Software Management, Measurement and Quality Control

Establishing the Project Foundations

Instructor Dr. Morales

Concordia University Department
of Computer Science and Software
Engineering

Reading from Textbook: chapter 3 sections 3.4.5, 3.5

Introduction to project foundations

Foundations elements of software projects

Product foundations

- Operational requirements
- System requirements and system architecture
- Software requirements
- Design constraints

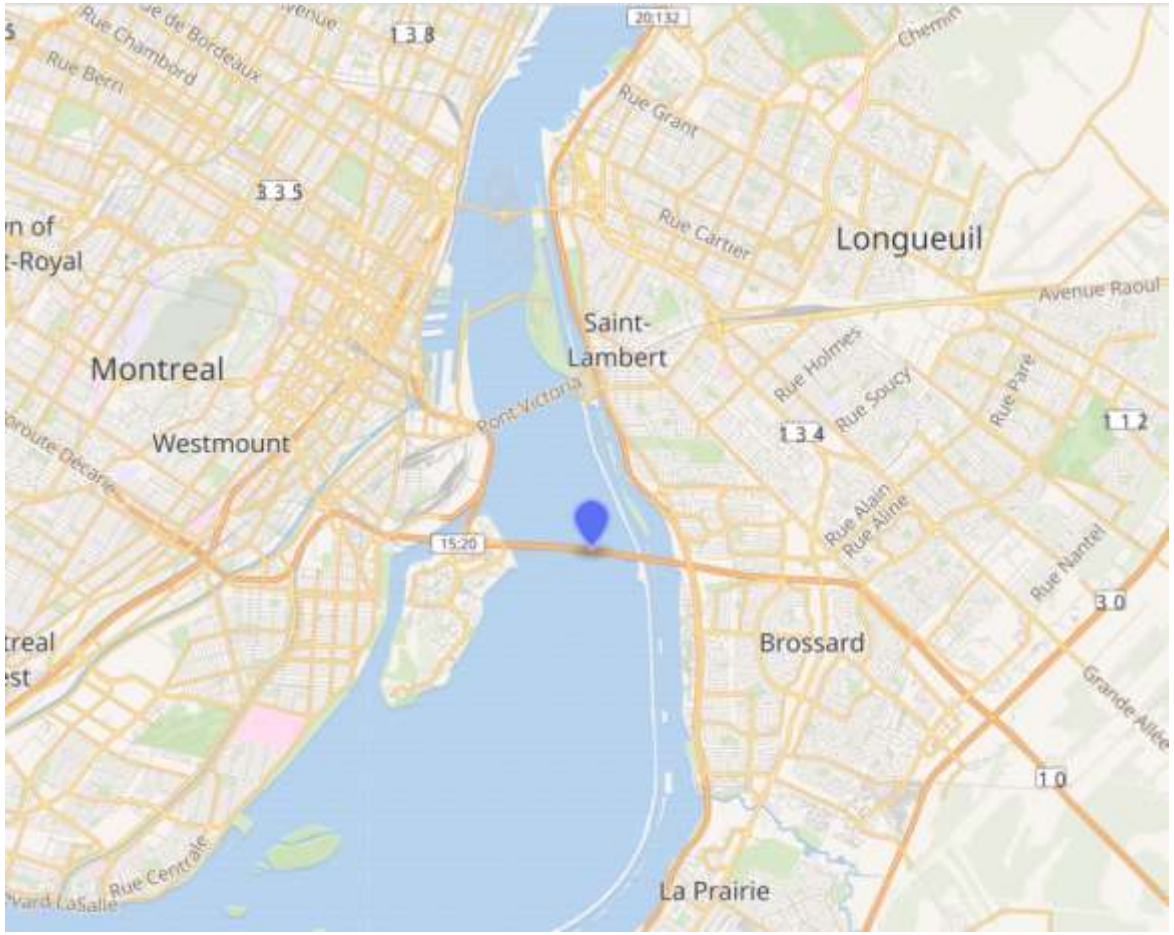
Process foundations

- Contractual agreement
- Workflow model
- Technical work activities and work products
- The project roadmap

