas, and Lynette I. Millett, Editors, Committee on Certifiably Dependable Software Systems, National Research Council, pp 1—15.
- Articles
 - Each group will need to read and evaluate papers
- Pamphlet
 - *Dr. Richard Paul and Dr. Linda Elder: The Miniature Guide to Critical Thinking: Concepts and tools. The Foundation for Critical Thinking*
- Standards and guidelines
 - IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998
 - Requirement specification templates

Course web pages

- All information concerning this course including lecture notes, assignments announcements, etc. can be found on the TISPIT web pages on campusnet
- You should check this site frequently for new information and changes. It will be your main source of information for this unit.
- Group solutions shall be uploaded to the exercises part of CampusNet.

Education Form

- Lectures and exercise hours
 - Mondays 12:15 – 14:00 in Edison 416
 - Thursdays 8:15 – 12:00 in Edison 416
- Mandatory exercises
 - Hand-ins most weeks on Saturdays at 16:00 CET before lectures
 - Done in groups with 3 participants (must be formed today and send by email to pjl@eng.au.dk)
- Combination of
 - Lessons teaching theory (both by teachers and students)
 - Weekly assignments where theory is turned into practice
 - Guest lectures
- Exam form
 - 15 minutes oral examination without preparation + 5 minutes for evaluation the XXX 2014
 - curriculum is all that we read

Learning Objectives

The participants must at the end of the course be able to:

- **Compare** alternative templates for requirements specification
- **Describe, compare, discuss** and **judge** specification techniques,
- **Describe** tendering processes and their components,
- **Compare** alternative requirement elicitation techniques,
- **Write** specifications.

Contents of the course

- Lectures
- Mandatory Hand-ins
- Groups need to select:
 - An existing specification for reviewing (1 group per specification)
 - A set of 3 articles related to specifications to be analyzed and written up in a survey paper (1 group per set of articles)
 - A formal VDM-SL specification to be explained and extended with an additional functionality (1 group per VDM-SL model)
 - A new case to write a specification for (max 2 groups for each)

Lectures in the course

Week	Lecture
1	Introduction and course overview Specifications and their contexts Requirements elicitation The tendering process <i>by PGL</i>
2	Scientific report writing (scope, story, organization, reading) <i>by JWC</i>
3	Formal specification techniques + specific presentation on VDM <i>by PhD student Luis Couto + PGL</i>
4	Task descriptions <i>by JBB</i>
5	Review and inspection technologies for specifications <i>by JBB</i>
6	Requirements work in the real world (Netcompany, Mjølner, Terma, Vestas, CGI) Certification <i>by JBB</i>
7	Repetition, course evaluation and round off <i>by PGL & JBB</i>

MANDATORY Hand-ins during the course

Week	Due date	Hand-in
1	1/2/14	Groups must be formed and Validation of existing specification
2	8/2/14	Report about comparing 3 articles with approaches to requirements specification
3	15/2/14	Document with an explanation and extension of the selected formal VDM-SL specification
4		Nothing (but working on the assignment for the week after)
5	1/3/14	Document for tender (specification of external system)
6	6/3/14	Review report of other groups specifications
7	10/3/14	Final report

Existing Specifications to analyze

1. Contactless Smart Card based Multi-Utility System
2. Integrated Library System
3. Name Authority Service System
4. Real-Time Biosurveillance Program
5. AgentMom: A Multi-Agent System
6. AzureusHistory Plugin
7. Content Delivery Networks through Peering
8. Invisible Meeting Scheduler
9. Water Use Tracking
10. Beamline Operating System
11. Cafeteria Ordering System
12. Criteria Action Table
13. Open Source Web Archive Tools
14. Sustaining the Earths Watersheds: Agricultural Research DB
15. Financial Management System

Formal VDM-SL specifications

1. Bill of material
2. Cash dispenser
3. Gateway
4. Hotel
5. Metro
6. Monitor
7. Network Database
8. Tracker
9. Traffic light
10. Library
11. Country coloring
12. Worldcup

Systems awaiting - New Specifications

1. Poultry piece batching system
2. Digital Answering Machine
3. Skema planlægning (in Danish)
4. Hotspot events (in Danish)
5. Training Information System
6. Handling of exercises and reviews for discrete maths
7. Airport surveillance system
8. Intelligent Medical Compression stockings
9. ...

Possible Sets of Articles to Review

1. How Perspective-Based Reading Can Improve Requirements Inspections
2. Towards an Inspection Technique for Use Case Models
3. Assessing Defect Detection Methods for Software Requirements Inspections Through External Replication
4. Comparing Inspection Strategies for Software Requirement Specifications
5. Experiences from a Tender Process
6. Communication Gaps in a Tender Process
7. Software Inspections: An Effective Verification Process
8. Identifying and Measuring Quality in a Software Requirements Specification
9. Goal-Oriented Requirements Engineering: A Guided Tour
10. Specifying Software Requirements for Complex Systems: New Techniques and Their Application
11. Task Descriptions as Functional Requirements
12. Processing Natural Language Software Requirement Specifications
13. Inspection of software requirements specification documents: a pilot study
14. An experiment to assess different defect detection methods for software requirements inspections
15. An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development
16. Preventing Requirement Defects: An Experiment in Process Improvement
17. Four dark corners of requirements engineering
18. Problem Frames: a Case for Coordination
19. Architecture-driven Problem Decomposition
20. A reference model for requirements and specifications
21. Problem Frames and Software Engineering
22. Formal Specification: a Roadmap

Set 1:	1, 2, 3
Set 2:	4, 7, 8
Set 3:	9, 10, 11
Set 4:	13, 14, 16
Set 5:	12, 16, 17
Set 6:	6, 7, 12
Set 7:	9, 11, 19
Set 8:	20, 21, 22
Set 9:	10, 17, 22
Set 10:	3, 12, 15

Exam Questions

1. Contents of a good specification
2. Discuss pros and cons with informal specification
3. Discuss pros and cons with formal specification
4. Relationships between assumptions, specifications and system properties
5. Compare different specification techniques
6. Reviewing and validating a specification
7. The tendering process using specifications
8. Requirements elicitation
9. Formal VDM-SL specifications

Agenda

- ✓ Administrative information about the course
- Specifications and different kinds of systems
 - Structure of a Software Requirement Specification
 - Different levels of requirements
 - Requirements Elicitation
 - The Tendering Process

What is a specification?

