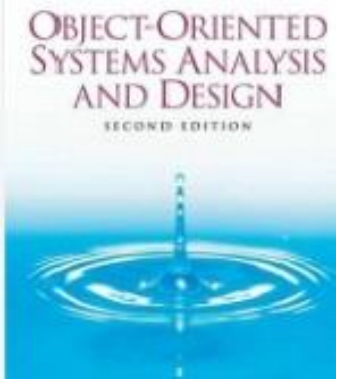
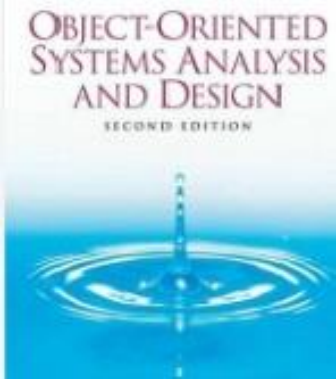
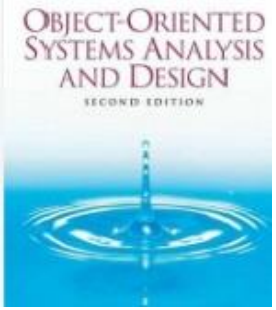


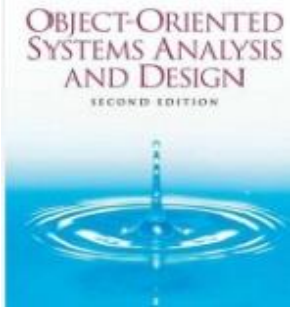




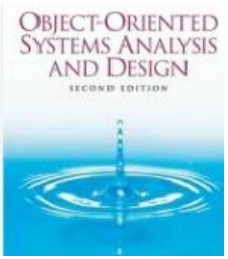
Object Oriented System Analysis and Design (OOSAD)
INSY3063



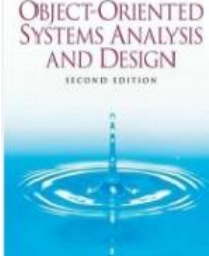
CHAPTER IV



Chapter Four: Requirement Gathering and Modeling



CHAPTER FOUR: REQUIREMENT GATHERING AND MODELING



Outline of topics

4.1. Define requirement and related motivational concepts?

4.1.1. Define requirements.

4.1.2. Requirements discovery vs. requirements gathering in general.

4.2. Classifying requirements

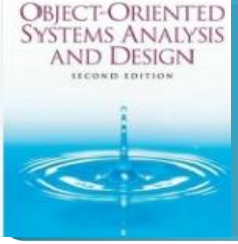
4.3. Fundamental Techniques for requirements Elicitation