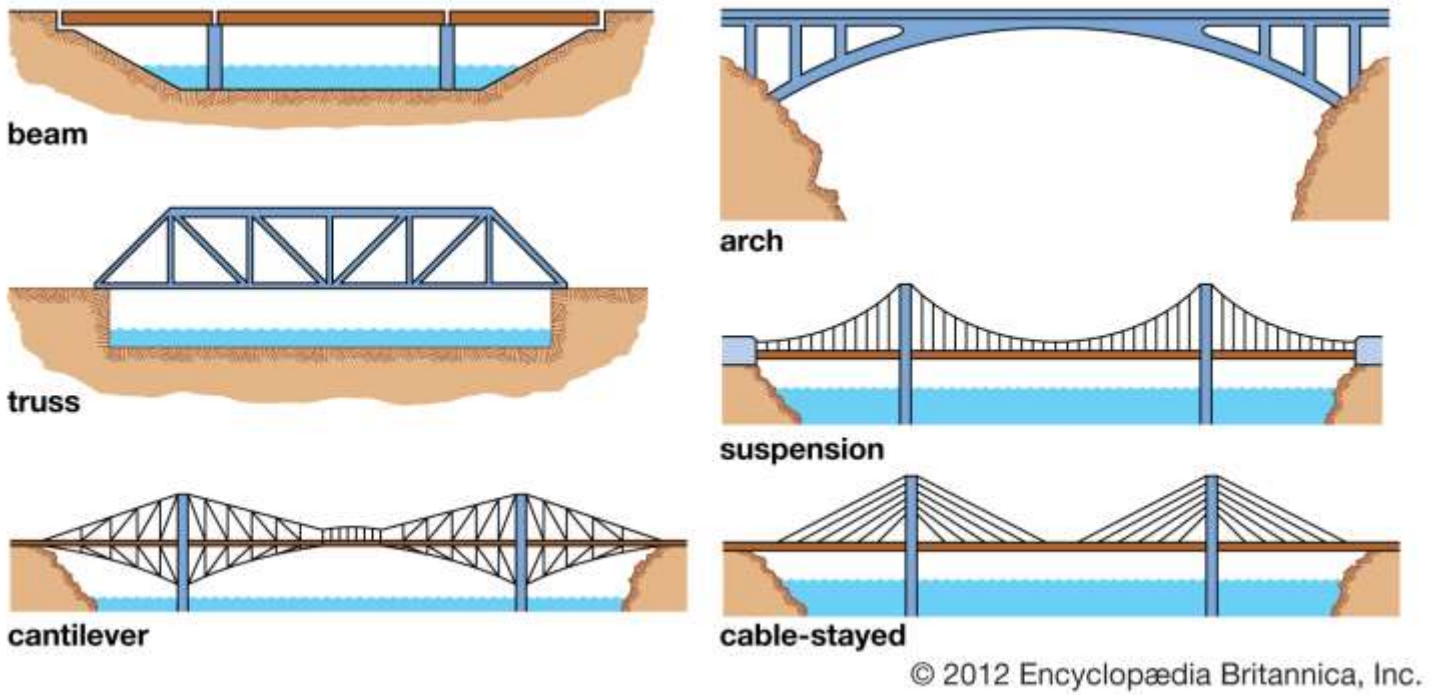
Requirements engineering

- Provide the basis for all that follows on a software project
- Product foundations in CMMI - DEV - v1.2, for example, include requirements development and requirements management