- Exact statement of the particular needs to be satisfied, or essential characteristics that a customer requires and which a vendor must deliver.
- Specifications are written usually in a manner that enables both parties (and/or an independent certifier) to measure the degree of conformance.
- *A software specification is a precise, **unambiguous** and complete statement of the requirements of a system (or program or process), written in such a way that it can be used to predict how the system will behave*

Where is a specification used?

Project type	Customer	Supplier
In-house Product development Time and materials	Other part of firm Marketing/Sales Company	IT dept IT dept External IT firm
COTS	Company	(Vendor)
Tender Contract development Sub-contracting	Company Company Company (IT dept)	External IT firm External IT firm External IT firm
Unknown	?	?

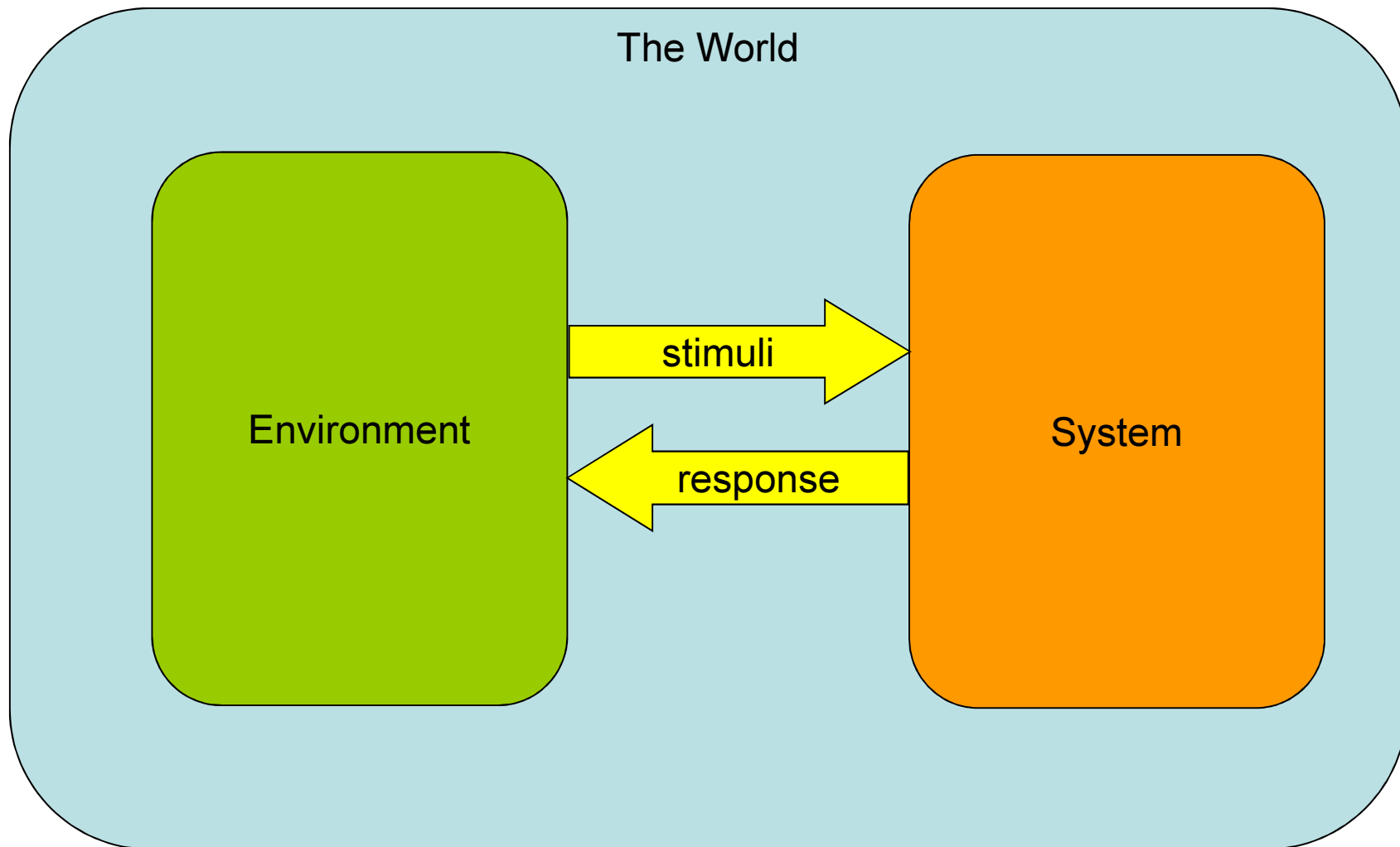
Reactive systems – first characterization

- Two main classes of systems
 - Transformational systems
 - Reactive systems
- Reactive systems respond to stimuli in order to bring about desirable effects in their environments
- Reactive systems may
 - Manipulate complex data
 - Engage in complex behavior
 - Communicate with (many) other systems