4.3. Ensuring Your Requirements Are correct: Requirement validation Techniques

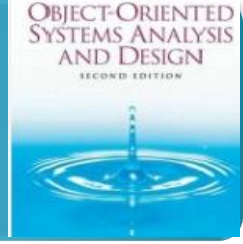
4.4. Essential Use Case Modeling

4.5. Domain modeling with class responsibility collaborator (CRC) cards

4.6. Essential User Interface Prototyping

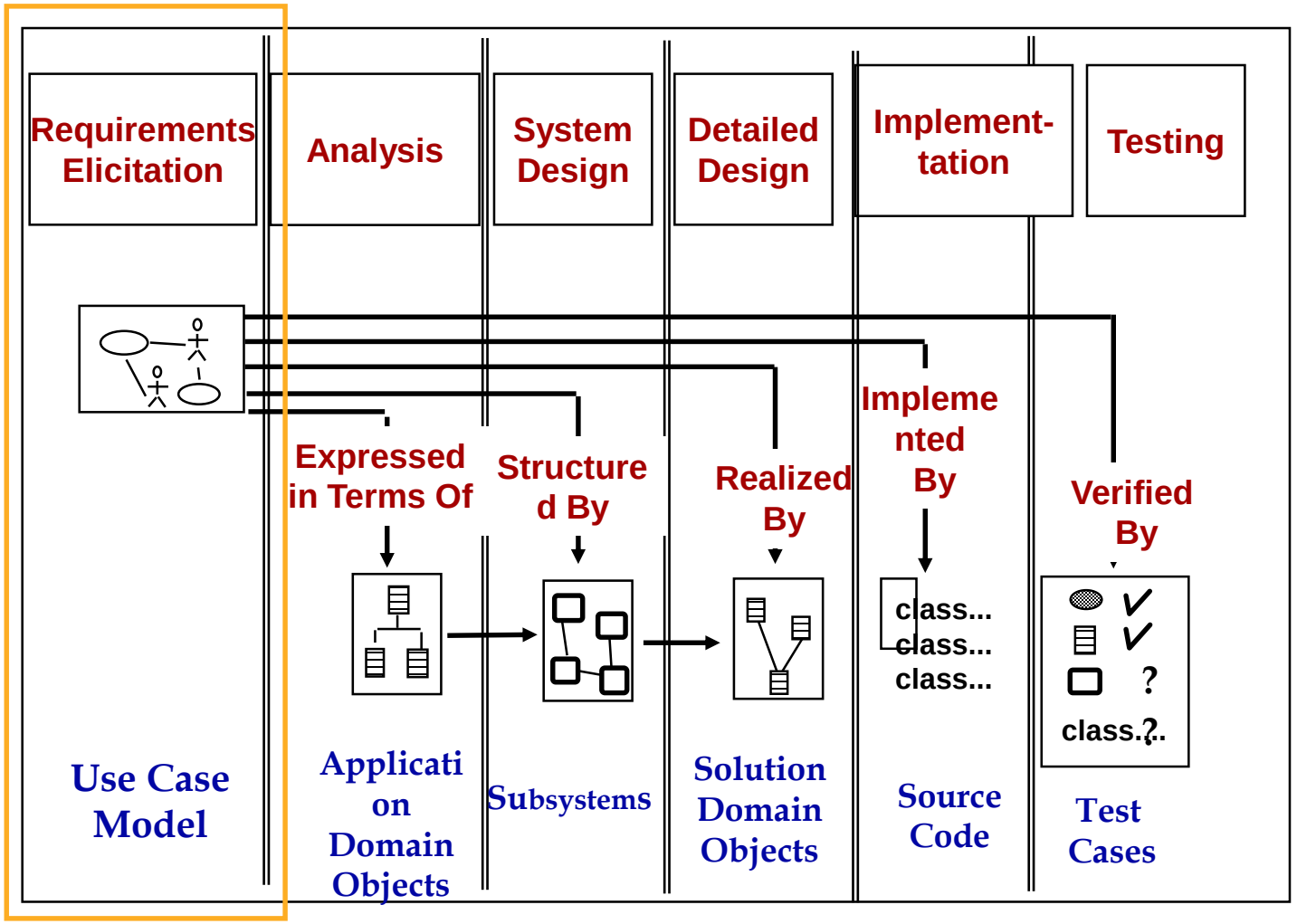


Objectives



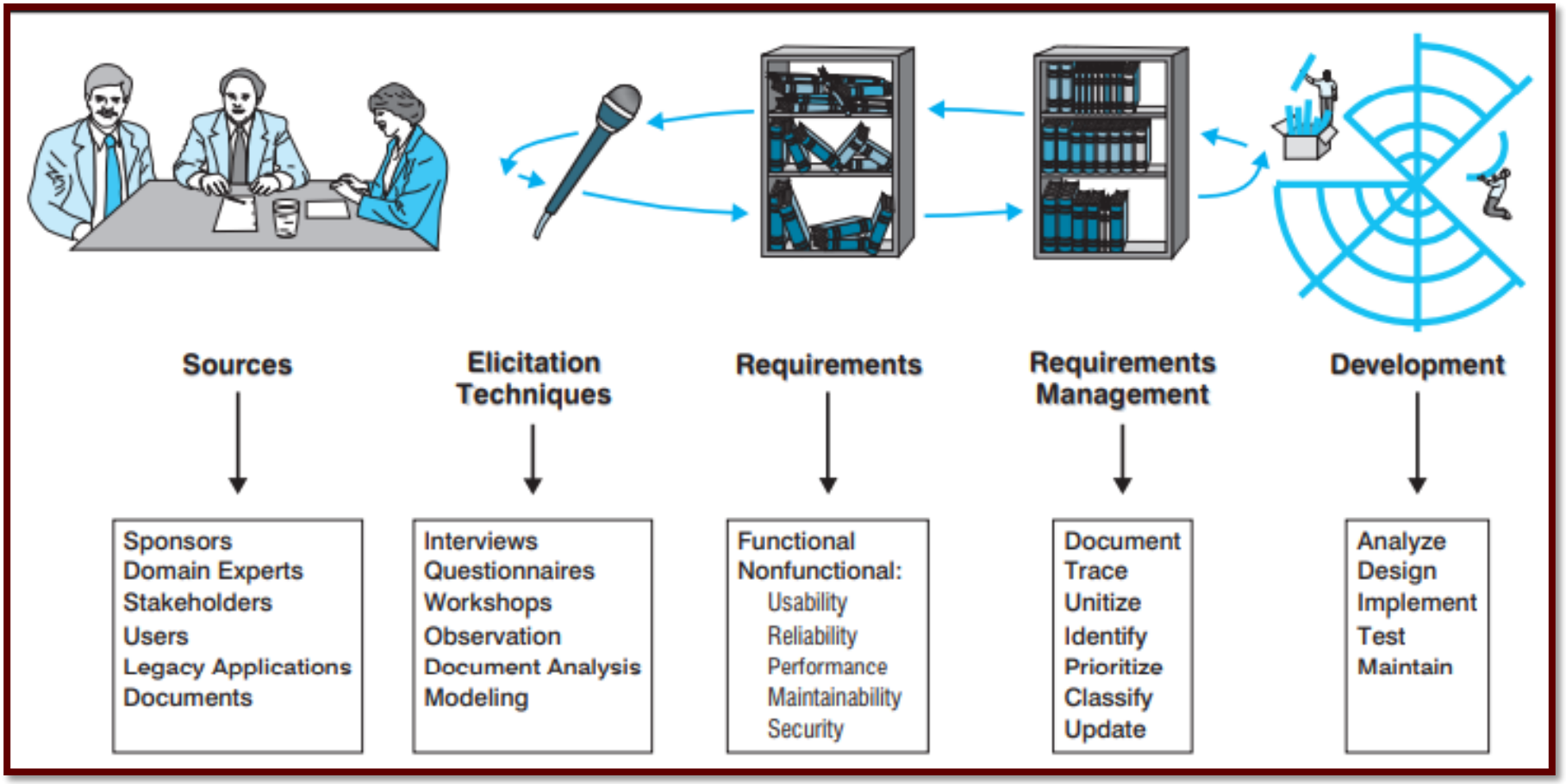
- Define requirement and related concepts?
- Understand requirement identification and gathering techniques
  - Using traditional method
  - Essential use case modeling
  - CRC modeling
  - Essential user interfaces modeling
  - Supplementary specifications
- Being familiar with validating, organizing and documenting requirements.

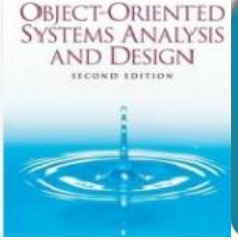
Software Lifecycle Activities



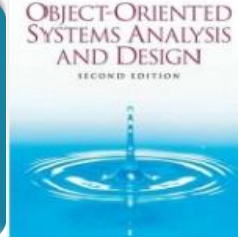
Putting together requirements gathering team

- **Requirements Gathering: An Ongoing Activity.**



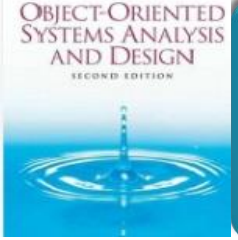


Introduction Define requirement and related motivational concepts?

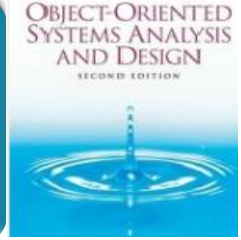


Define requirements.

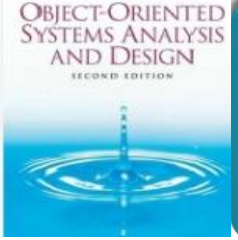
- In **a model-driven and object-oriented approach** to software development, we use **models to capture, define, and represent requirements.**
- An information system is a product, and a product must have features that its **customer wants in order to achieve specific objectives**.
- Some features represent the **functionality of the product**, while **others are required by the customer, by law, by standards, or by a number of other considerations or rules**.
- **Requirements** identify the objectives that the information system must help its users to achieve.
- **Requirements** are **whatever a product does** and **whatever a product has that its users need to reach their goals.**



