Requirements Engineering

Lecture 01:

Overview of Requirements Engineering (RE)

Overview of Requirements Engineering

- What is Requirements Engineering (RE)?
- Why Requirements Engineering?
- Model-based Requirement Engineering

The Problem World and the Machine Solution

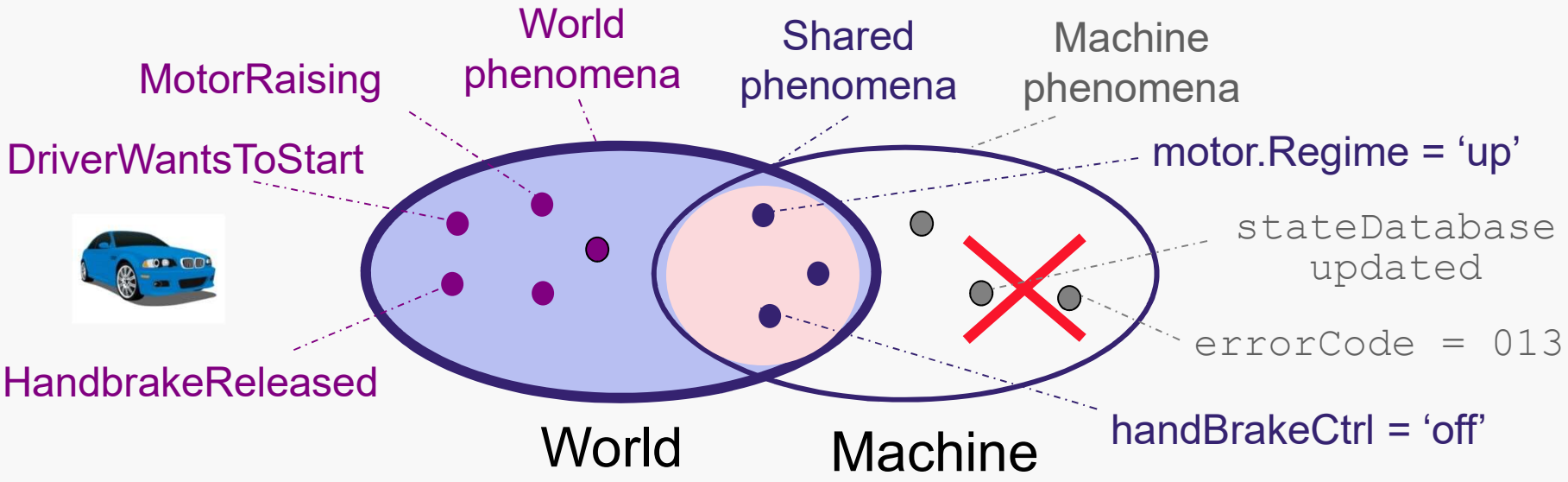
- To make sure a software solution "correctly" solves some real-world problem, we must first fully understand and define ...
  - what problem needs to be solved in the real world
  - the context in which the problem arises
- Example: car control
  - Problem: manual handbrake release can be inconvenient in certain situations
  - Context: car driving, braking, driver 's intent, safety rules, ...

The Problem World and the Machine Solution

- World: problematic part of the real-world, made of
  - human components: organization units, staff, operators, ...
  - physical components: devices, legacy software, mother Nature, ...
- Machine: what needs to be installed to solve the problem
  - software to be developed and/or purchased
  - hardware/software implementation platform, associated input/output devices (e.g. sensors & actuators)
- Requirements engineering (RE) is concerned with ...
  - the desired machine's effect on the problem world
  - the assumptions and relevant properties about this world

The Problem World
and the Machine Solution

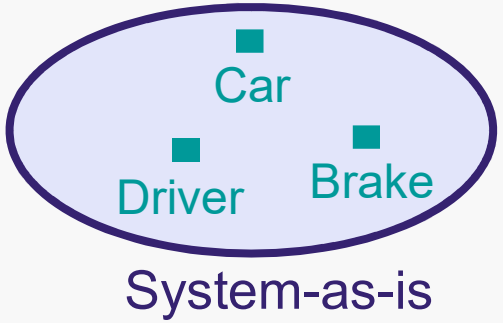
- The world and the machine have their own phenomena while sharing others
- RE is solely concerned with world phenomena, including shared ones [Jackson95]
  - unlike software design, concerned with machine phenomena