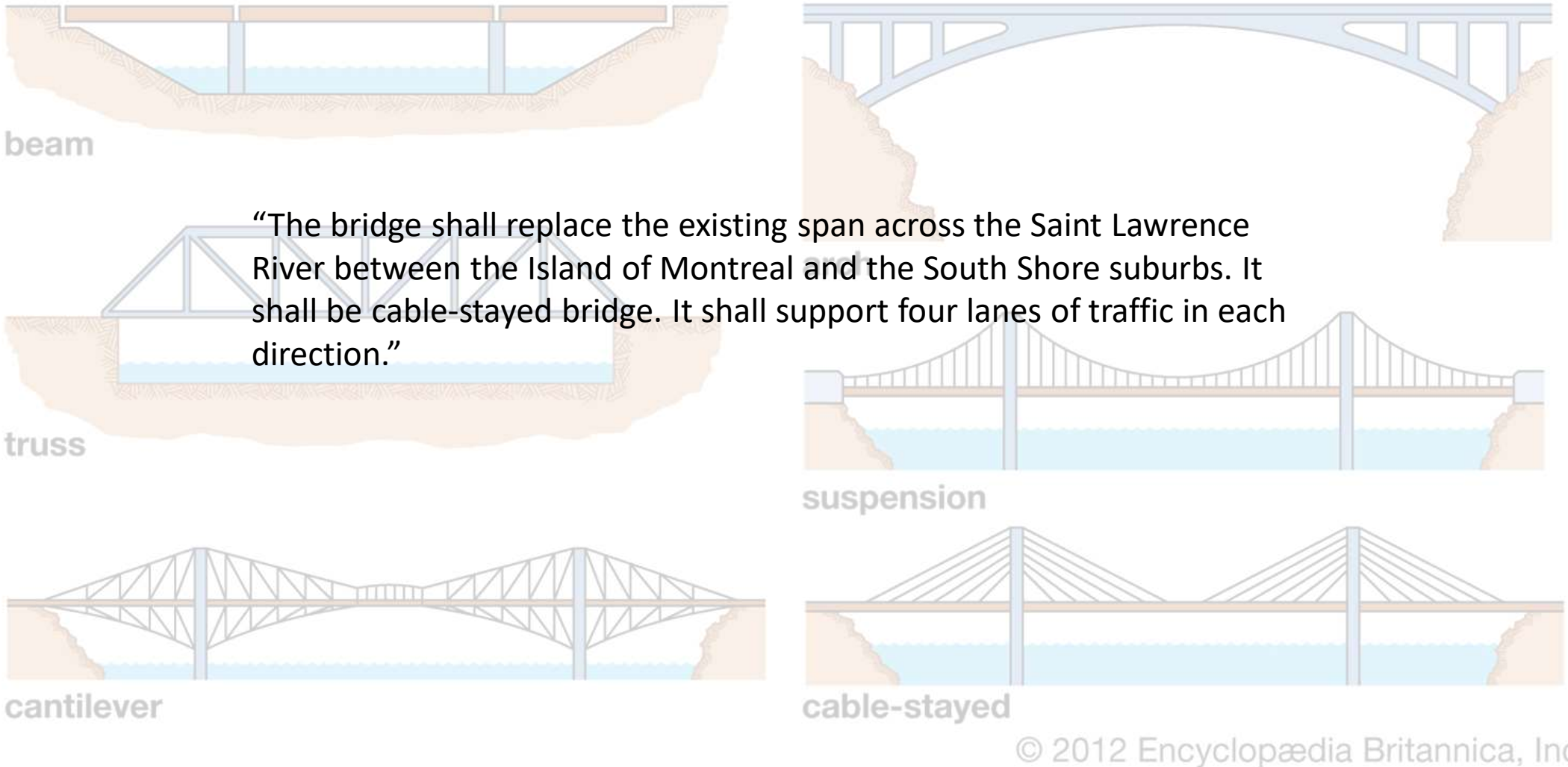
A simple example: connecting two lands



How many options?



Requirements for a bridge (example)



"The bridge shall replace the existing span across the Saint Lawrence River between the Island of Montreal and the South Shore suburbs. It shall be cable-stayed bridge. It shall support four lanes of traffic in each direction."

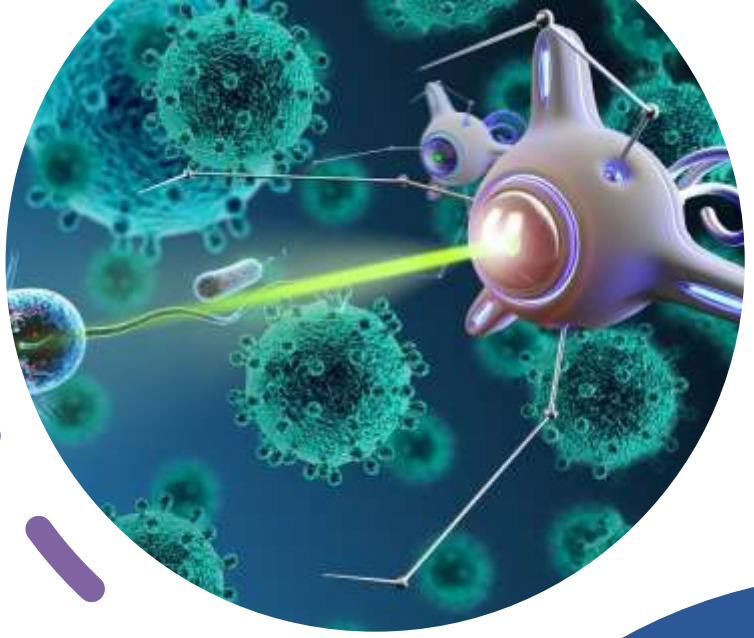
Requirements engineering (RE) is about

WHY SUCH A PROBLEM NEEDS TO BE SOLVED?

WHAT PROBLEM SHOULD BE SOLVED?

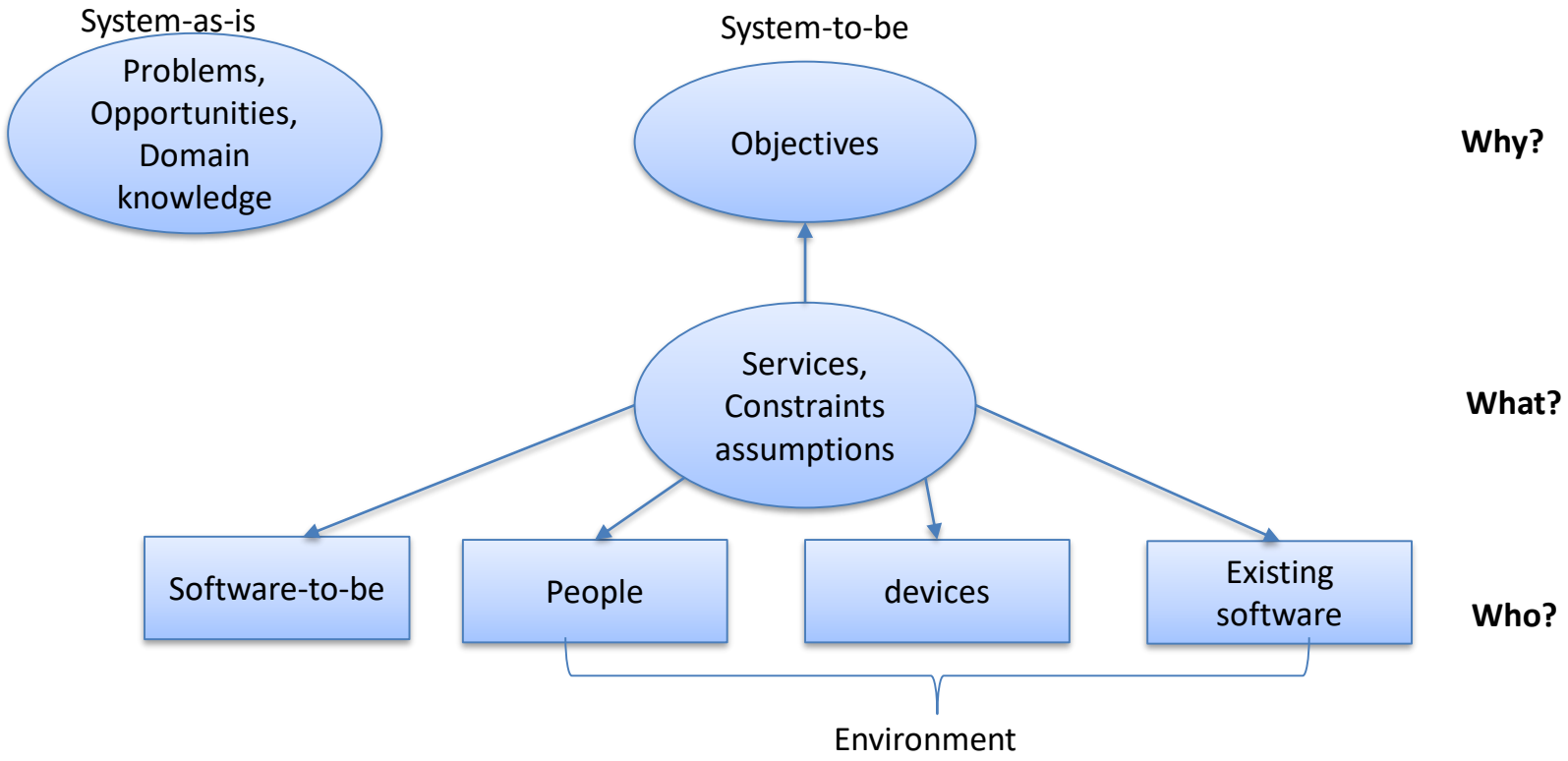
WHO SHOULD BE INVOLVED IN THE RESPONSIBILITY OF SOLVING THAT PROBLEM?

