Requirements Engineering Introduction

Gruia-Catalin Roman

edited and extended by Soraya Abad-Mota

January 2021

Department of Computer Science

University of New Mexico

Lectures available on Requirements Engineering

RS 1 Introduction

RS 2 Process

RS 3 Products

RS 4-6 Methods

Note: we are compressing them into 2, will post them all if you find them useful.

Software Crisis Revisited

Stil the 1.0

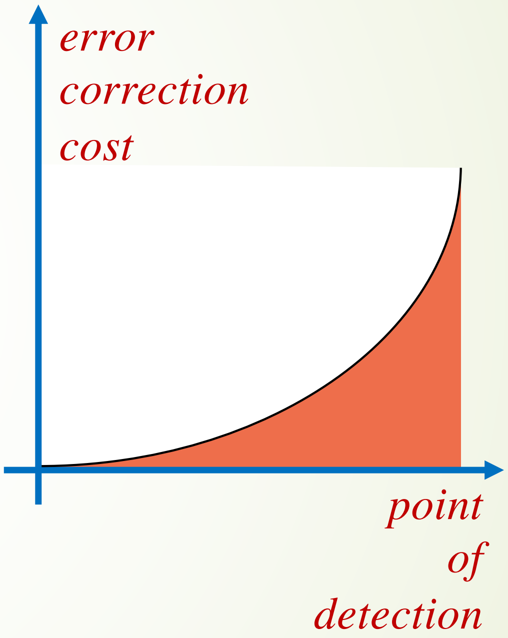
Causes (not only technical)

Summary of the main causes of failure in SW development:

- Complexity
- System requirements
- Weak management controls
- Lack of technical maturity

Costs

- The cost of correcting an error grows very rapidly as the project moves along its life-cycle
- This observation argues for early error detection and provides the motivation for technical reviews
- The highest cost errors are those involving the systems requirements formulation



Implications

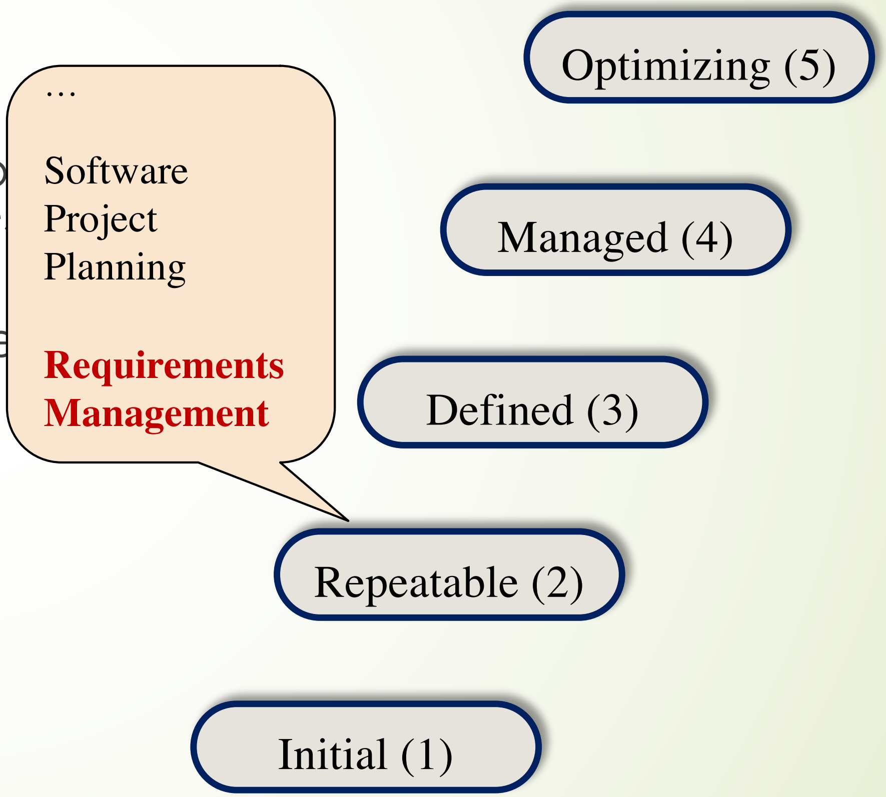
- Problems relating to the identification and documentation of system requirements present the highest risk for a project
- Investments in other areas of the software development process can be easily undercut by problems with the requirements
- Meaningful measurement and evaluation must take into consideration the relationship between error introduction and error detection points
  - effectiveness of the quality assurance
  - weaknesses in the development process