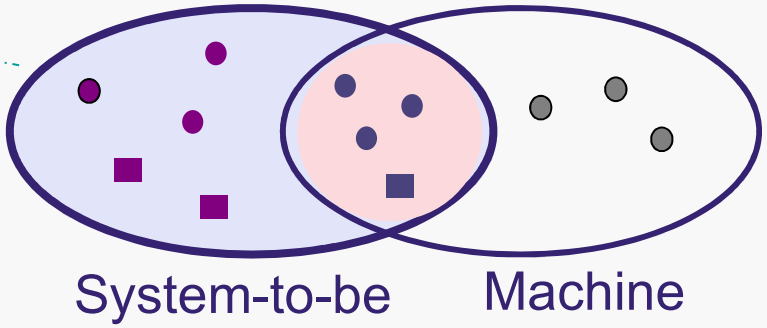
The problem world involves two system versions:

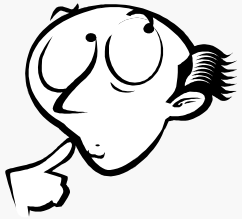
- System: set of interacting components structuring the problem world
- System-as-is: system as it exists before the machine is built into it
- System-to-be: system as it should be when the machine will operate into it

Concepts, phenomena, rules about car handbraking



Concepts, phenomena, rules about automated handbraking





RE: A Preliminary Definition

Coordinated set of activities ...

- for exploring, evaluating, documenting, consolidating, revising and adapting the objectives, capabilities, qualities, constraints & assumptions on a software-intensive system
- based on problems raised by the system-as-is and opportunities provided by new technologies

What others have said ...

Ross'77

- Requirements definition must say ...
  - why a new system is needed, based on current or foreseen conditions,
  - what system features will satisfy this context,
  - how the system is to be constructed

Zave'97

- RE is concerned with the real-world goals for, functions of, constraints on software systems; and with their
  - link to precise specifications of software behavior,
  - evolution over time and families

What is Requirements Engineering ?

- Identify & analyze problems with an existing system (system-as-is)
- Identify & evaluate objectives, opportunities, options for new system (system-to-be)
- Identify & define functionalities of, constraints on, responsibilities in system-to-be,
- Specify & organize all of these in a requirements document to be maintained throughout system evolution

System = software + environment (people, devices, other software)

System Requirements vs. Software Requirements

- Software-to-be: software to be developed - part of the machine, component of the system-to-be
- Environment: all other components of the system-to-be, including people, devices, pre-existing software, etc.
- System requirements: what the system-to-be should meet; formulated in terms of phenomena in the environment

