

### Requirements Discipline - Outlines

- ◆ **Presenting the Scope of Requirement**
  - Requirement captures the basic understanding of the stakeholders. It must be expressed in a way that facilitates communication, enables validation and supports change
- ◆ **Defining the Requirement Components**
- ◆ **Eliciting the Requirements**
- ◆ **Evolving Requirements**

© 2000 École Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-1

### Purpose of Requirements Discipline

- ◆ Basis of communication between all parties
- ◆ Contractual agreement between parties
- ◆ **Input**
  - to design team
  - to software test
  - to quality assurance
  - to user documentation
- ◆ The software manager's reference
- ◆ Controls the system's evolution



**Software Requirements Specification (SRS)**

Adapted from Alan Davis  
© 2000 École Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-2

### Stakeholders

A stakeholder is any individual or organization which is materially affected by the outcome of the system.

- partners
- users
- customers
- domain experts
- industry analysts
- developers

© 2000 École Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-3

### Requirements Discipline Consists of Five Activities

**Understand the problem**



Elicit Stakeholder Requests



Find Actors and Use Cases

**Define the system**



Structure the Use-Case Model



Detail a Use Case

**Review**



Review Requirements

© 2000 École Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-4

### Artifacts are Divided into Three Groups

The diagram illustrates the classification of requirement artifacts into three groups:

- Input:** Vision, Glossary
- Descriptive:** SRS, Supplementary Specification, Glossary
- Modeling:** Use Case, Use-Case Model

© 2000 École Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-5

### Requirements - Lecture Outline

- ◆ Presenting the Scope of Requirement
- ◆ Defining the Requirement Artifacts
  - Vision document
  - Glossary document
  - Functional requirements
  - Non-functional requirements
- ◆ Eliciting Requirements
- ◆ Evolving Requirements

© 2000 École Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-6

### High Level Requirements

- ◆ System-level documentation which describes the “Whats” and “Whys” of the product or application
- ◆ Focus is on:
  - User needs
  - Goals and objectives
  - Target markets
  - User environments and platforms
  - Product features

Vision Artifact

**An artifact which gets “all parties working from the same book”**

© 2000 École Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-7

### Characteristics of a Well Constructed SRS

- ◆ Correct
  - every requirement contributes to the satisfaction of needs
- ◆ Complete
  - contains all significant requirements, responses to all inputs and full labels and references
- ◆ Consistent
  - no subset of individual requirements is in conflict.
- ◆ Unambiguous
  - every requirement within it has only one interpretation
- ◆ Ranked for importance and stability
  - identifier indicates its importance and stability Verifiable
- ◆ Modifiable
  - changes can be made easily, completely, and consistently.
- ◆ Traceable
  - facilitates the Backward and Forward referencing

ref - IEEE 1993

© 2000 École Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-8

### Supplementary Specification Attributes

- ◆ **Usability:**
  - The ease by which the software can be learned and operated
- ◆ **Reliability:**
  - The ability for the software to behave consistently
- ◆ **Performance:**
  - A measure of speed and efficiency of the running system.
- ◆ **Supportability:**
  - The ability of the software to be easily modified to accommodate enhancements and repairs
- ◆ **Design constraint**
  - A requirement that leaves no options for design

**UPEDU Concept: Requirements**

© 2000 École Polytechnique de Montréal & Rational Software    Learning software process with UPEDU    Slide 4-9

### Use-Cases Elicit and Clarify Requirements

© 2000 École Polytechnique de Montréal & Rational Software    Learning software process with UPEDU    Slide 4-10