Response

- There is no silver bullet for the very difficult task of requirements definition and management
- The state of the art, however, is ahead of the state of the practice
- A standardized framework can be the conduit for bridging the gap
  - increased awareness
  - common terminology
  - assimilation of very basic practices

Capability Maturity Model (1993)

- Capability Maturity Model (CMM) is a framework designed to facilitate the introductio of basic sound practice across the industry
- CMM does not solve the technical problem
- CMM facilitates the adoption of sound technical and managerial practices
- Born to "help organizations improve their software process".



…

Software Project Planning

Requirements Management

Optimizing (5)

Managed (4)

Defined (3)

Repeatable (2)

Initial (1)

Repeatable Level

Key process areas

- Requirements management
- Software project management
- Software project tracking and oversight
- Software subcontract management
- Software quality assurance
- Software configuration management

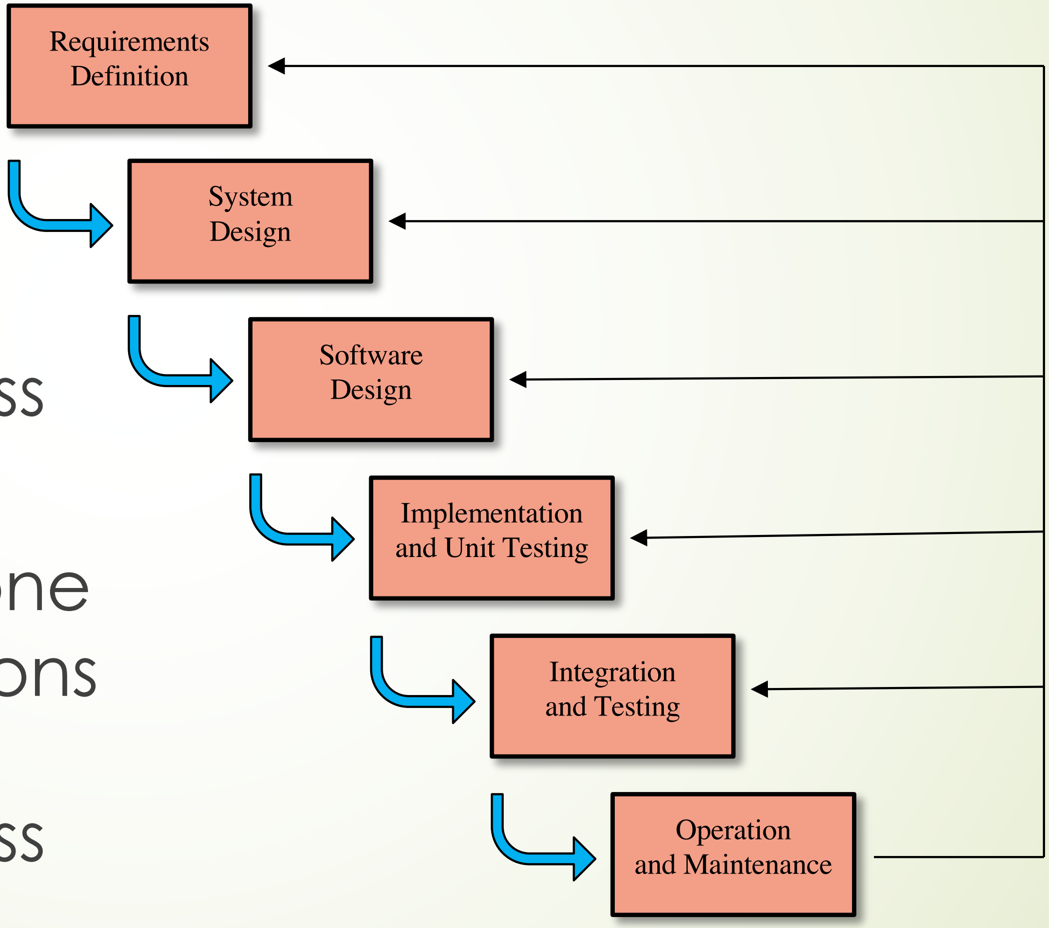
Life Cycle Perspective

Beyond Development

- Modern software engineering is based on the premise that design decisions and planning must consider the entire life of the product
- A narrow (development only) perspective is likely to lead to failures, lack of dependability and later expenses much greater than the development costs
- Maintainability, enhanceability, portability, etc. are fundamental life-cycle concerns
- The life cycle starts with the requirements definition

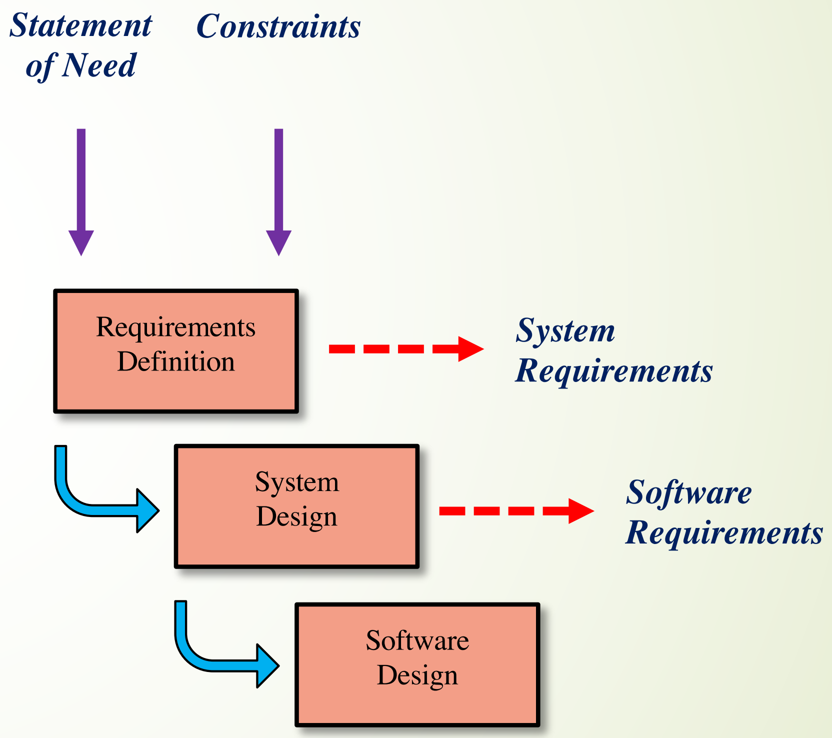
The phases

- Understanding requirements presupposes a good grasp of the development process as a whole
- This model remains one of the best abstractions for the software development process



Requirements in Context