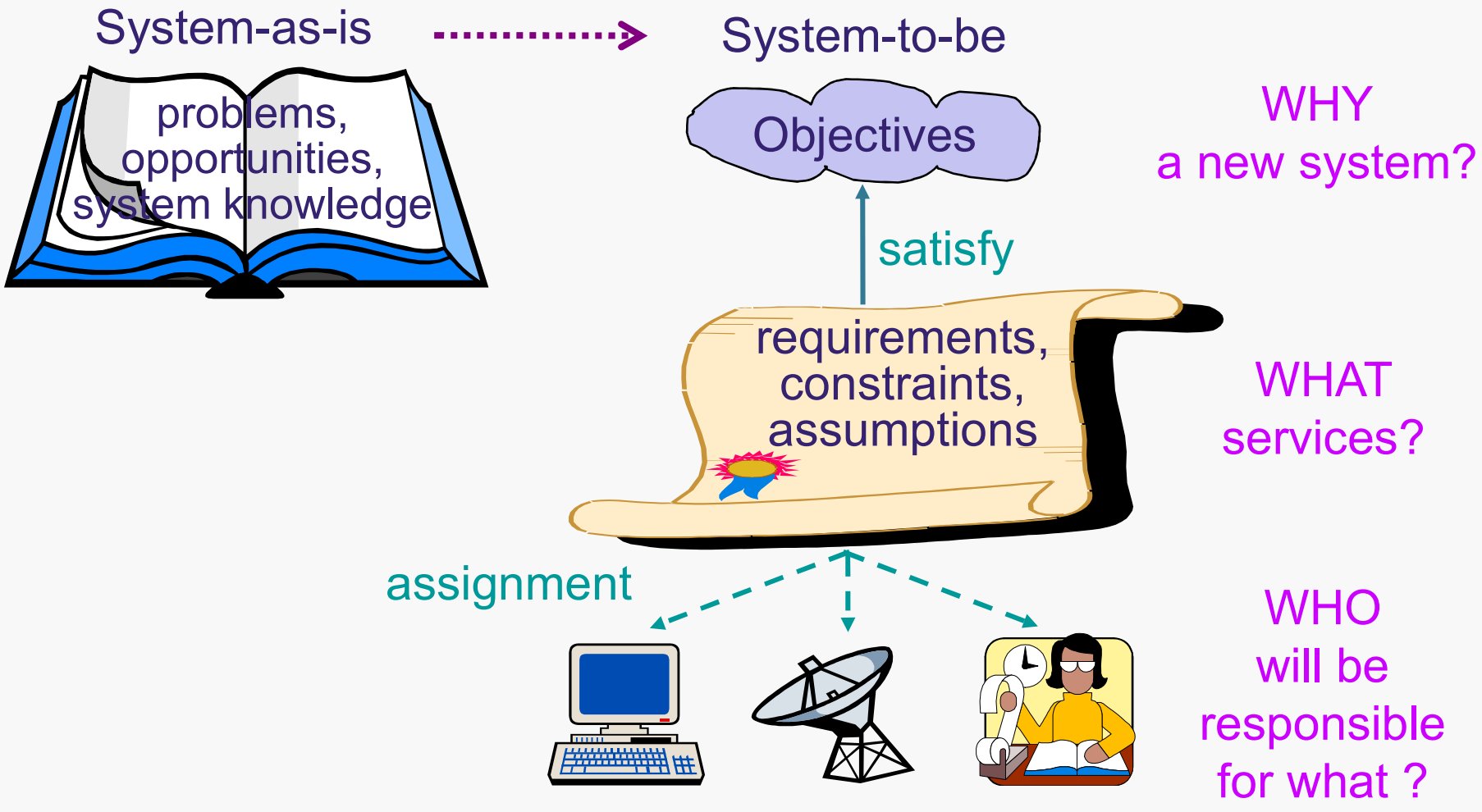
  "The handbrake shall be released when the driver wants to start."
- Software requirements: what the software-to-be should meet on its own; formulated in terms of phenomena shared by the software and the environment

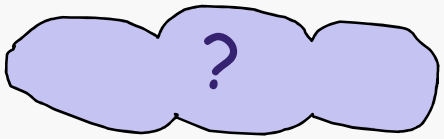
  "The software output variable handBrakeCtrl shall have the value off when the software input variable motorRegime gets the value up."

Example (Transportation between airport terminals)

- Problem (system-as-is):
  - passengers frequently missing flight connections among different terminals; slow & inconvenient transportation
  - number of passengers regularly increasing
- Objectives, options (system-to-be):
  - support high-frequency trains between terminals
  - with or without train drivers ?
- Functionalities, constraints:
  - software-based control of train accelerations, doors opening etc. to achieve prompt and safe transportation
- Requirement Engineering deliverable: requirements document for system-to-be

The Scope of RE: the WHY, WHAT, WHO Dimensions



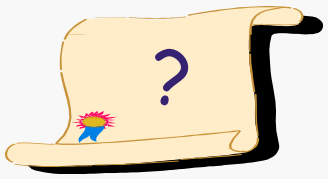


The WHY Dimension

- Identify, analyze, refine the system-to-be's objectives
  - to address analyzed deficiencies of the system-as-is
  - in alignment with business objectives
  - taking advantage of technology opportunities
- Example: airport train control

  "Serve more passengers"

  "Reduce transfer time among terminals"
- Difficulties
  - Acquire domain knowledge
  - Evaluate alternative options (e.g. alternative ways of satisfying the same objective)
  - Match problems-opportunities, and evaluate these: implications, associated risks
  - Handle conflicting objectives