Reactive systems – definition (Wieringa)

- A reactive system is a system that, when switched on, is able to create desired effects in its environment by enabling, enforcing or preventing events in the environment.
- Characteristics
 - Interactive
 - Nonterminating
 - Interrupt-driven
 - State-dependent response
 - Environment-oriented response
 - Parallel processing
 - Real-time
- Examples
 - Information systems, workflow systems, groupware, web market places, production control software, embedded software

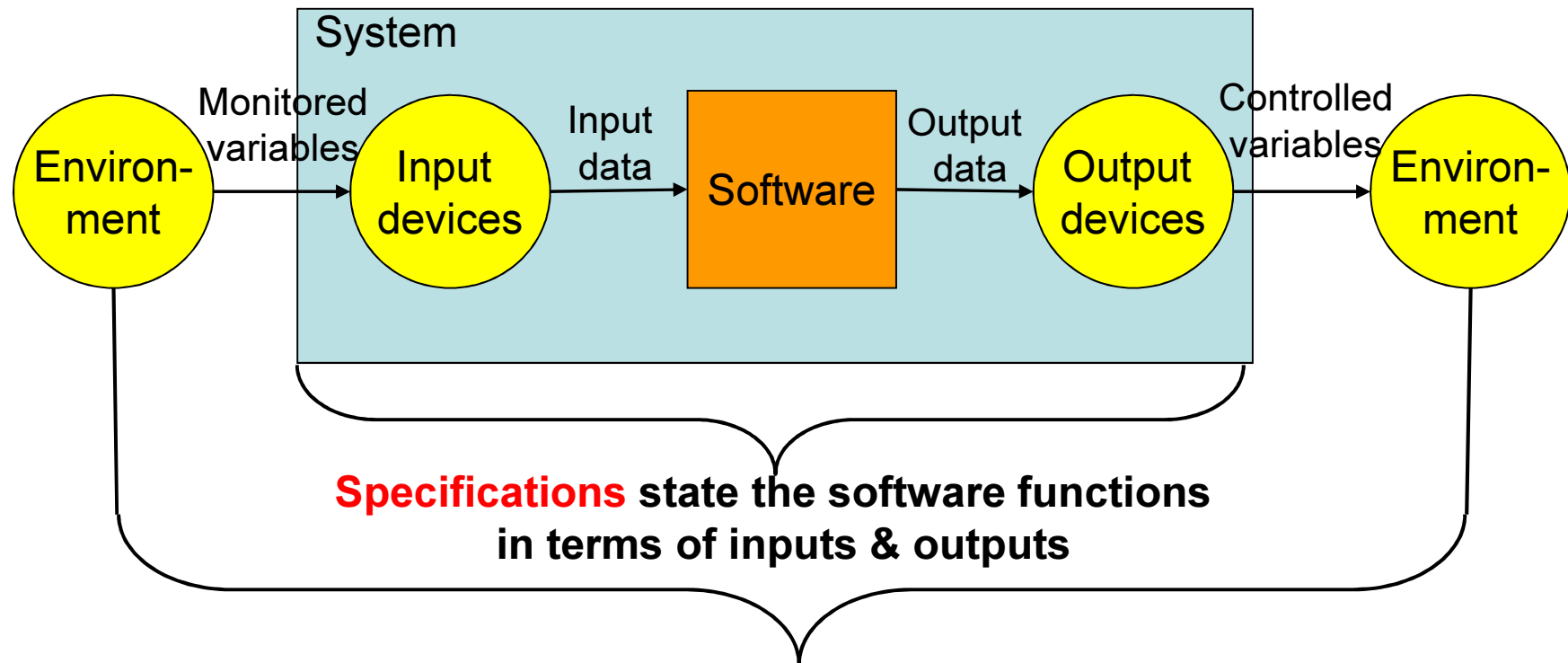
Reactive systems Nature



Transformational systems – definition (Wieringa)