- Requirements may vary in level of abstraction and contents from one context to another
- System requirements are the result of an analysis or discovery process
- Software requirements are the result of a design process involving requirements allocation
- Sometimes there is no distinction between them



Fundamental Concerns

- What are requirements?
- Why are they significant?
- When are they generated?
- How are they generated?
- How are they documented?
- How are they managed?
- When are they discarded?
- Can requirements be implicit?

Terminology

Requirement

- A requirement is a technical objective which is imposed upon the software, i.e., anything that might affect the kind of software that is produced
- A requirement may be imposed by
  - the customer
  - the developer
  - the operating environment
- The source, rationale, and nature of the requirement must be documented
- Requirements fall into two broad categories
  - functional
  - non-functional

Functional Requirements

- Functional requirements are concerned with what the software must do
  - capabilities, services, or operations (features)
- Functional requirements are not concerned with how the software does things, i.e., they must be free of design considerations
- Functional requirements are incomplete unless they capture all relevant aspects of the software's environment
  - they define the interactions between the software and the environment
  - the environment may consist of users, other systems, support hardware, operating system, etc.
  - the system/environment boundary must be defined

Non-Functional Requirements

- Place restrictions on the range of acceptable solutions
- Cover a broad range of issues
  - interface constraints
  - performance constraints
  - operating constraints
  - life-cycle constraints
  - economic constraints
  - political constraints
  - manufacturing