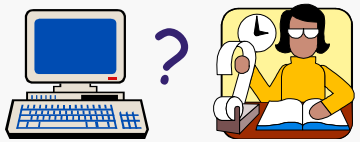


The WHAT Dimension

- Identify & define the system-to-be's functional services (software services, associated manual procedures)
  - to satisfy the identified objectives
  - according to quality constraints: security, performance, ...
  - based on realistic assumptions about the environment
- Example: airport train control

  "Computation of safe train accelerations"

  "Display of useful information for passengers inside trains"
- Difficulties
  - Identify the right set of features
  - Specify these precisely for understanding by all parties
  - Ensure backward traceability to system objectives



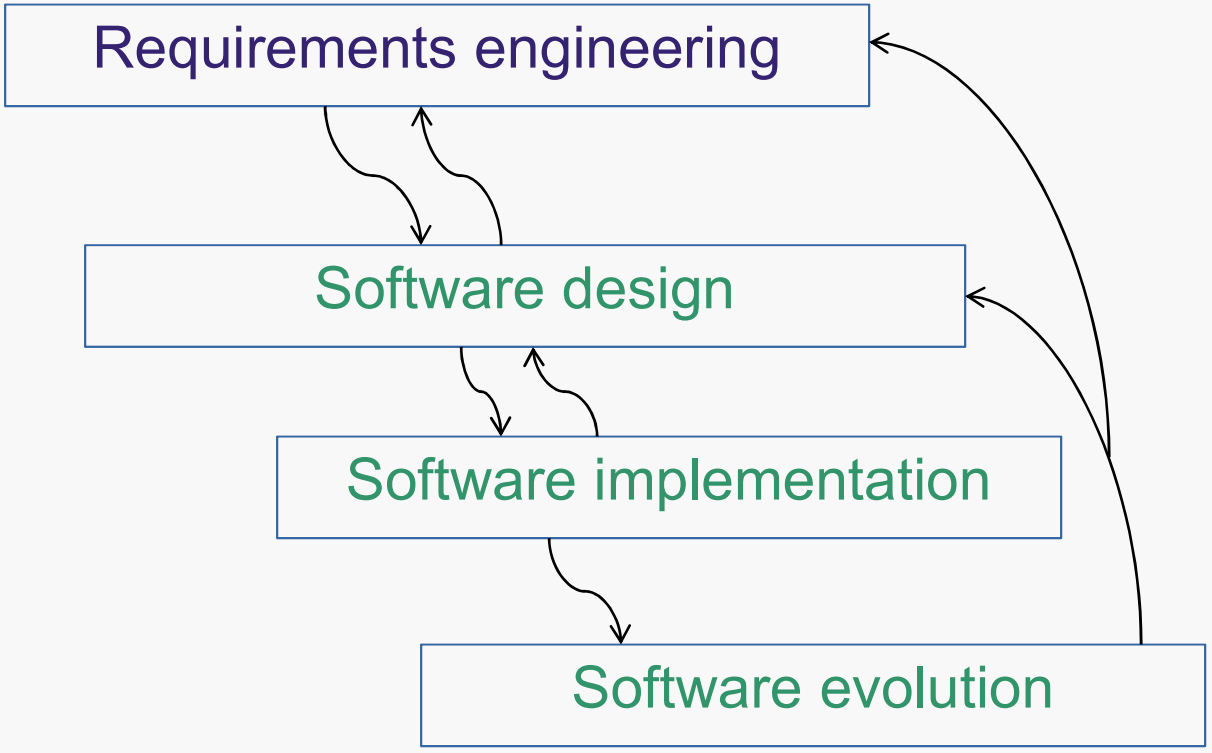
The WHO Dimension

- Assign responsibilities for the objectives, services, constraints among system-to-be components
  - based on their capabilities and on the system's objectives
  - yielding the software-environment boundary
- Example: airport train control
  - "Safe train acceleration" ... under direct responsibility of software-to-be (driverless option) or of driver following software indications ?
  - "Accurate estimation of train speed/position" ... under responsibility of tracking system or of preceding train ?
- Difficulties
  - Evaluate alternative options to decide on the right degree of automation