Requirements on complex systems



- Systems, such as biomechanical or nanotechnology systems with highly specialized domain language, have seemingly exotic requirements and constraints
- Still other complex systems have so many kinds of behaviors that need to be embodied (even word processing software can support thousands of functions) that the specification of said systems becomes very challenging indeed

The three dimensions of RE in software engineering



The job of the requirements engineer

{***SOFREQ***,**ASM**, ***DOM***} I= *SysReq*

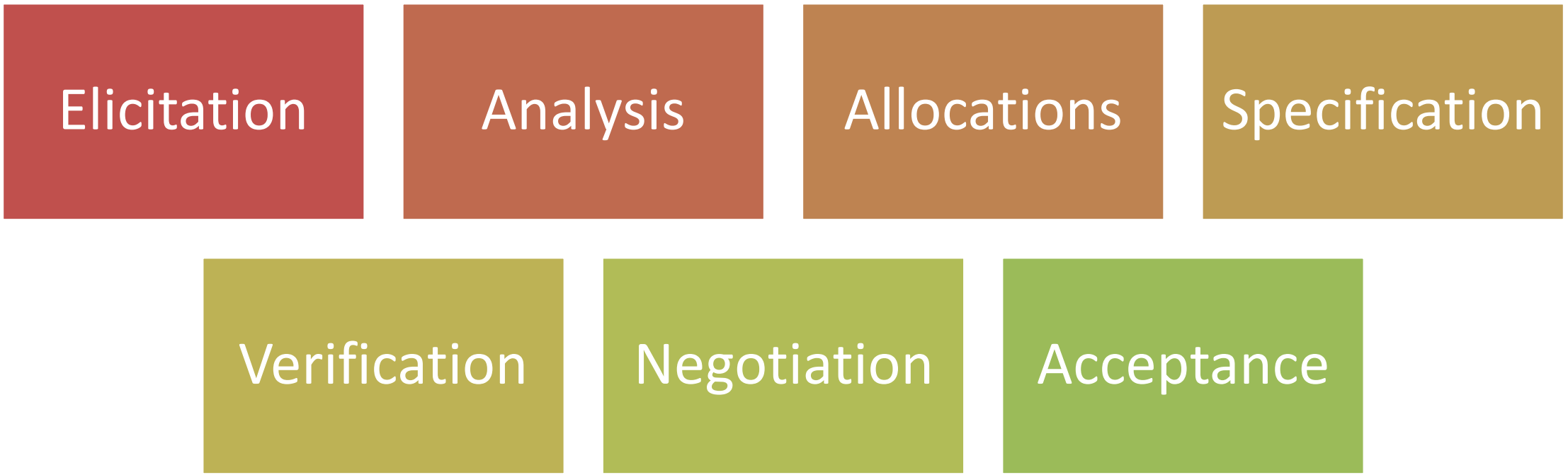
which reads:

If **the software requirements in set *SOFREQ* are satisfied by the software**, **the assumptions in set *ASM* are satisfied by the environment**, **the domain properties in set *DOM* hold** and all those statements are consistent with each other, **then the system requirements *SysReq* are satisfied by the system**

Stakeholder(s)

- A group or individual affected by the system-to-be, who may influence the way this system is shaped and has some responsibility in its acceptance
- They may include strategic decision makers, managers of operational units, domain experts, operators, end-users, developers, subcontractors, customers, and certification authorities