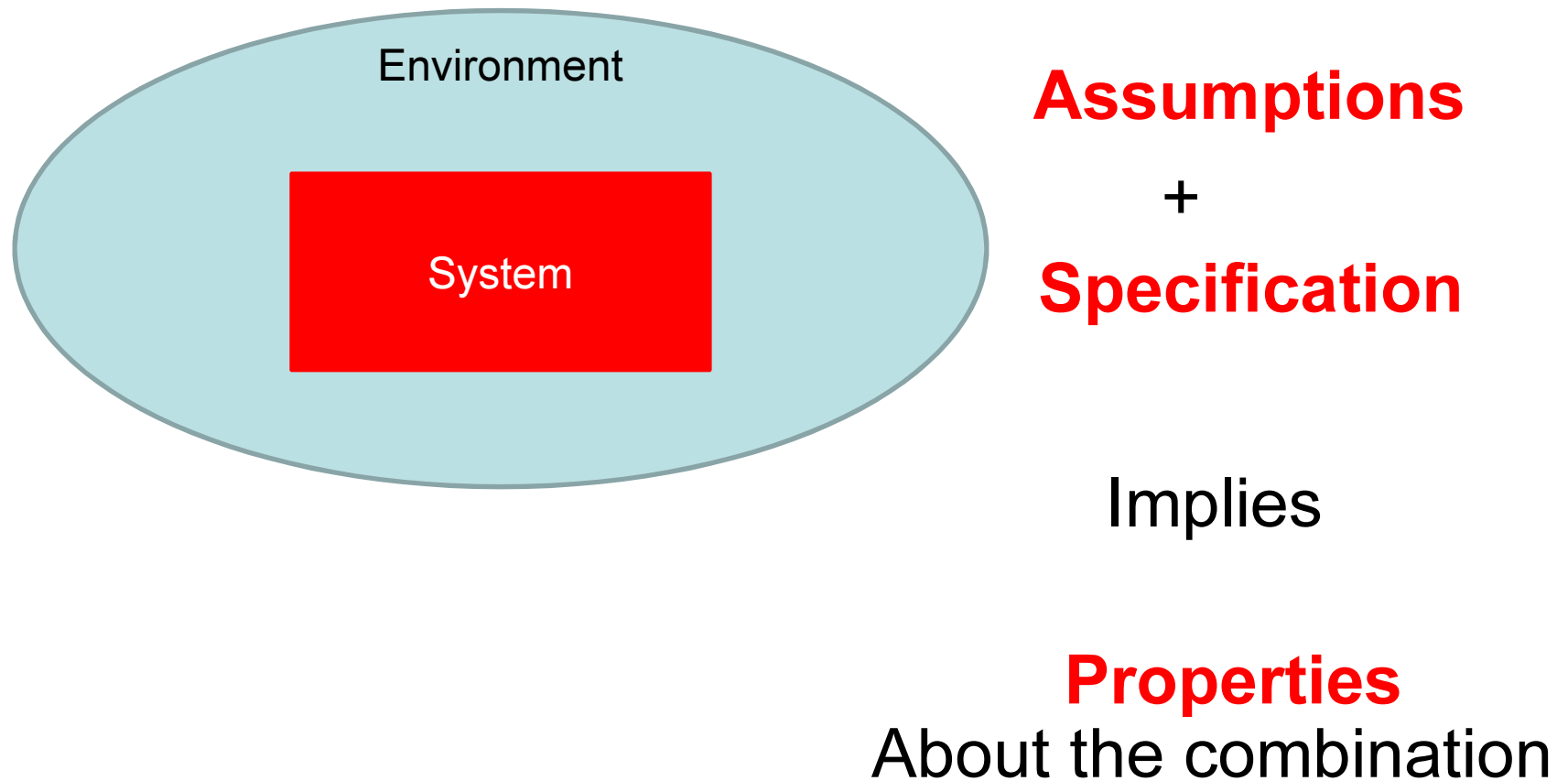
- Contrast reactive systems with transformational systems, that compute output from an input and then terminate
- Characteristics
 - Terminating
 - Sometimes interactive
 - Not interrupt-driven
 - Output not state-dependent
 - Output defined in terms of input
 - Sequential
 - Usually not real-time
- Examples
 - Compiler, web-shop, search algorithm, LaTeX, etc.

How Software Engineers see the world



Assumptions about expectations of the environment behaviour are important
Properties about the desired behaviour of the new system in the environment express a desired relationship between monitored and controlled variables

Assumptions, specifications and system properties



Engineering Arguments

- The ability to predict the properties of a product before it is built
- Feed-forward loop from specification to product properties
- **A**ssumptions and **S**pecification entail **P**roperties
- Properties are those desired for the System under Development:
 - System Requirements
 - System **Constraints**
- See Software for Dependable Systems: Sufficient Evidence? Daniel Jackson et al.


Example Engineering Arguments

- Engineering knowledge
 - Products with this kind of decomposition *usually* have properties P
 - Since this product will have this kind of decomposition
 - It will *probably* have properties P
- Throw-away prototyping
 - Since the prototype has properties P,
 - And the prototype is similar to the final product
 - The final product *probably* has properties P