### Notions Behind Use-Cases

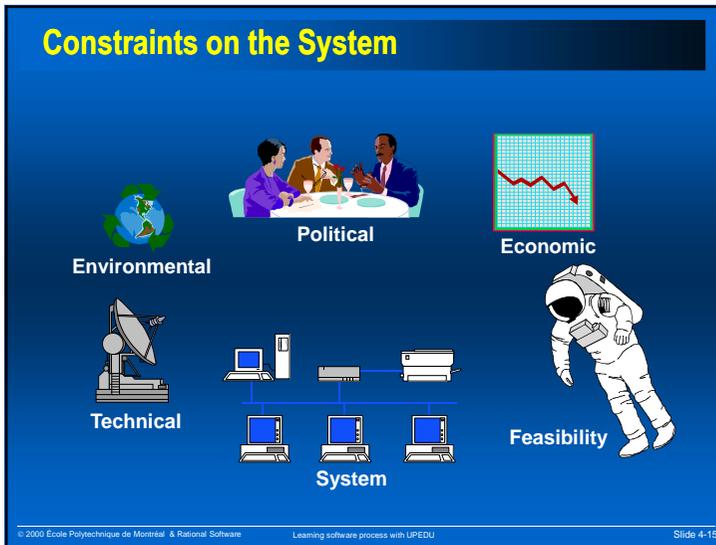
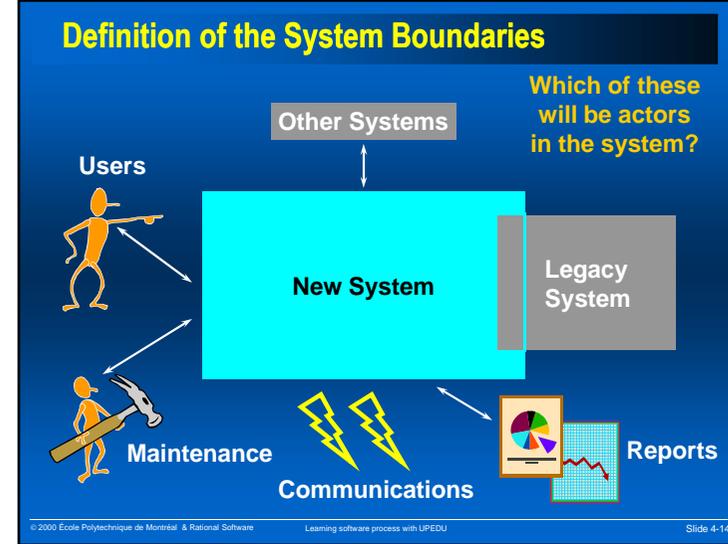
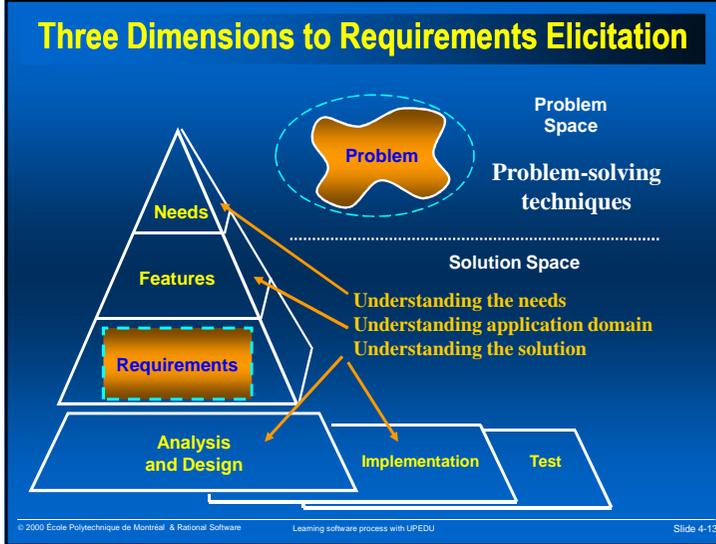
**UPEDU Guideline: Actors**  
**UPEDU Guideline: Use-Case**  
**UPEDU Guideline: Use-Case Model**  
**UPEDU Guideline: Use-Case Diagram**

© 2000 École Polytechnique de Montréal & Rational Software    Learning software process with UPEDU    Slide 4-11

### Requirements - Lecture Outlines

- ◆ Presenting the Scope of Requirement
- ◆ Defining the Requirement Components
- ◆ **Eliciting Requirements**
  - Delimiting boundaries
  - Interviews and questionnaires
  - Requirements workshop
  - Prototyping
- ◆ Evolving Requirements

© 2000 École Polytechnique de Montréal & Rational Software    Learning software process with UPEDU    Slide 4-12



- ### Direct Person-to-Person Interviews
- ◆ Don't ask people to describe things they don't usually describe.
  - ◆ Ask open-ended questions
  - ◆ Avoid questions that begin with "Why...?"
  - ◆ Don't expect simple answers
  - ◆ Don't rush the interviewee for answers
  - ◆ **Listen, listen, and listen!**
  - ◆ **Questionnaires are not a substitute for an interview**
- © 2000 École Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-16

### Conducting a Requirements Workshop

- ◆ Accelerate the Elicitation Process
- ◆ Gathers stakeholders together for an intensive, focused period
- ◆ Everyone gets their say
- ◆ Results are immediately available
- ◆ Provides a framework for applying other elicitation techniques
  - Brainstorming
  - Storyboarding
  - Role-playing
  - Review existing requirements



© 2000 Ecole Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-17

### Several Varieties of Prototypes

- ◆ An early demonstration of some or all of the externally observable behaviors of a system
- ◆ Used to
  - Gain feedback on proposed solution
  - Demo the problem domain
  - Validate known requirements
  - Discover unknown requirements
- ◆ Prototyping Tools
  - Visual Basic, PowerSoft, Gupta, Access, Delphi
  - Toolkit
  - Demo programs
  - Simulations

Throw-away

Evolutionary

Operational

© 2000 Ecole Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-18

### Requirements - Lecture Outline

- ◆ Presenting the Scope of Requirement
- ◆ Defining the Requirements Components
- ◆ Eliciting Requirements
- ◆ Evolving Requirements

© 2000 Ecole Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-19

### Minimizing the Impact

- ◆ Effective problem analysis and elicitation of user needs
- ◆ Gain agreement with the customer/user on the requirements
- ◆ Model interaction between the user and the system
- ◆ Establish a baseline and change control process
- ◆ Maintain forward and backward traceability of requirements
- ◆ Use an iterative process
- ◆ Make reviews
  - Walkthrough
  - Inspection
  - Formal review

© 2000 Ecole Polytechnique de Montréal & Rational Software Learning software process with UPEDU Slide 4-20