Requirements development

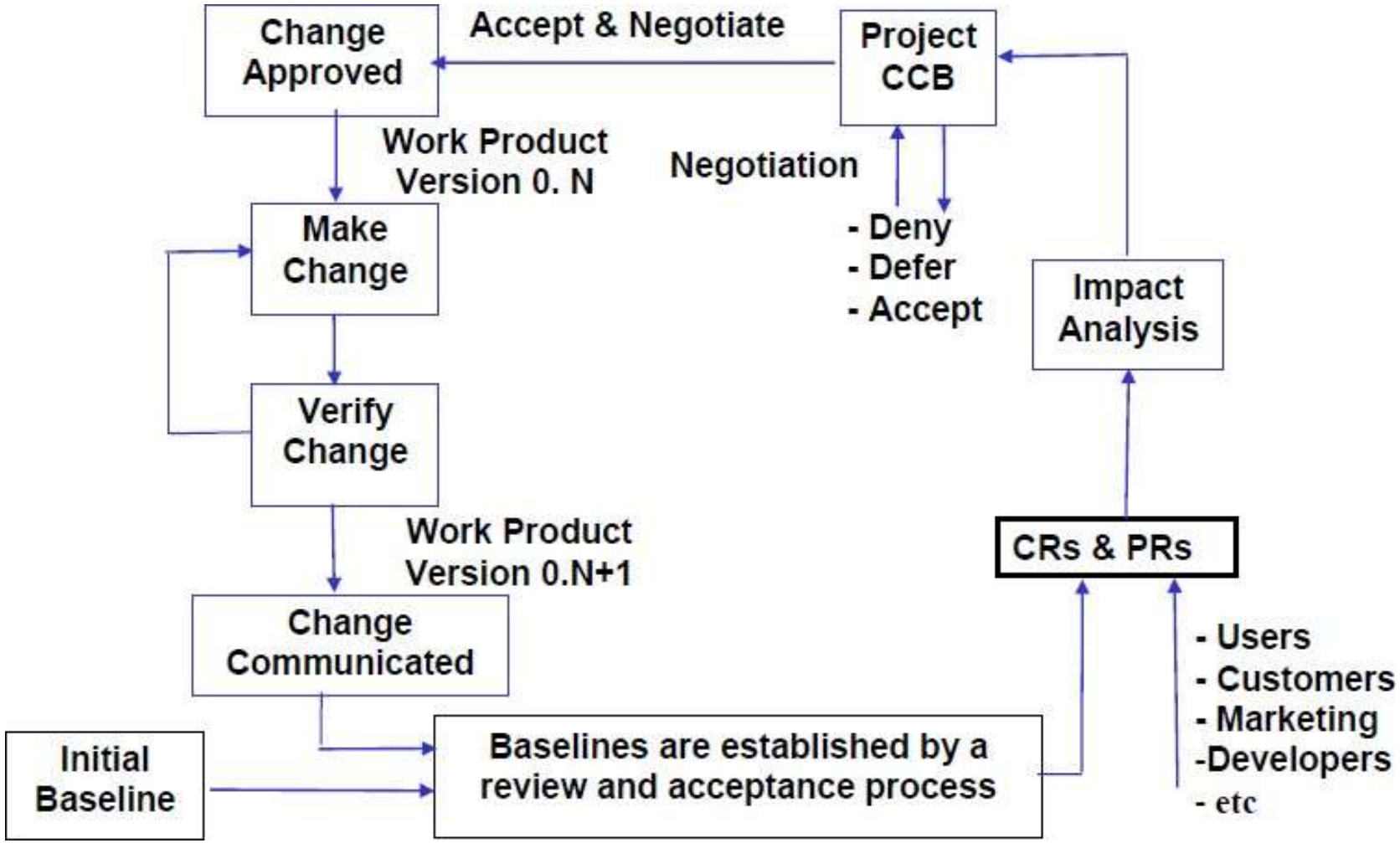


Requirements management

- Concerned with baseline control of requirements and with maintaining consistency among requirements, effort, schedule, budget, and technology

In software development, Change control board (CCB) is a group of experts who decide whether include or not proposed changes to a software project

Requirements Management and Change Control Boards (CCB)



Documenting Contractual Agreements

- **scope of work to be performed**
- deliverable work products
- delivery date(s)
- customer-user-developer review schedule
- change request procedures
- design constraints
- development constraints
- product acceptance criteria
- training schedule (as appropriate)
- price and payment schedule (as appropriate)

Project management constraints (revisited)