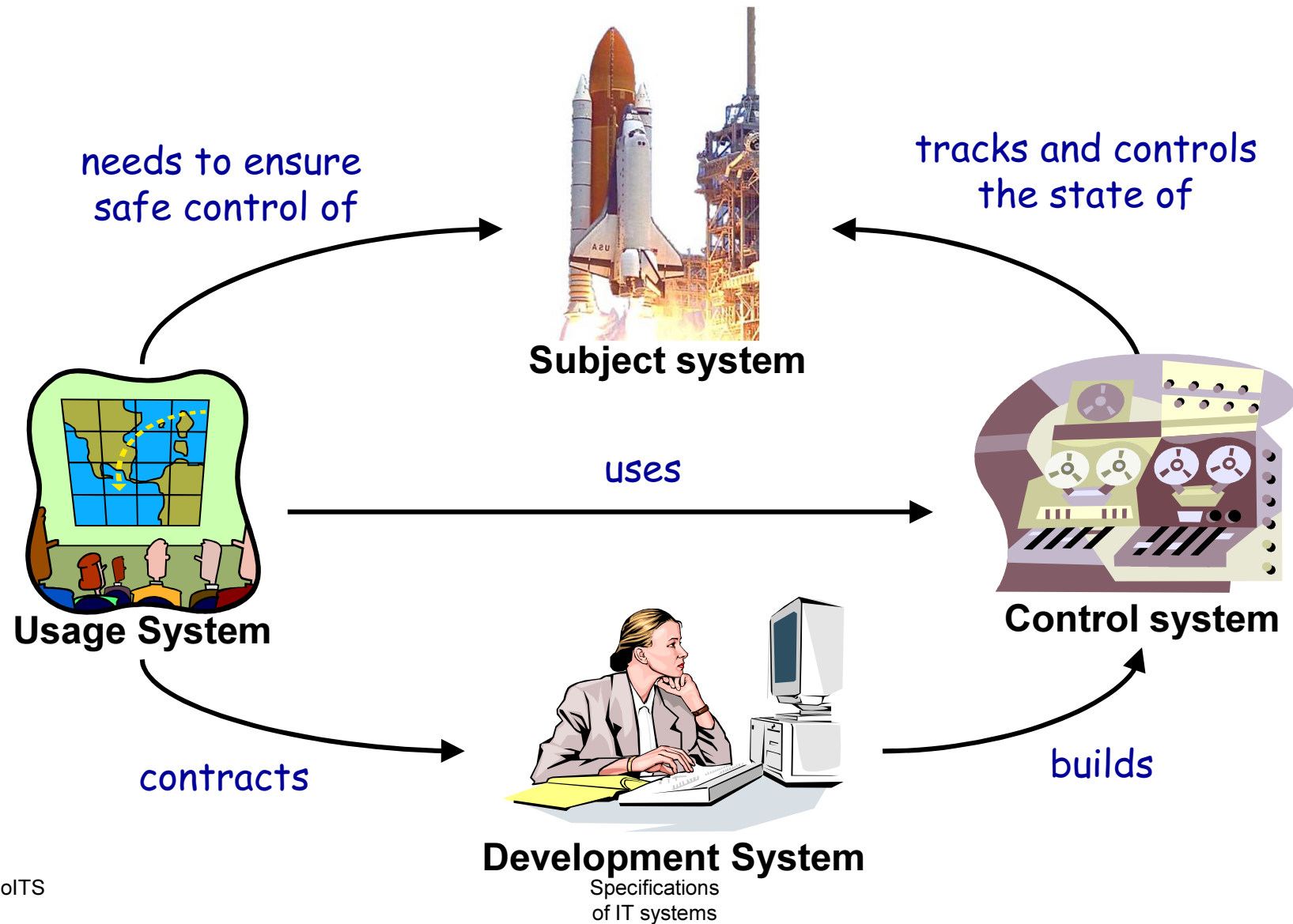
Example Engineering Arguments

- Model execution
 - Since the model execution has properties P
 - *If* the system implements this model exactly
 - Then the system will have properties P
- Model checking
 - Since the state transition graph has properties P ,
 - *If* the system implements this graph correctly
 - Then the system will have properties P
- Theorem proving
 - Since the decomposition has been proved to have properties P ,
 - *If* the system correctly implements this decomposition
 - Then the system will have properties P

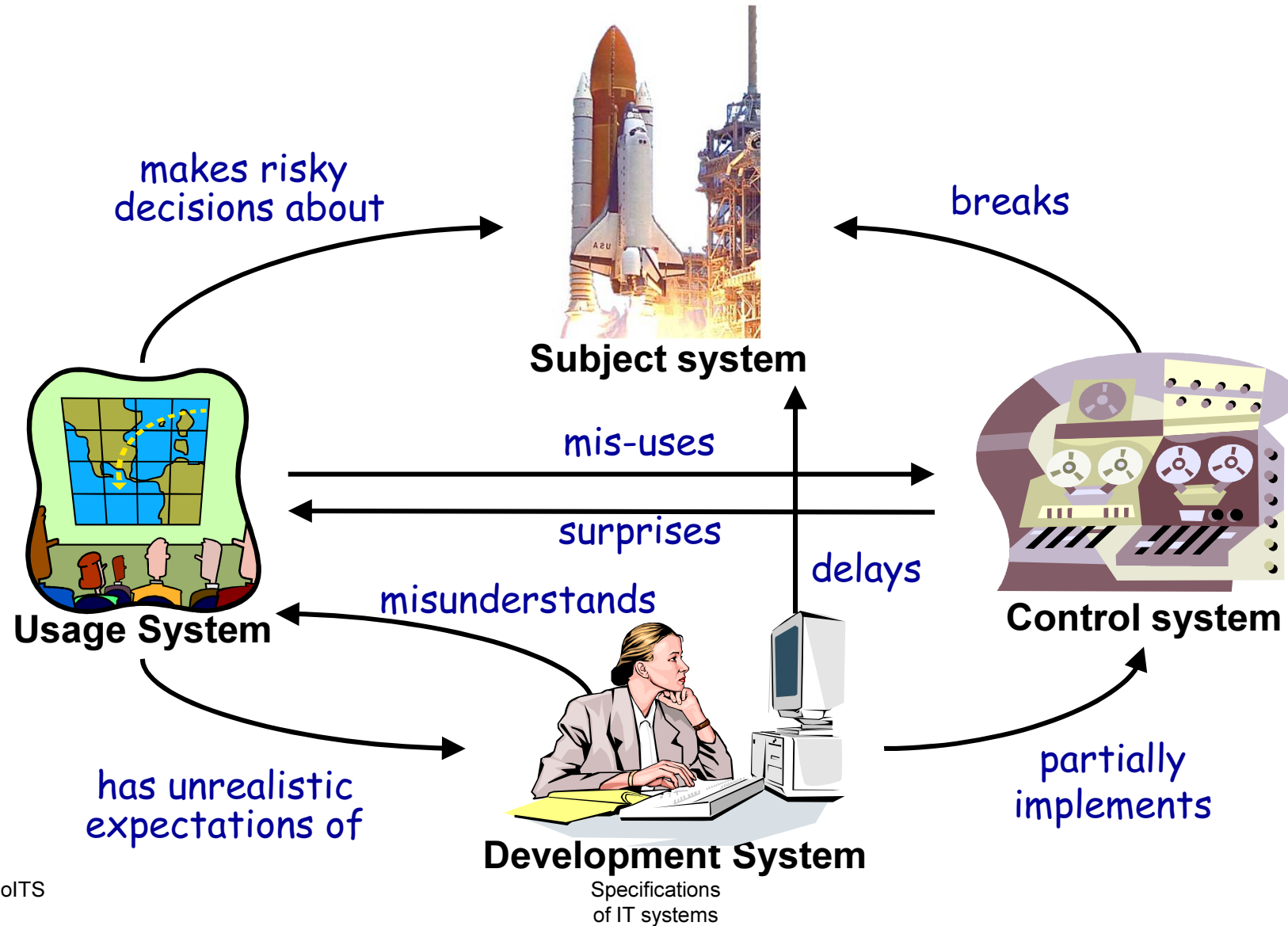
Assumptions about environment

- Assumptions are statements about the environment
 - Must be true for the (stimulus, response) pair to be desirable 
 - Beyond control of System under Development
- Examples of categories of assumptions
 - Laws of nature
 - Properties of devices
 - Properties of people (users, operators, ...)
- Some are explicit – and some are implicit