Requirements in the Software Lifecycle

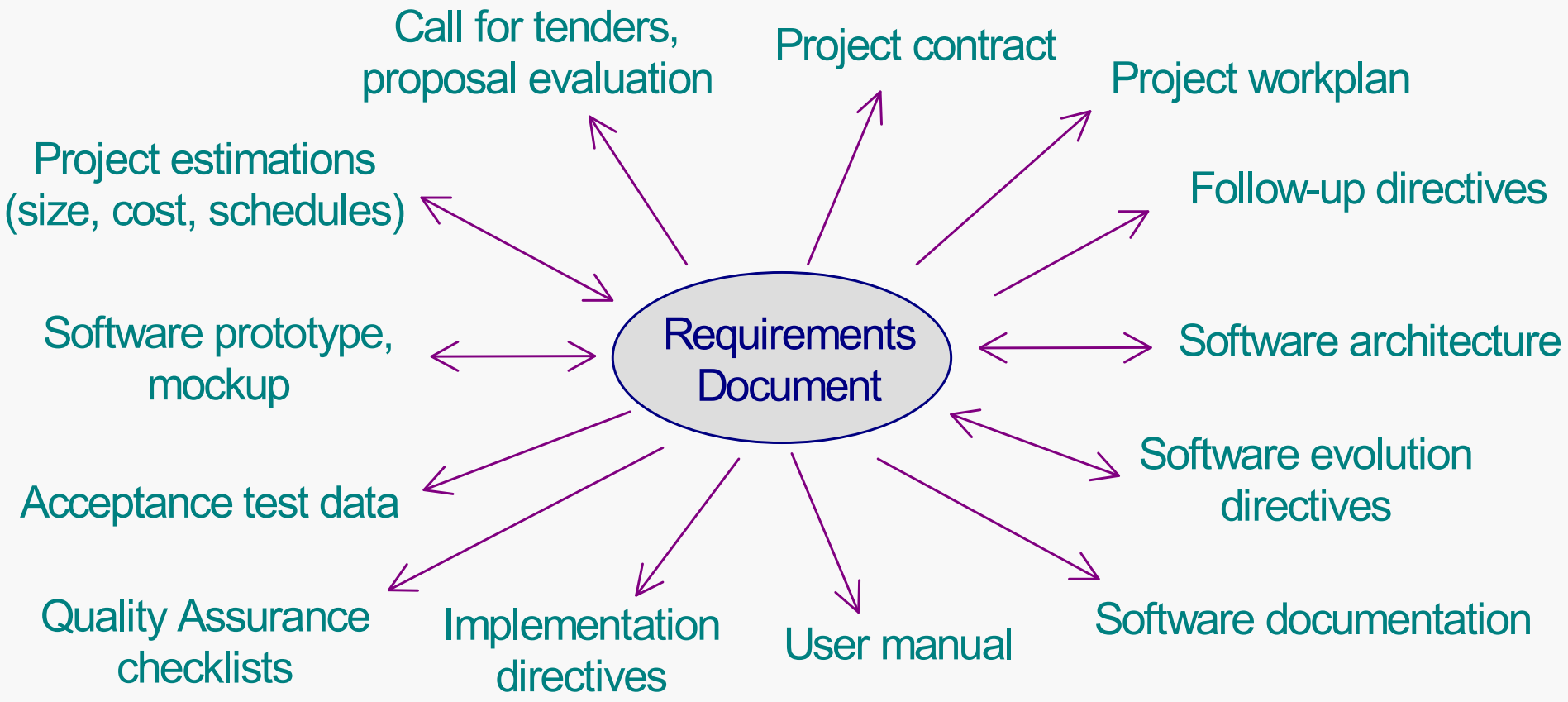
Getting the *right system*

Getting the *right software*



Requirements in the Software Lifecycle (2)

- The RD impacts on many software artifacts



- RE, system design & software architecture design are inevitably intertwined

Why Requirements Engineering?

- Requirement Engineering is critical
  - Major cause of software failure
    Requirements-related errors are the most numerous, persistent, expensive, dangerous
  - Severe consequences: cost overruns, delivery delays, dissatisfaction, degradations, accidents, ...
  - Requirement Engineering has multiple impact: legal, social, economical, technical
  - Certification issues
- Requirement Engineering is hard

What makes Requirements Engineering hard?