Scope Management : Time, Resources, Functionality

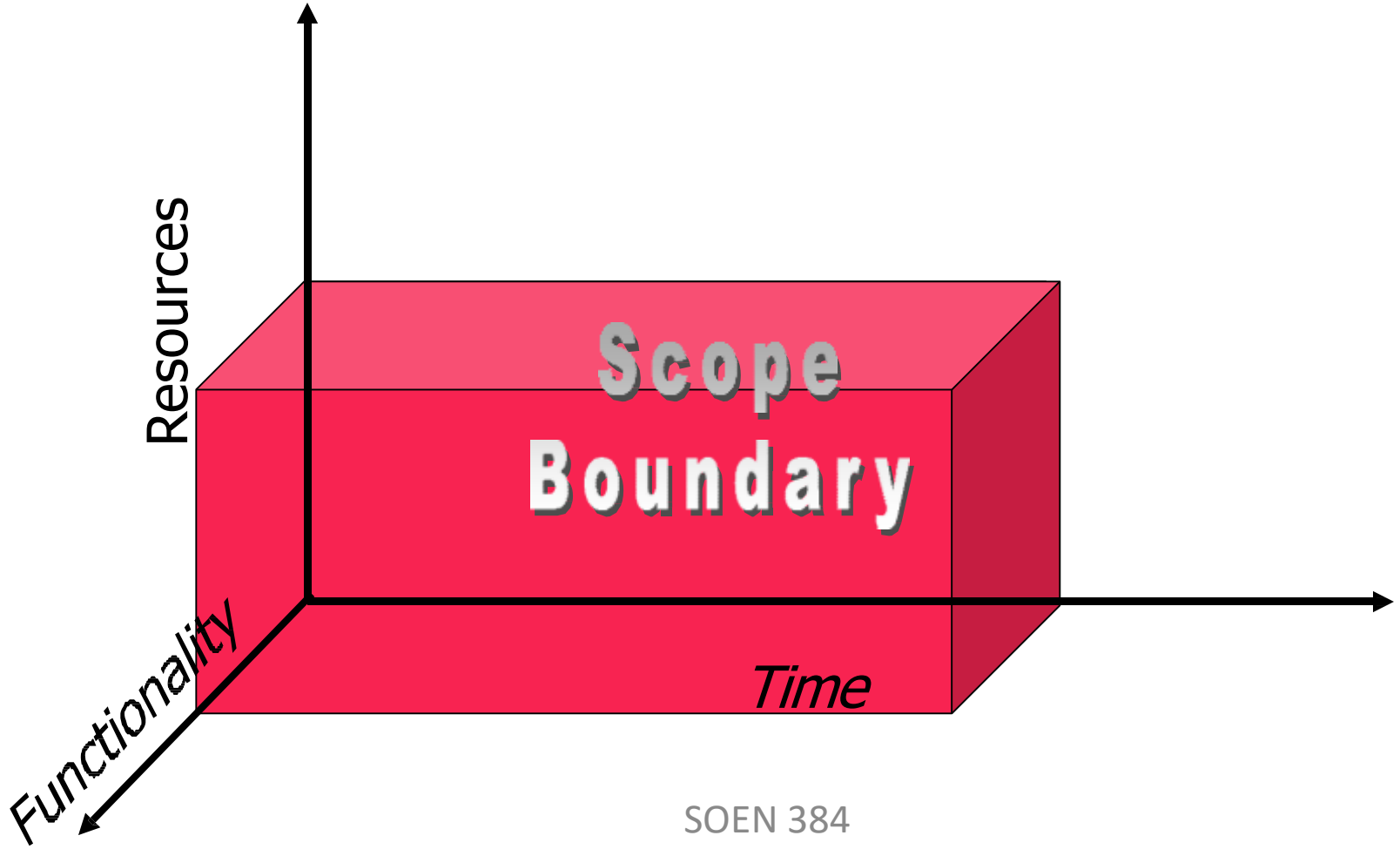
Scoping: define the boundaries of the project

- **Time is "Soft" Boundary**
  - Time (raw) estimation
  - Subject to change
- **Resources**
  - Remember Brook's Law!
- **Functionality**
  - Functional Size of the system features (estimated number of instructions, COSMIC, FPA, etc.)

Brooks' law

- It is an observation about software project management according to which "**adding human resources to a late software project makes it later**“
- It was coined by Fred Brooks in his 1975 book *The Mythical Man-Month*

Scope Management : Dimensions



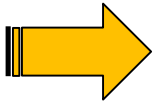
Scope assessment:
Very Simplified Equation

Let

- **R:** resources available for project **(# of developers**)
- **T:** development time (**hours**) available to work on project
- **Effort(f) :** effort **(person-hours)** required to realize feature *f*.

Project Scope Management

- How to assess project scope?
  - Use **Ratio (%)** as an indicator



$$\frac{\sum_{f \in Feature} effort(f)}{R \times T}$$

- Project scope < 100% is acceptable:

$$\sum_{f \in Feature} effort(f) < R \times T$$

Budget

- Estimated budget for a project:

$$Cost = \sum_{f \in Feature} effort(f) \times salary(+overhead)$$

How to use effort in planning? Examples

- Example 1: *A person working full time in a small program for 1 day:*
  - **Duration of the activity = 1 workday***
  - **Work Effort = 8 person-hours**
- Example 2: *Five persons working full time for six weeks in a project, 35 hours of work per week.*
  - **Duration = 6 weeks**
  - **Work Effort = (5 persons * 6 weeks * 35 hours/week) = 1,050 person-hours**

Why Effort & Cost Estimation?