How Systems Engineers see the world



How the world really is...



Who uses requirements?

- What are the uses of a software requirement specification?
 - For customers it is a specification of the product that will be delivered, **a contract**
 - For managers it can be used as a basis **for scheduling and measuring progress**
 - For the software designers it provides a specification of **what to design**
 - For coders it defines the range of **acceptable implementations** and the **outputs that must be produced**
 - For quality assurance personnel it is used for **validation, test planning, and verification**

Agenda

- ✓ Administrative information about the course
- ✓ Specifications and different kinds of systems
- Structure of a Software Requirement Specification
 - Different levels of requirements
 - Requirements Elicitation
 - The Tendering Process

What does a specification consist of?

- Functional requirements
 - What the system should do
 - Requirement for each input and output
 - Often combined in reactive systems
- Behavioural requirements
 - Only describes the behaviour of the system
- Structural requirements
 - Describes constraints on the internal composition of the system

What does a specification consist of?

- Non-functional requirements
 - How the system is supposed to be
 - Often described as “Quality of Service” components
 - Performance
 - Usability
 - Security
 - Maintainability
 - Reliability
 - Efficiency
 - Etc.

An Example Outline for SRS

(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

- List software products that will be produced
- Summarize what software products will do
- Describe the application of the software being specified, including relevant benefits, objectives and goals

1.3 Definitions, acronyms, abbreviations

- Definition of all terms, acronyms, abbreviations required to properly interpret SRS

1.4 References

- Provide a complete list of referenced documents

1.5 Overview

- Describe what is in the reminder of the document
- Explain how SRS is organized

2. Overall description

2.1 Product perspective

- Identify the interface between the proposed software and existing systems, including a diagram of major system components.
- A block diagram showing major components of the larger system, interconnections, and external interfaces can be helpful.

2.2 Product functions

- Provide a summary of the major functions that the software will perform
- The functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document
- Diagrams can be used to explain different functions and their relationships

2.3 User characteristics

- General characteristics of the intended user of the product, level of expertise/training required to use the product

2.4 Constraints

- List all the constraints that will limit the developers options, interfaces to other applications, programming language requirements, hardware limitations, etc.

2.5 Assumptions and dependencies

- List the factors that affect the requirements in the SRS (assumptions on which operating system is available etc.)

3. Specific requirements (these are the detailed requirements)

- This section of the SRS should contain the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements

3.1 External interface requirements

- This section should specify various interfaces in detail: 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:
 - Should include: source of input and destination of output; valid range, accuracy and/or tolerance; units of measure; timing; screen formats/organization; window formats/organization; data formats; command formats, etc.
- Should complement but not repeat the information given in section 2

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.

3.2 Functional requirements

- Functional requirements should define all the fundamental actions that the system must take place in the software in accepting and processing the inputs and in processing and generating the outputs
- Should include: validity checks on input; exact sequence of operations; responses to abnormal situations; relationship of outputs to inputs
- It can be organized in various ways, such as with respect to user classes, features, stimulus or a combination of those.
- Use-case diagrams, scenarios, activity diagrams can be used here
- ***This section is very important. You need to organize use-cases etc. very well so that they are comprehensible. You need to make sure that your functional requirements are unambiguous, complete and consistent.***

3.3 Performance requirements

- Speed, availability, response time, recovery time of various software functions, etc.
- Performance requirements should be specified in measurable terms. For example: *“95% of the transactions shall be processed in less than 1 second.”* rather than *“An operator shall not have to wait for the transaction to complete”*
- There can be a separate section identifying the capacity constraints (for example amount of data that will be handled)

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

Agenda

- ✓ Administrative information about the course
- ✓ Specifications and different kinds of systems
- ✓ Structure of a Software Requirement Specification
- Different levels of requirements
 - Requirements Elicitation
 - The Tendering Process

A good requirement specification

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

What a good requiremen specification includes

- Complete trail of changes (traceability)
- Environment description
 - Environment assumptions
- Necessary design requirements
 - Bandwidth limitations
 - Memory constraints

What should be specified?