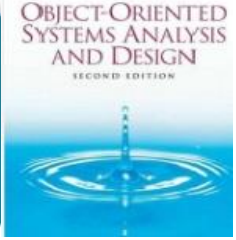
Introduction



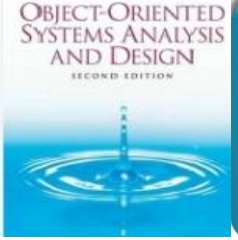
- **Define requirements.**
- Before anything is done to develop a product, we must identify what the product must do.
- This proposition seems reasonable and obvious, but in reality, **inadequate or wrong identification of requirements** is a major cause of **failure in software development.**



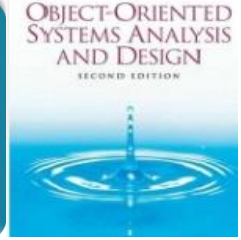
Introduction



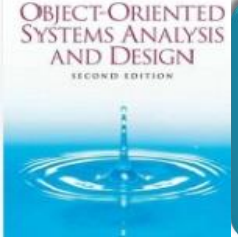
- **Define requirements.**
- Therefore, **understanding what requirements are, what they are not, and how best to identify them is crucial to the success of the product**
- **Requirements** identify the specific **objectives** that the product must help its users to achieve.
- **Requirements** are not product specifications.
- A finished product has **two set of features:**
  1. One set satisfies the objectives of the product (business requirements);
  2. The other set includes those features that are necessary to make the first set possible (solution features).



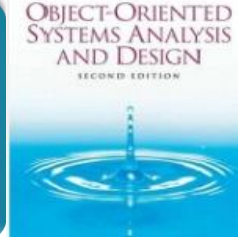
Introduction



- **Define requirements.**
- Both sets of product features are called "**requirements**." Such a use should not trigger a scholastic debate. Just remember that the two sets of features are not the same.
- A product has **two sets of features**: one set **satisfies requirements** while the other makes the **product possible**.
- **Requirements** are not the same as **product specifications**. The **former** express business needs and problems, whereas the **latter** include features that relate to the solution itself.



Introduction



- **For Example:**

1. **The ATM system must allow the customer to withdraw cash from his or her account.** (Among other things, the system must have cash and must be able to dispense it to achieve this goal.)
2. **The billing system must allow the account manager to inspect the billing activities of a client for the prior six months**. (The system must have at least six month's worth of transactions at its disposal and must be able to report it.)
3. **The online registration system** must **allow the student to register for desired classes.**

What Is Requirements Gathering?

- **The task of requirements gathering** is to collect and define all features that the information system must have in order to fulfill the objectives that the customer has set.
- **Gathering requirements** starts when the development project starts, but it is an ongoing activity, not a phase that is completed at a certain point in time.
- Without clear boundaries, **gathering requirements** would be in danger of missing its destination: It would end up with either irrelevant or insufficient information.

What Is Requirements Gathering?

- To arrive at the correct and relevant requirements, we must choose the right sources and **employ relevant elicitation techniques.**
- **Talent and experience** are, with out a doubt, very important to the gathering requirements, but they must be sharpened by **learning about techniques** and by **paying attention to detail**.
- Finally, requirements are useless to the development of the information system **if they are not correctly documented or cannot be traced, identified, or verified.**
- **The task of requirements management** is to ensure that requirements are **well-organized, easy-to-locate, reliable, and consistent**. Since it supports an ongoing activity, **requirements management is an ongoing task as well**

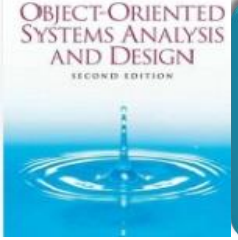
What Is Requirements Gathering?

- However, requirements specify **not only what the users can do** with the product, but also **what they cannot do.** In other words, requirements define **specific objectives and constraints on those objectives**.

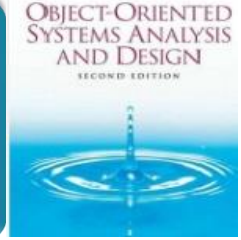
1. The user must be able to change his or her password (**objective**), **but no password can be less than 6 characters or more than 15 characters long (constraint).**
2. The customer must be able to use a credit card to order books (**objective**), but the **credit card must be in the customer's name (constraint).**
3. The student must be able to register for courses online (**objective**), but for each course the student **must have passed prerequisite courses (constraint).**

What Is Requirements Gathering?