Important Messages

- Constraints are the main source of design difficulties
- No formal foundation on which to base the treatment of most non-functional requirements is available today
- Non-functional requirements are at least as dynamic as the functional ones

Significance and Impact

- Requirements are the foundation for the software development process
- Requirements impact the life cycle throughout its phases
  - customer/developer interactions
  - contractual agreements
  - feasibility studies
  - quality assurance
  - project planning
  - risk analysis
  - testing
  - user documentation

Requirements Engineering Process

Goals

- Fundamental goals of the requirements definition phase
  - to understand the nature of the problem
  - to establish a baseline for the software development process
  - to facilitate communication among participants in the development effort

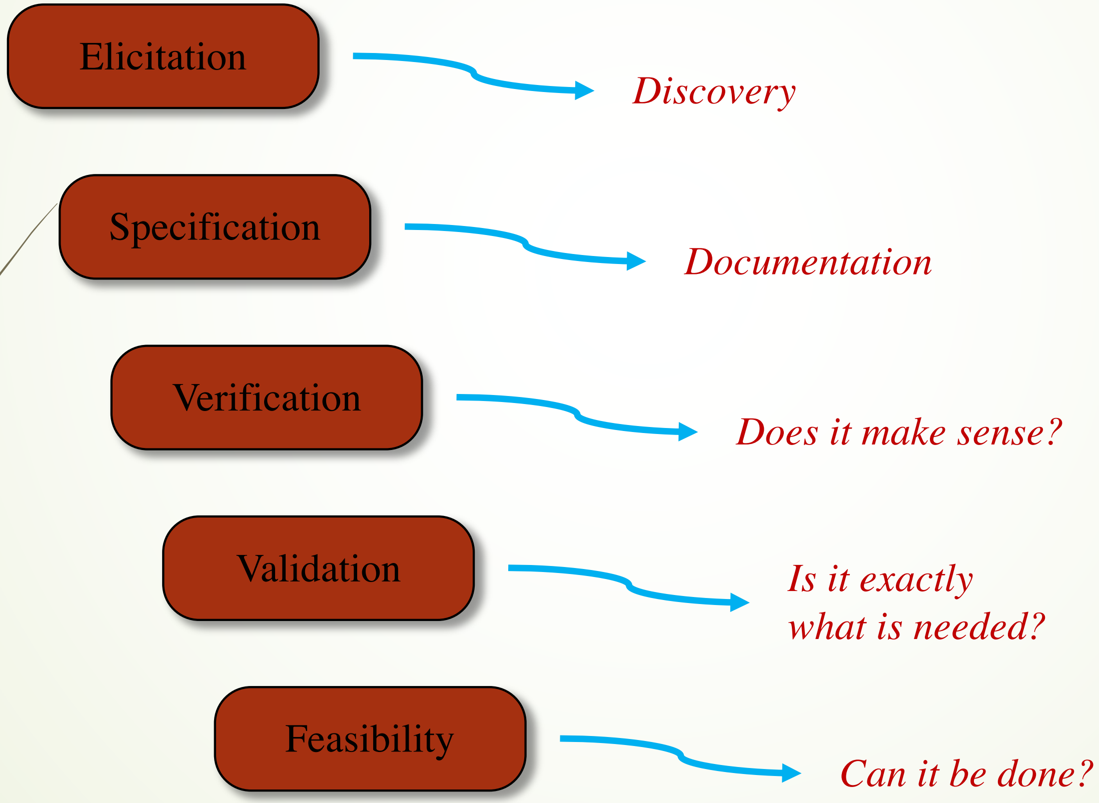
Observations

- Problem understanding is a prerequisite to starting any software development
- The establishment of a baseline involves
  - formal recording of the requirements (documentation)
  - analyzing them (feasibility)
  - accepting them as the basis for planning and development

Observations (cont.)