- Specify what the system should do – not how
 - Maintain freedom to choose solution in design phase
- Different levels 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
- 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
 - Describing in- and output of the system
 - Identifying functions and features of the system
- 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 specifying 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.”
- To precise descriptions limits the possible solutions
- If the descriptions are too vague, the customer may not get what he wants
- Very easy to test design-level requirements

Choosing the right level

- Depends on who needs to implement the system
 - Level of domain knowledge
- Often a mixture of several levels is the best solution
 - Business goals to describe overall goal
 - Domain-level requirements to describe interaction with environment or other existing system components
 - Product-level requirements to describe the functional requirements of the system

Agenda

- ✓ Administrative information about the course
- ✓ Specifications and different kinds of systems
- ✓ Structure of a Software Requirement Specification
- ✓ Different levels of requirements
- Requirements Elicitation
 - The Tendering Process

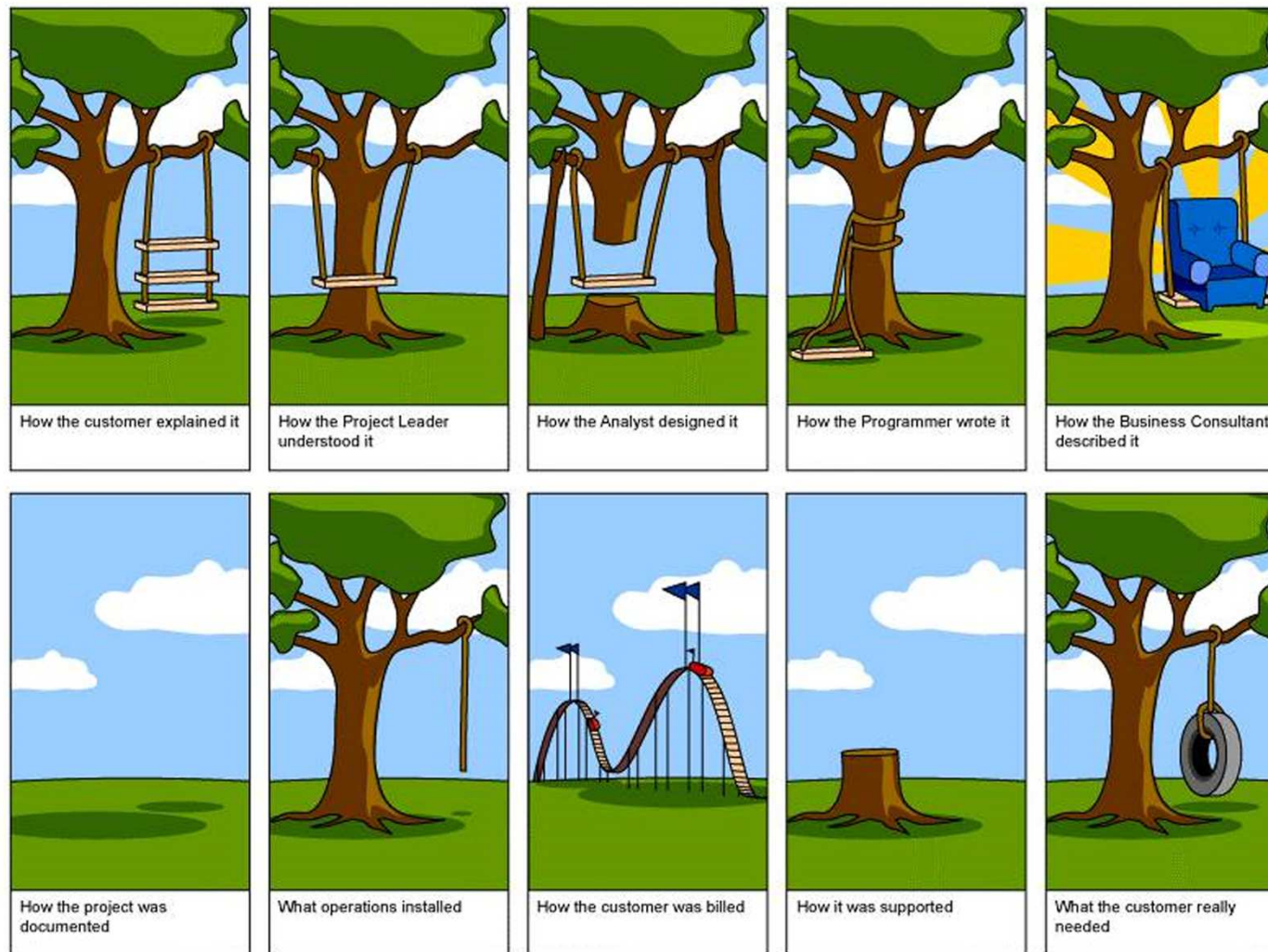
Elicitation: Finding and formulating requirements

- Includes a lot of steps:
 - Formulating overall goal of the system (**mission statement**)
 - **Describing** current work process and problems
 - Detailed description of issues the system must solve
 - Come up with **possible solutions**
 - Turn issues and possibilities into requirements
- No sequential process – but iterative

Elicitation – why is it hard?

- Stakeholders cannot express what they want
- Hard for users to explain their daily tasks
 - And why they do these tasks
- Stakeholders come up with solutions – not demands
- Hard to imagine new ways of doing tasks
 - And consequences for this
- Different stakeholders have conflicting views
- General resistance to change
- Too many “nice to have” requirements are specified
- Changes spawns new requirements

Elicitation – why is it hard?



Elicitation – good advice

- Setting goals
 - Decide how to analyze data once collected
- Relationship with participants
 - Clear and professional
 - Informed consent when appropriate
- Triangulation
 - Use more than one approach
- Pilot studies
 - Small trial of main study

Elicitation techniques (1)

- Stakeholder analysis
 - Who are the stakeholders?
 - What are their goals?
 - Which risks and costs do they see?
 - Made in large, small or 1-on-1 meetings
- User interviews
 - Current work process and problems are identified
 - Broad interview with many different users
 - **Use open questions** → more open discussion

Elicitation techniques (2)

- Observations
 - Hard to describe what is done – easier to observe work process
- Task demonstration
 - Task specific observation
 - Usability tools:
 - Think-out-loud
 - Measure time used
 - Measure errors made
 - Count number of keystrokes

Elicitation techniques (3)

