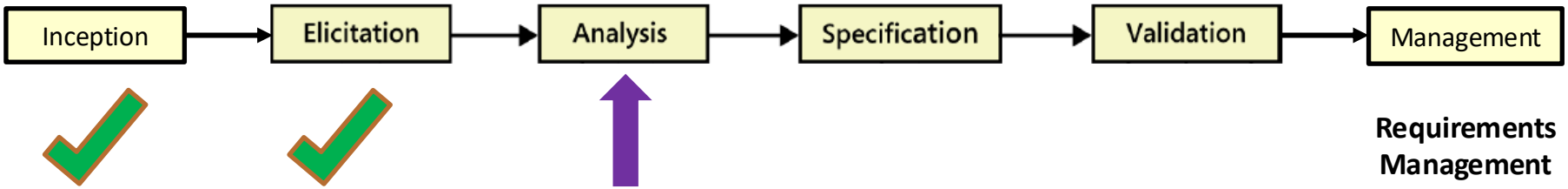
Requirements Analysis

RE Process


Inception
Elicitation
Analysis
Specification
Validation
Management
Requirements Management

Requirements Analysis Activities

Requirements Analysis Activities

- Capture Data Requirements: Domain Modeling
- Capture Functional Requirements: Use-Case Modeling
- Capture Nonfunctional Requirements
- Validate System Requirements (Discussed in Part 12)

Requirements Analysis Activities

- **Understand the application domain and identify user needs**
  - collect data on system requirements and system constraints
  - define development effort scope and system design goals

- **Determine the risks of developing the system**
  - economic
  - operational
  - legal
  - technical
  - organizational

- **Capture the system requirements**
  - data requirements → domain model
  - functional requirements → use-case model
  - nonfunctional requirements → supplementary text

- **Validate the system requirements**
  - verify the correctness and completeness of the system requirements

Our focus

SYSTEM REQUIREMENTS SPECIFICATION (SRS)

- The SRS documents the system requirements.
  - It is the official statement of what is required of the system.
  - It should include both a definition of user requirements and a specification of the system requirements.

- It is **NOT** a design document.
  - It should state **WHAT** the system should do, but not **HOW** it should do it.

NOTE: Many agile methods argue that producing a SRS is a waste of time as requirements change so quickly.

WAYS OF WRITING A SRS

- **Natural language**
  - Uses sentences supplemented by diagrams and tables.
- **Structured natural language**
  - Uses a restricted language that follows a fixed standard/template.
- **Graphical notations** **UML + structured text**
  - Uses graphical models plus text annotations.
- **Design description languages**
  - Uses a language like a programming language.
- **Mathematical specifications**
  - Uses notations based on mathematical concepts.

Requirements Analysis Activities

- **System Requirements Capture Activities**
  - **Capture Data Requirements: Domain Modeling**
  - **Capture Functional Requirements: Use-Case Modeling**
  - **Capture Nonfunctional Requirements**
  - **Validate System Requirements**

DOMAIN MODELING

- Captures the most important classes and their associations.
  - ☞ **Things that exist or events that occur for which data must be stored.**

- The classes and associations are found from user requirements:
  - requirements statement
  - domain experts (users)

- The classes can be:
  - business objects (e.g., orders, accounts, etc.).
  - real-world objects and concepts (e.g., suppliers, customers, etc.).
  - events (e.g., aircraft arrival/departure, sales, reservations, etc.)

Described in a class diagram.

Provides a glossary of terms.

DOMAIN MODELING:

IDENTIFYING CLASSES AND ASSOCIATIONS

- Naturally occurring things or concepts in the user requirements:
  - classes appear as **nouns/noun phrases**
  - associations appear as **verbs/verb phrases**

☞ **Put all terms into singular form/active voice.**

- Identify only relevant classes/associations.
  - ➢ Those that are *essential* and *will always exist*; *are not transient*.

☞ **This leads to a stable system.**

The decomposition of user requirements into classes and associations depends on judgment and experience and the nature of the problem.

There is usually no one correct decomposition!

DOMAIN MODELING: IDENTIFYING ATTRIBUTES

- Attributes usually correspond to nouns followed by possessive phrases

  e.g., password of student; student's address.

- Adjectives usually represent specific enumerated values

  e.g., Fall term; PhD degree.

- Only identify attributes that directly relate to the application domain.

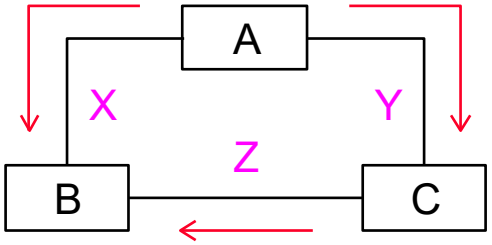
Most attributes will not be given in a requirements statement and will have to be obtained from domain experts or application documentation.

DOMAIN MODELING: EVALUATING CLASSES

- Are any classes irrelevant to the domain model? → **eliminate**
  - ☞ **A class should represent persistent data, not transient data.**
- Are any classes vague (ill-defined)? → **make specific or eliminate**
  - ☞ **A class should be specific and well-defined.**
- Are any classes redundant? → **keep most descriptive**
- Should any classes really be attributes? → **consider application requirements**
  - ☞ **Is the concept dependent or independent?**
- Do any class names describe a role?
  - ☞ **Roles also appear as nouns!**
- Do any classes describe an operation/action? → **eliminate**
- Do any classes describe implementation constructs? → **eliminate**

DOMAIN MODELING: EVALUATING ASSOCIATIONS

- Are any associations irrelevant to the domain model? → **eliminate**
- Do any associations describe an operation? → **eliminate**
  - ☞ **Should describe a structural property, not a transient event.**
- Can ternary associations be decomposed into binary associations? → **decompose**
  - ☞ **Should try to keep the model as simple as possible.**
- Are any associations derived associations? → **eliminate**
  - ☞ **Introduces redundancy into the model.**



E.g., is association X ≡ associations Y and Z?

If yes, then we can derive X from Y and Z.

- Do any associations describe implementation constructs? → **eliminate**

DOMAIN MODELING: EVALUATING ATTRIBUTES

- Are attributes closely related to the class they are in?
  - ☞ **A class should be simple and coherent.**
- Should any attributes really be classes?
  - ☞ **The concept should be dependent on the class.**
- Should any attributes be in an association class?
  - ☞ **Does the attribute depend on the existence of an association?**
- Have object identifiers been included as attributes?
  - ☞ **Object identifiers should not be included in the domain model!**

DOMAIN MODELING: DETAIL

- For each *class* we specify its **attributes**.
  - Each class is assumed to have two standard operations, get and set, which *do not need to be specified*.

- For each *attribute* we specify
  - name (unique within a class)
  - type (e.g., string, integer, date, etc.)
  - multiplicity (if greater than 1)

- For each *association* we specify
  - name → should say "what" not "how" or "why"
  - role names (if needed)
  - multiplicity (if known)
  - association class (if needed)

SYSTEM REQUIREMENTS ANALYSIS

DOMAIN MODELING EXERCISE

1- Movie Shop System (5.1.1)

2- PSU Registration System(5.1.2)