- **Requirements can be complex or simple in relative terms**.
- That is, one requirement may translate into more than one requirement upon analysis.
- For example, to state that a student must be able to register online is to imply other requirements:
  i. **The student must be able to find and select the desired course.**
  ii. **The student must be able to drop a selected course.**
  iii. **The system must verify the student's identity and refuse access to imposters**.

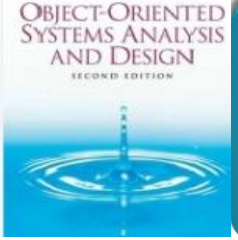


Introduction

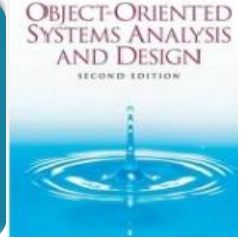


What Is Requirements Gathering?

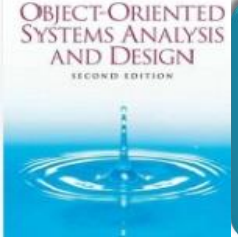
- **Gathering requirements** is an **ongoing process** that provides system development with **features and rules** that it must implement **to satisfy its objectives.**
- **The reliability and the correctness of requirements** is dependent on their
  - **Sources**
  - **The techniques that we employ to elicit and verify them, and**
  - **Effective management.**



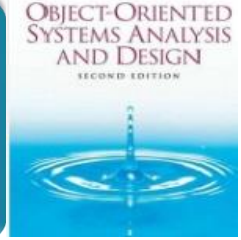
Introduction



- **What Is Requirements Gathering?**.
- **The task of requirements gathering** is to **collect and define all features** that the information system must have in order to fulfill the objectives that the customer has set.
- **Requirements gathering** must be done **thoroughly and correctly**; otherwise, we will build a product that might be **stylish and impressive**, but will not be what the customer wants.
- **Gathering requirements** starts when the development project starts, but it is an **ongoing activity**, **not a phase that is completed at a certain point in time.**

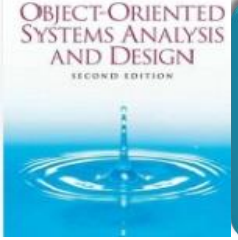


Introduction

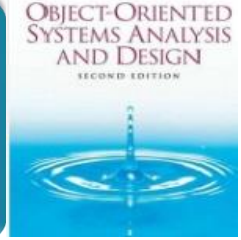


What Is Requirements discovery ?.

- During the process of development, the role of **requirements gathering** changes, **both in quantity and in nature**, but it does not entirely fade away until the information system is deployed, tested, and accepted by the customer.
- **Requirements discovery is** in the initial stages, this activity determines the requirements that the product must address: **the problems that it must solve, desires that it must satisfy, or opportunities of which it must take advantage.**
- **Requirements discovery** sets the boundaries of requirements gathering by defining its scope.
- **Without clear boundaries**, gathering requirements would be in danger of missing its destination:



Introduction

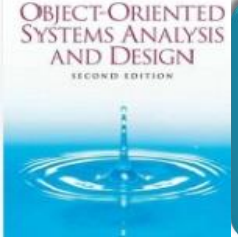


➢ What Is Requirements discovery ?.

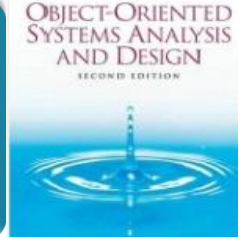
- ✓ It would end up with **either irrelevant or insufficient information**.
- ✓ **Requirements discovery** is **a finite activity** that is conducted on the surface of requirements, **unlike requirements gathering** that must go increasingly deeper as development goes forward.
- ✓ To arrive at the correct and relevant requirements, we must choose **the right sources and employ relevant elicitation techniques.**
- ✓ **Talent and experience** are, without a doubt, very important to the gathering requirements, but they must be sharpened by learning about techniques and by paying attention to detail.
- ✓ Finally, requirements are useless to the development of the information system if they **are not correctly documented or cannot be traced, identified, or verified**.