- Planning new development work (**schedule, effort, cost, people**)
- Estimating how many features you can deliver within a specific development iteration
- Estimating time needed for defect correction work (maintenance)
- Estimating number of developers
- …

Art vs Science of Software Estimation

- **Science:**
  - Complex formulas, requires experience
  - Requires estimation tools for calibration and estimation
  - Goal: accuracy can achieve +- 5%
- **Art:**
  - Rules of thumb, simple formulas, 5th grade arithmetic at most
  - Accuracy: +- 25%

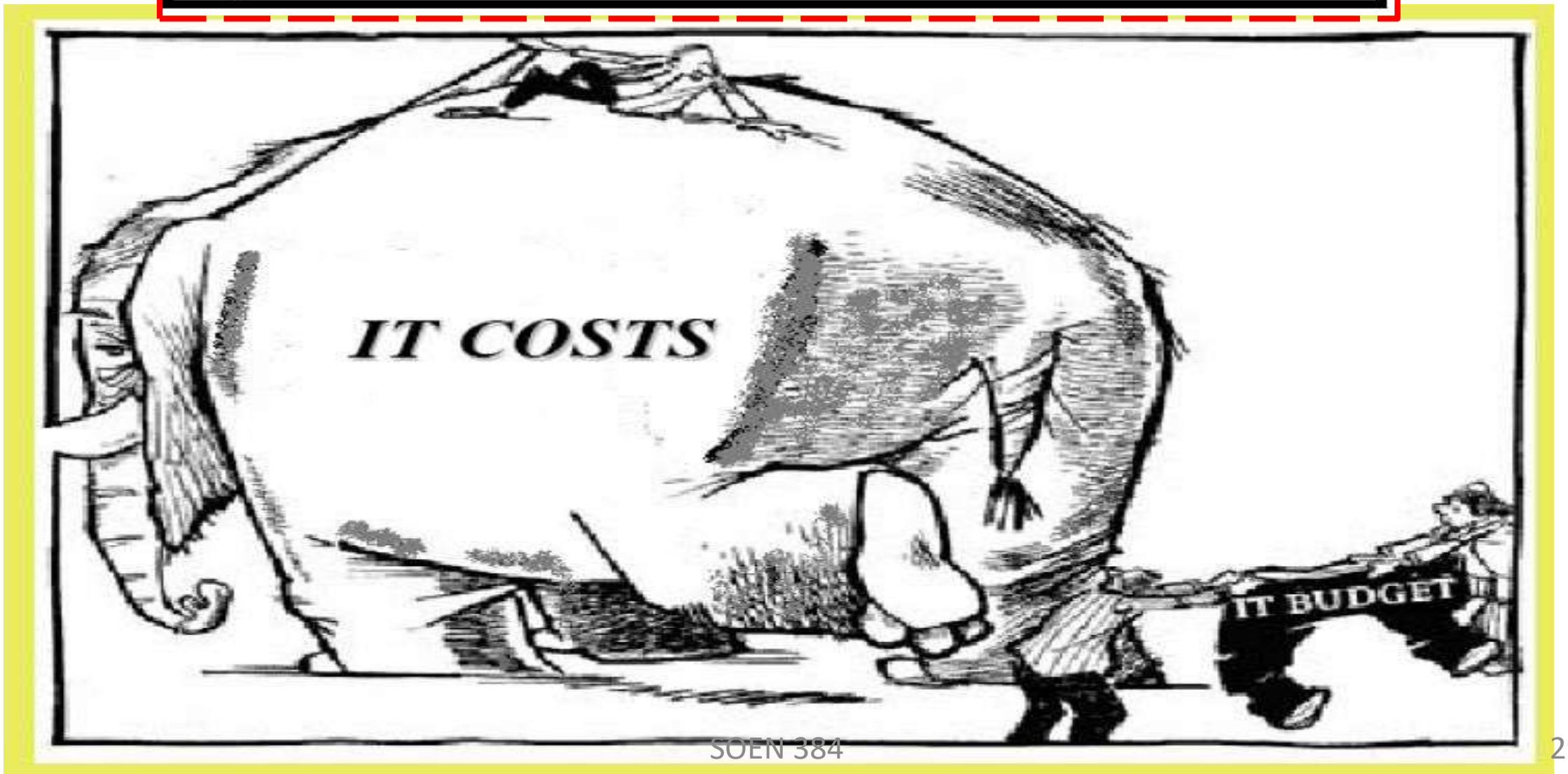
Overscoping: Very Simplified Equation

Let

- **R:** resources available for project **(# of developers**)
- **T:** development time (**hours**) available to work on project
- **Effort(f) :** effort **(person-hours)** required to realize feature *f*.
- A project is in jeopardy (**overscoped**) if

$$\sum_{f \in Feature} effort(f) \geq R \times T$$

$$\sum_{f \in Feature} effort(f) \geq R \times T$$



Project Scope Management

- Project scope > 100% is a **problem.**
  - What can we do in such a situation?
- Solution: **REDUCE SCOPE**
  - Define a feature subset, ***B*** (release/project **baseline**) such that

$$\sum_{f \in B} effort(f) < R \times T$$

Requirements Baseline

- Definition:

  The itemized set of features intended to be delivered in a specific version of the application

- Must be agreed upon by primary stakeholders, especially customer and developer representatives.
- Achievable (25 – 50?) features

Steps to Defining a Requirements Baseline

- Have feature list at hand
- ***Assign priority:*** for each feature:
  - Establish ***business value***
  - Assess ***effort*** **(person-hours)** required to develop feature.
  - Assess ***Risk*** involved in developing feature.

Assessing Scope: Effort

- Difficult to do early in the project when
  - Requirements have not been fleshed out in sufficient detail
  - Design (alternatives) do not exist or are unclear.
- Still, it is crucial to provide a rough estimate (what happens if we don't?)
- Scales for effort,
  - low, medium, high
  - Person-months

Assessing Scope: Risk

- Based on experience
- Factors to consider:
  - New unproven technologies being used?
- Scale often:
  - High, Medium, Low

Requirements Priority

- **Critical** means the requirement must be incorporated in the next product release.
- **Important** means the requirement is necessary but it can be deferred to a later release if necessary.
- **Useful** means it would be nice to have, but we realize it might have to be dropped if we have insufficient time or resources.