- Questionnaires
 - Larger group of users
 - Hard to analyse results – cannot ask users additional questions in order to understand their answers
 - Easy to misunderstand answers
- Brainstorm
 - Mixed group of stakeholders
 - Open to all suggestions
 - Suggestions often spawn new ideas

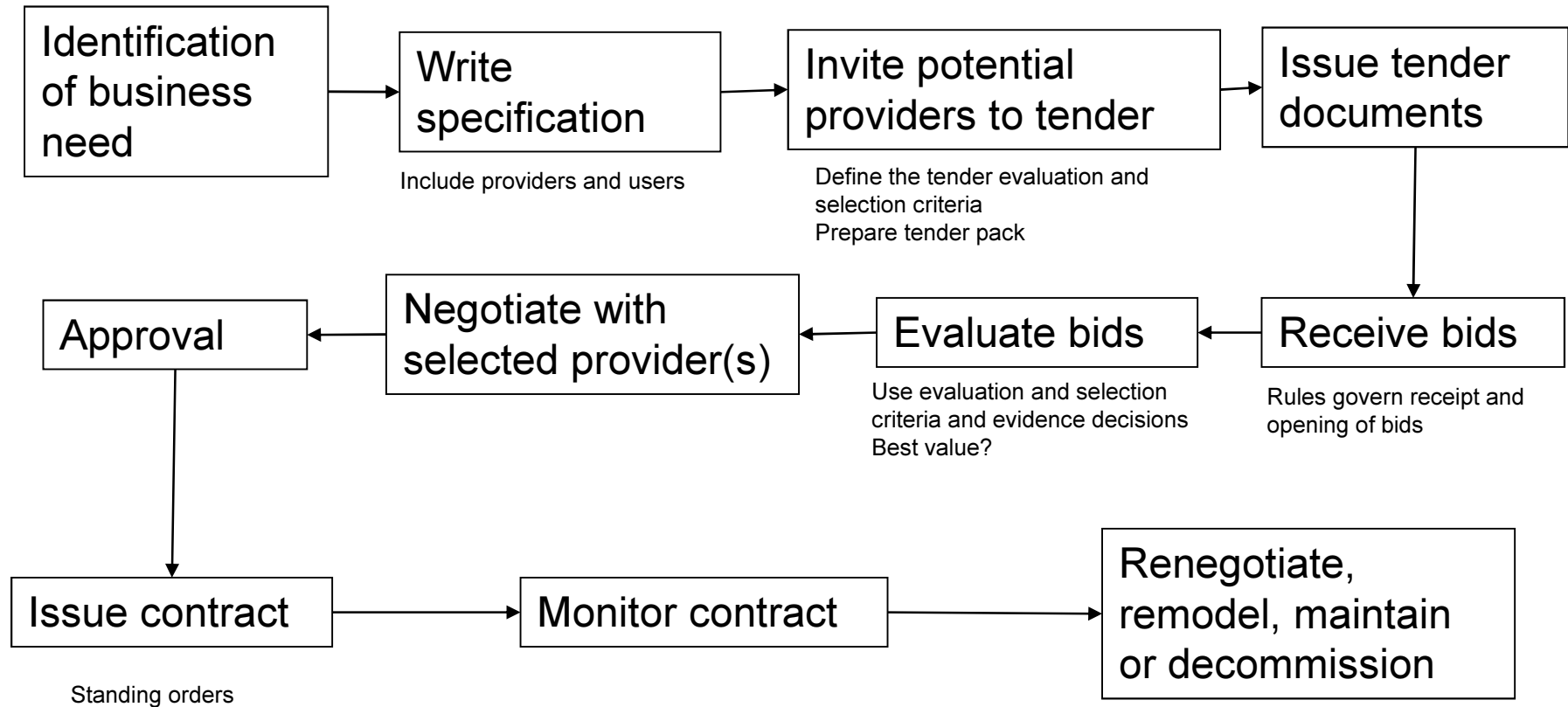
Elicitation techniques (4)

- **Focus group**
 - More structured than a brainstorm session
 - Focus on finding:
 - Problems and issues
 - Ideal solutions
 - Specify why the ideas are good
- **Domain Workshop**
 - Users and developers cooperate to analyse and design
 - Mixture of brainstorm and prototype sessions

Agenda

- ✓ Administrative information about the course
- ✓ Specifications and different kinds of systems
- ✓ Structure of a Software Requirement Specification
- ✓ Different levels of requirements
- ✓ Requirements Elicitation
- The Tendering Process

EU Tendering process flow



The traditional tender process

- Possibly pre-qualification
- Document for tender with list of requirement specification
- Possibly questions asked about requirements
- Potential suppliers create bid for tender
- Bids are evaluated by customer
- A supplier is selected
- The supplier needs to develop the promised solution
- The solution is delivered to customer
- The customer deploys the solution (possible with supplier)

Pre-qualification

The purpose of a pre-qualification process is to achieve all or some of the following:

- *To establish the interest, capability and suitability of contractors*
- *To identify companies for the bid list*
- *To maximise the % response to invitations to bid*
- *To demonstrate full and fair opportunity*
- *Information gathering*

Summary

- What have I presented today?
 - Administrative information about the course
 - Specifications and different kinds of systems
 - Structure of a Software Requirement Specification
 - Different levels of requirements
 - Requirements elicitation
 - The tendering process
- What do you need to do now?
 - Form groups
 - Read the IEEE SRS standard
 - Review and criticize the selected existing specification
 - Turn in hand-in 1 before Saturday 16:00 CET

Quote of the day

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



By Mark Ardis
Professor
Rochester Institute of Technology