Requirements management

- The task of **requirements management** is to ensure that requirements are **well-organized, easy-to-locate, reliable, and consistent.**
- During the **later stages of development** the **role of gathering requirements diminishes, but it never disappears.**



Introduction



➢ Requirement Gathering VS Requirements discovery

1. **Requirements discovery** is a phase that takes place at the start of the develop ment process. **Requirements gathering**, on the other hand, is an ongoing pursuit that accompanies every analysis activity and, although its level usually drops as the system nears completion, it never disappears.
2. **Requirements discovery** is very wide but shallow as it tries to identify the scope of the system. **Requirements gathering** goes deep within the boundaries that requirements discovery has identified. In other words, requirements discovery locates **where we must "drill,"** while requirements gathering is the **act of drilling down.**

Introduction

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

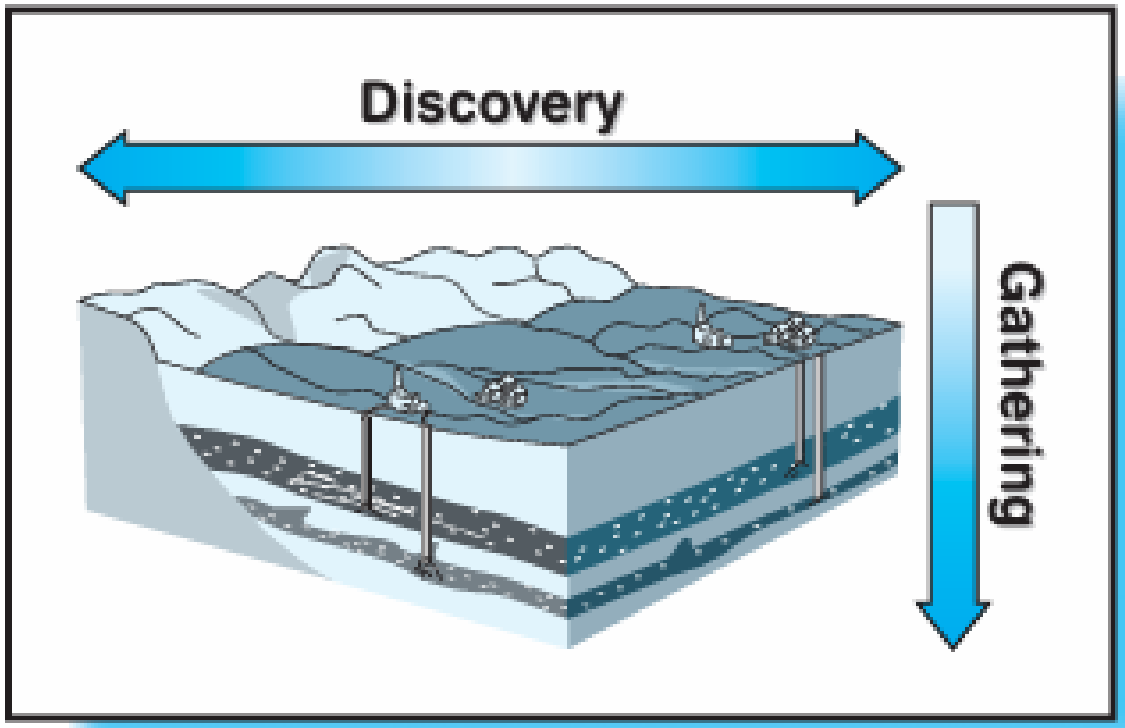
Requirement Gathering VS Requirements discovery

3. **Requirements discovery** aims to elicit the objectives of a vast group of stake holders and then verify them with business decision makers to separate "wish lists" from actual business goals. **Requirements gathering** acts on the belief that the goals are defined and we must identify what is needed to achieve the goals.
4. **Requirements discovery** is more chaotic than requirements gathering because we have yet to ascertain what we are seeking. Requirements gathering is **more ordered as we know what we are searching for.**

Introduction

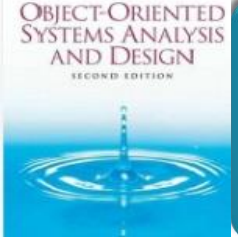
OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- **Requirement Gathering VS Requirements discovery**

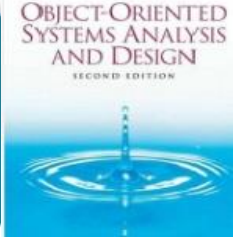


To hit "pay dirt," you must first "prospect" the field before drilling down Gathering requirements using an object-oriented perspective emphasizes on

1. **Objects,**
2. **Patterns,**
3. **Responsibilities, and**
4. **scenarios.**



Introduction



➢ Gathering requirements using an object-oriented perspective

1. Objects

- ❖ **An object** is **a person, place, or thing, such as student, faculty, sales clerk, city hall, famous park, ATM machine, and video tape**.