- Broad scope
  - multiple system versions: as-is, to-be, to-be-next
  - hybrid environment:
    - human organizations, policies, regulations
    - devices, physical laws
- Multiple concerns
  - functional, quality, development concerns
- Multiple abstraction levels
  - strategic objectives, operational details

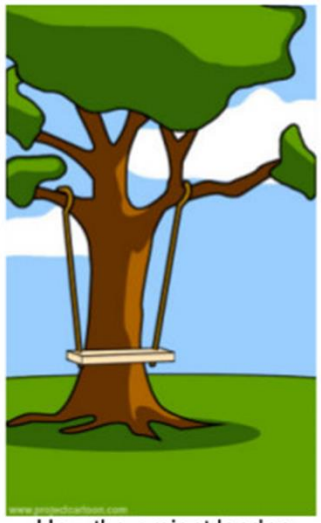
What makes Requirements Engineering hard? (2)

- Multiple stakeholders
  - with different background
  - with different interests and conflicting viewpoints
- Multiple intertwined tasks during iterative elicitation-evaluation-specification-consolidation
  - conflict management
  - risk management
  - evaluation of alternatives, prioritization
  - quality assurance
  - change anticipation

How Projects Really Work (version 1.0)

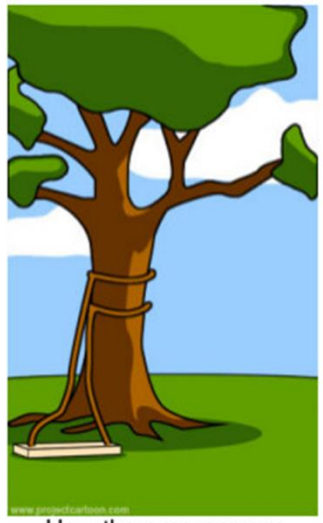
Create your own cartoon at www.projectcartoon.com

How the customer explained it



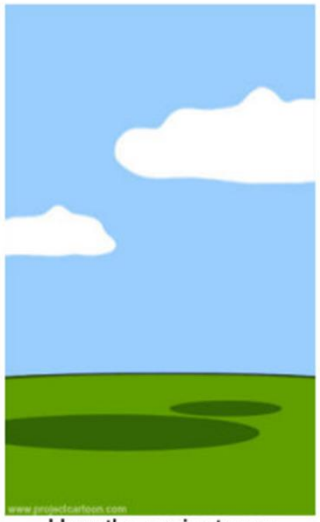
How the project leader understood it

How the analyst designed it

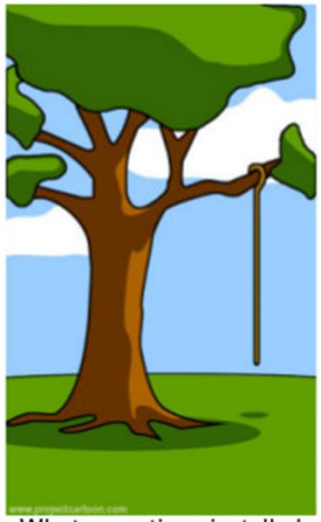


How the programmer wrote it

How the business consultant described it

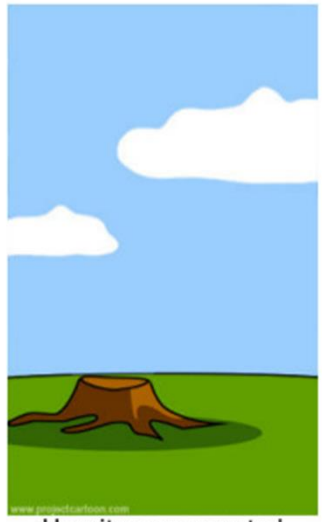


How the project was documented



What operations installed

How the customer was billed



How it was supported

What the customer really needed

Model-based Requirements Engineering

- Model
  - abstract representation of system (as-is or to-be)
  - highlights, specifies, inter-relates key system features

Why models for Requirements Engineering?

- Focus on key aspect (abstraction from multiple details)
- Provides structure for Requirement Engineering activities
  - target for what must be elicited, evaluated, specified,
  - consolidated, modified
  - interface among Requirement Engineering activities:
  - produce/consume model items
- Facilitates analysis
  - support for early detection and fix of errors
- Support for understanding, explanation to stakeholders
- Basis for making decisions
  - multiple options made explicit
- Basis for generating the requirements document