Example of an Assignment of Priorities

Feature	Priority
F1: external relational DB support	Critical
F4: portability to a new OS release	Critical
F6: import of external data by style	Critical
F3: ability to clone a project	Important
F2: multiuser security	Important
F5: new project wizard	Important
F7: implementation of tool tips	Important
F8: integration with a version-manager subsystem	Useful

Requirements Prioritization techniques

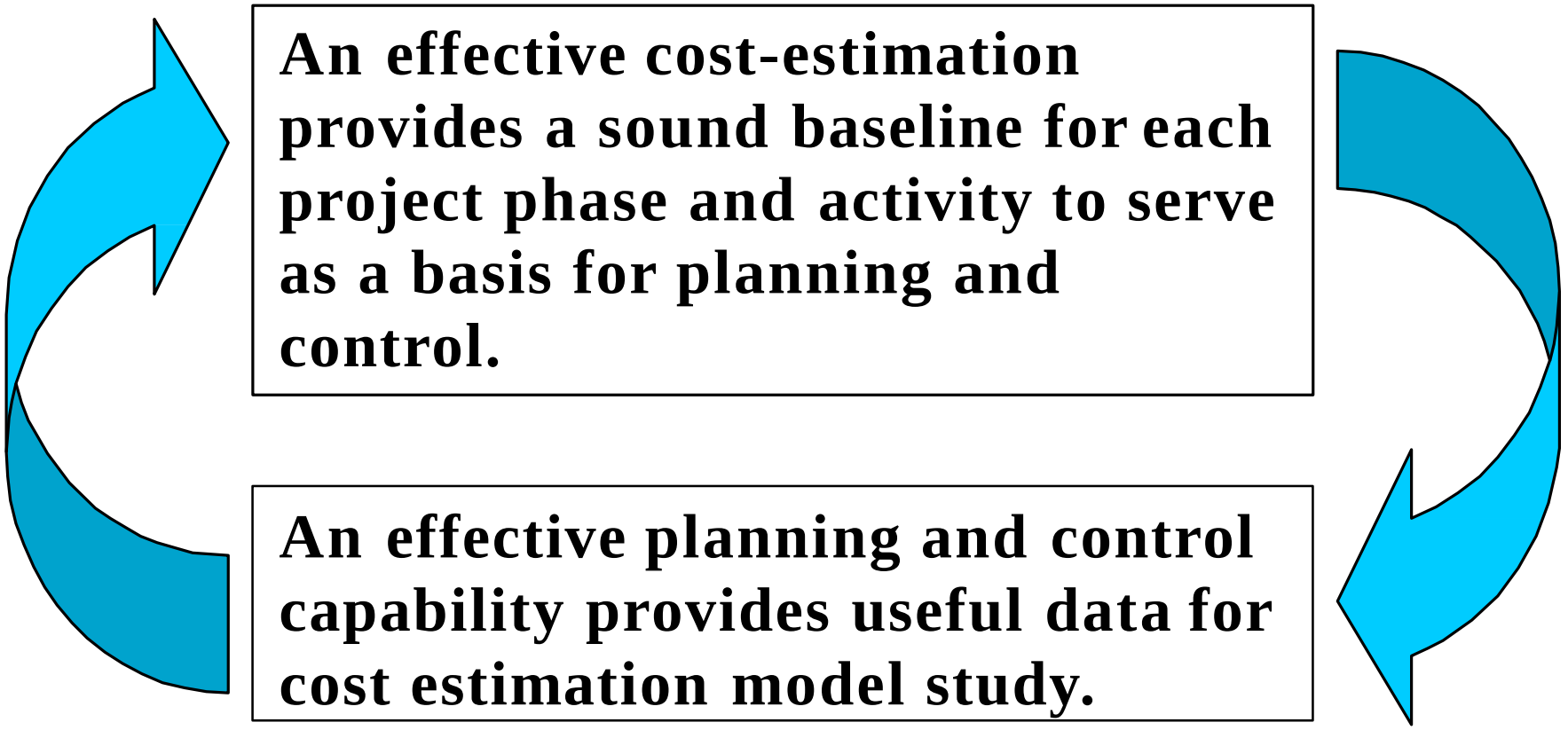
- Ranking (rank requirements using an ordinal scale)
- Grouping (group requirements based on priority groups: critical, important, and useful)
- Hundred dollar method (stake holders get a conceptual 100 bucks to distribute among requirements)

Requirement Prioritization techniques (cont.)

Which prioritization technique is the best?

- "*The one that fits your needs the most and accomplishes your goals in the least amount of time and with minimal amount of resources*"

Why Cost-Effort Estimation?
Key point: Project Planning



Summary

- Software projects have four kinds of product foundations and four kinds of process foundations
- Requirements provide the basis for all that follows on a software project
- Software requirements include operational requirements and technical specifications
- The primary activities of requirements development are elicitation of operational requirements, analysis, translation into technical specifications, and acceptance baselining of the technical specifications
- Scope management define project boundaries