2. A pattern

- ❖ **A pattern** is **a template of objects with stereotypical responsibilities and interactions**; **the template may be applied again and again, by analogy**.
- ❖ **Pattern instances are building blocks used to assemble effective object models**.
- ❖ **For example**, a transaction object and transaction line item objects are a familiar pattern or template in business information systems. An actual instance of the transaction to transaction line item pattern is a sales order (transaction) with its associated sales order line items (transaction line item)

- **Requirement Gathering VS Requirements discovery**
- **Gathering requirements using an object-oriented perspective emphasizes**

3. Responsibilities

- **Responsibility** is associated with objects and has three aspects to it:

A. **What the object knows about itself**. The things that an object knows about it- elf are called attributes. An attribute is a characteristic associated with a person, lace, or thing object. Each characteristic has a value or state.

- For example, the following are attributes: person's name, person's telephone number, person's grade point average, place name, place location, vehicle name, and vehicle type. The following are values or states for the preceding attributes:

- **Requirement Gathering VS Requirements discovery**
- **Gathering requirements using an object-oriented perspective emphasizes**

3. Responsibilities

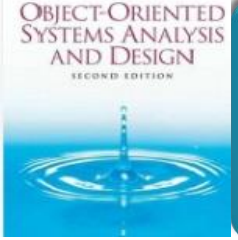
B. Who the object knows. A problem domain has many objects within it. Who the object knows identifies relationships between objects. A standard relationship is a connection between different types of objects, such as students and courses (relation- ship: students take courses; courses are taken by students), sales order and line item (relationship: sales orders have line items on them; line items are found on sales or- ders), and video tape and customer (relationship: video tapes are rented by customers; customers rent video tapes).

- **Requirement Gathering VS Requirements discovery**
- **Gathering requirements using an object-oriented perspective emphasizes**

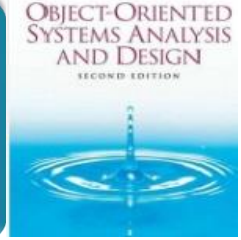
C. **What the object does.** This translates into a list of services for each object. A service is some functionality that an object is responsible for performing, such as registering for a course, dropping a course, checking out a video tape, purchasing products at the supermarket, and so on.

4. **Scenarios.**

- **A scenario** is a time-ordered sequence of object interactions to fulfill a specific responsibility. A scenario would be developed for each of the preceding services.



Requirement Classification

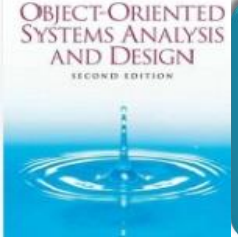


- Requirements can belong to two major categories**:**

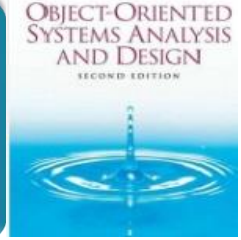
  A. **Functional (what the product does or behavioral)** and

  B. **Nonfunctional (what attributes the product has**).

- **Since both relate to the same product, they are interrelated and affect each other.**

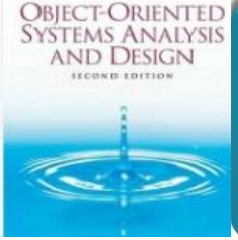


Requirement Classification

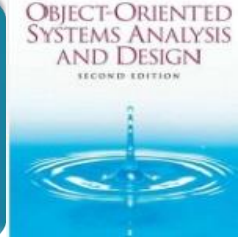


A. Functional Requirement

- **Functional requirements** specify the **behavior of the system** and the **constraints on that behavior.**
- The most top-level requirements express the **mission of the system**.
- An information system, however, must provide **specific services** that helps it users to achieve specific goals.
- The sum of these services is called **the behavior of the system**, specified by a set of **functional requirements.**

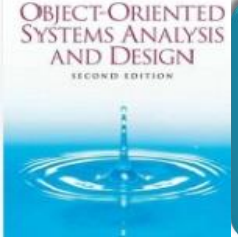


Requirement Classification

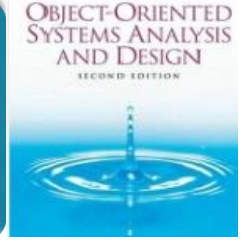


➢ Functional Requirement

- **A mission specifies a strategic goal; functional requirements** specify **how the mission is to be accomplished by achieving tactical goals**.
- To accomplish each tactical goal, **the system and its users** must interact in a set of clearly defined steps.
- For example, the mission of an ATM system is to allow customers to perform many banking transactions through automated teller machines.
  - The **mission** consists of several individual goals including, for example, **Get Cash, Deposit Check, Transfer Money, Get Account Balance, e**tc.