- Requirements definition is a communication-intensive phase whose goal is not only to extract information but to lay out a firm foundation for communication
  - between customers and developers
  - among various groups of developers

Activities



Elicitation → *Discovery*

Specification → *Documentation*

Verification → *Does it make sense?*

Validation → *Is it exactly what is needed?*

Feasibility → *Can it be done?*

Case Study: Thermostat

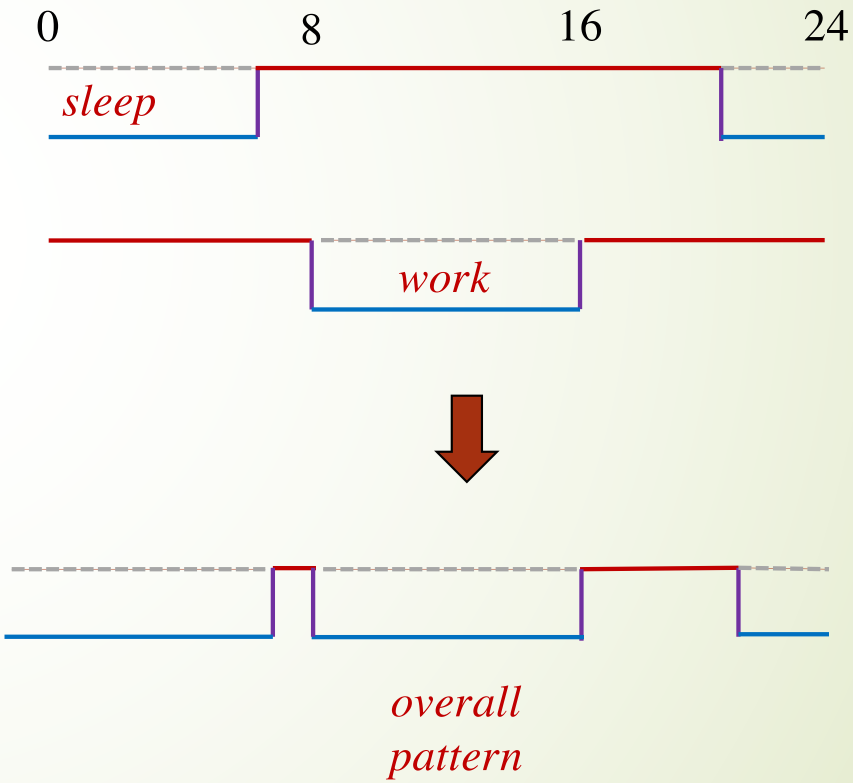
Elicitation

- Develop a thermostat controller for a heating system.
- Provide an energy saver feature designed to reduce the temperature setting by a fixed amount while the residents are at work and during the night.

Case Study: Thermostat

Specification

- Use a 24 hour profile diagram to capture the desired meaning for the control logic.
- View the falling and rising edges as events (offset on and off)

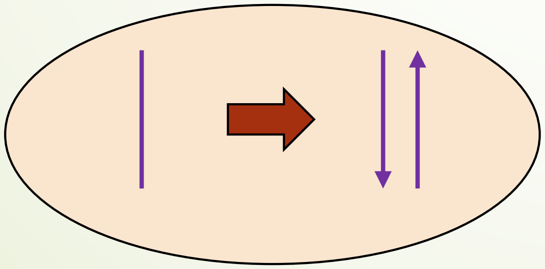


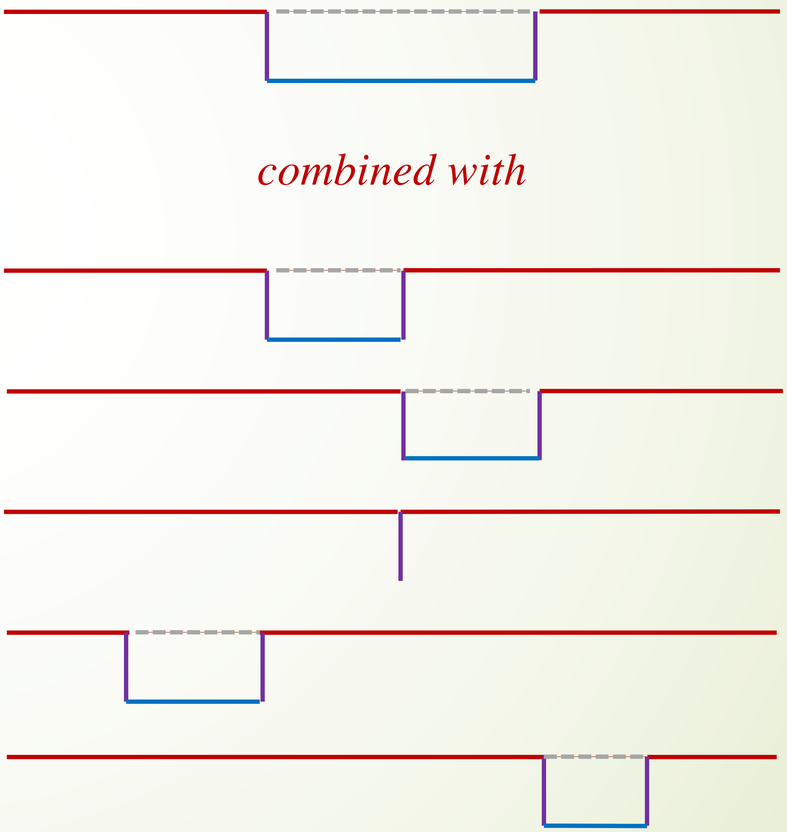
Case Study: Thermostat

Verification

- Evaluate the diagram interpretation against special cases where points on the diagram overlap.

Execute down events before up events





Case Study: Thermostat

Validation

- Evaluate against standard behavior patterns. Consider vacations (24 hour offset), weekends (override), etc.

Feasibility

- Check that all sensor and actuator controls are actually available.

Elicitation

Elicitation

- Discover and catalogue application needs
- Identify constraints
- Identify and prioritize objectives
- Reconcile conflicting views
- Define standard terminology
- Separate concerns
- Organize the information
- Pave the way to conceptualization
- Make technical specifications feasible

Issues

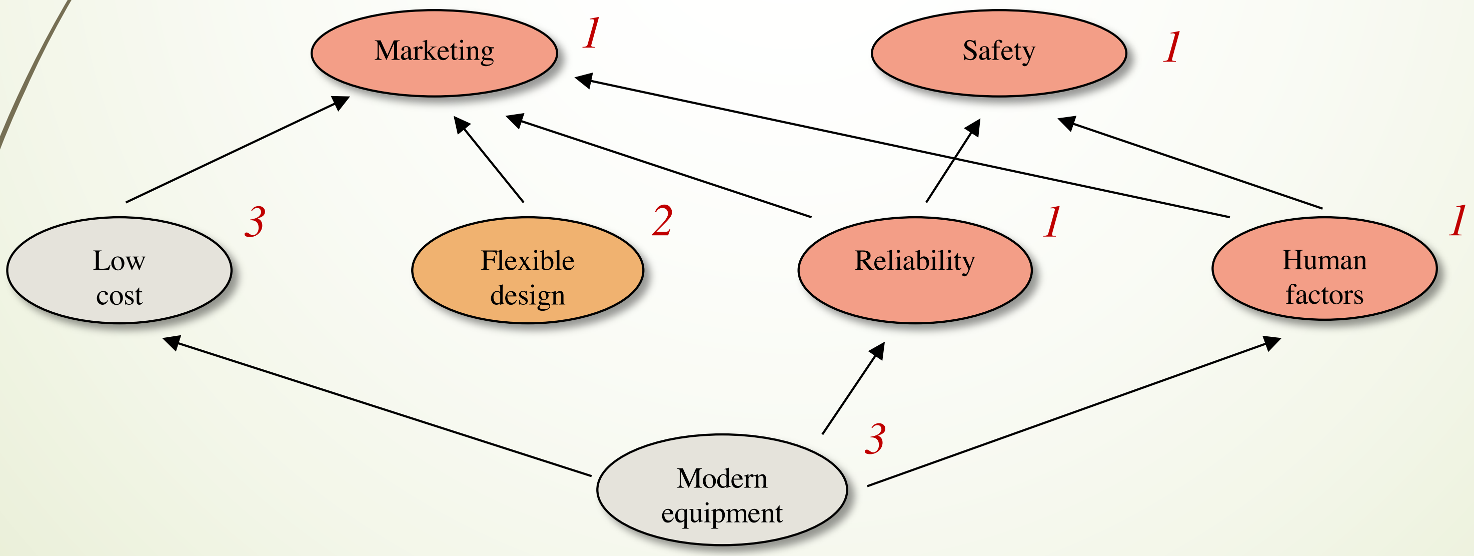
- Multiplicity of sources
- Conflicting interests
- Hidden objectives
- Unclear priorities
- Limited understanding of technology
- Communication difficulties
- Limited understanding of the application

Mechanics

- Systematic techniques can overcome the apparently ad-hoc nature of the process
- A simple five-step method
  - collect information
  - formulate working hypotheses
  - define terms
  - validate hypotheses and terms
  - separate concerns

Clarifying Objectives

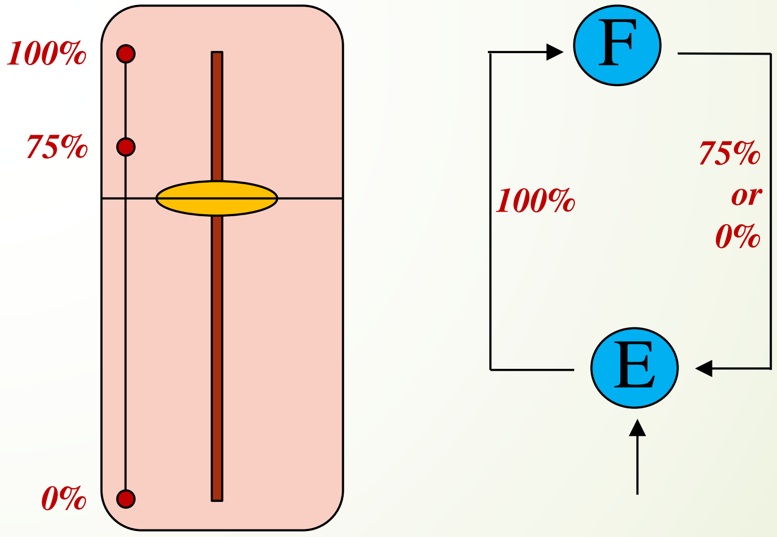
- Systematic acquisition of information must be accompanied by deeper understanding
- The relation among competing objectives is critical in carrying out technical trade-offs
- Illustration: *Rail traffic control system*



Seeking Simplicity

- The development of simple conceptual models helps clarify basic functional relationships
- Models also prepare the transition for the specification

- Illustration: *A tank refilling procedure*



Principal Product

- Requirements Definition Document (RDD)
  - is relatively high level
  - does not provide yet a baseline for the development (due to incompleteness)
  - does provide the basis for specification
  - is the starting point for a number of specialized preliminary studies
- The document must be accessible to a broad range of readers
  - customers, users, managers, designers

By-products

- Feasibility study
- Cost analysis
- Risk Analysis
- Market analysis
- Planning
- Component selection and evaluation
- Technology evaluation
- Human factors studies

Extra slides

- do not use

Implicit Requirements

- An interface specification can become a requirement definition only if
  - it is the only processing obligation
  - its semantics are well defined
- A product cannot be its own requirements definition because
  - the rationale for the design decisions is lost
  - there is no verification criterion

Non-Functional Requirements

- Non-functional requirements place restrictions on the range of acceptable solutions
- Non-functional requirements cover a broad range of issues
  - interface constraints
  - performance constraints
  - operating constraints
  - life-cycle constraints
  - economic constraints
  - political constraints
  - manufacturing

Important Messages

- Constraints are the main source of design difficulties
- No formal foundation on which to base the treatment of most non-functional requirements is available today
- Non-functional requirements are at least as dynamic as the functional ones

Significance and Impact

- Requirements are the foundation for the software development process
- Requirements impact the life cycle throughout its phases
  - customer/developer interactions
  - contractual agreements
  - feasibility studies
  - quality assurance
  - project planning
  - risk analysis
  - testing
  - user documentation

Multiple Perspectives
(additional ones)

- Waterfall model
  - product focused
- Evolutionary
  - increment driven
  - rapid prototyping
  - agile
- Spiral
  - risk analysis driven
- Transformational
  - specification driven