Requirement Classification



- **Classification of requirement**

  A. **Functional (or behavioral)**

- **For example,**

- In turn, each goal requires a set of interactions between the customer and the system. For instance, to deposit a check, the ATM and the customer must perform the following steps set of clearly defined steps.

  1. Customer swipes his/her banking card.
  2. System asks for password.
  3. Customer enters password

Requirement Classification

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- **Classification of requirement**

A. Functional (or behavioral)

4. If the password is correct, the system displays a menu of options from which a customer may pick one. **(If the password is wrong, the system requests the password again**. After **three failures**, the system blocks the account and asks the customer to contact the bank.)
5. **Customer selects the Deposit Check option.**
6. **System displays a list of customer's accounts and requests the customer to select one account for deposit. (If customer has only one account, this step is skipped.)**

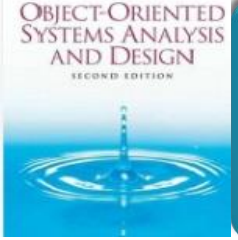
Requirement Classification

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

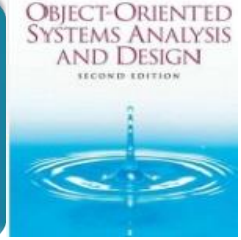
- **Classification of requirement**

A. Functional (or behavioral)

7. **System asks the customer to enter the amount of the check that is to be deposited.**
8. **System asks the customer to verify that the check and the deposit slip are in the deposit envelope and are ready for deposit.**
9. **After the customer's verification, the system turns on the rollers to reel in the deposit envelope into the deposit bin.**
10. **The system turns off the rollers after receiving the envelope.**
11. **The system asks the customer if a receipt is required.**



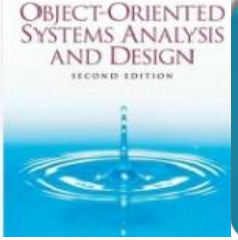
Requirement Classification



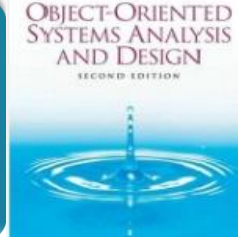
- **Classification of requirement**

  A. Functional (or behavioral)

- In these statements, **any action performed by the system is a functional requirement.**
- The **main device** for capturing functional requirements and modeling them is **the use case.**



Requirement Classification

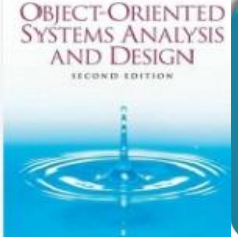


➢ Classification of requirement

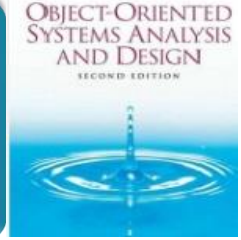
B. Nonfunctional

❖ There are two kinds of nonfunctional requirements:

- **Logical issues**: Business rules attached to a function. For example, during sales registration, a series of constraints could be considered, such as not closing the sale until the credit card operator confirms the payment, or not closing the sale if the last delivery to the same address has been returned due to invalid address.
- **Technological issues**: Constraints and qualities related to the technology used to perform the function, such as, for example, the user interface, the kind of communication protocol, security constraints, fault tolerance, and so on



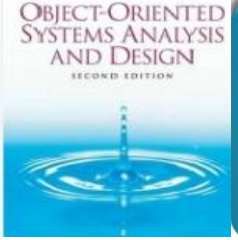
Requirement Classification



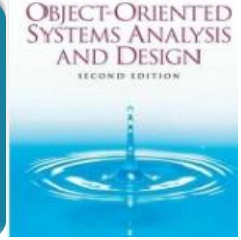
- **Classification of requirement**

B. Nonfunctional

- Constraints and qualities that are specifically attached to a function are called **nonfunctional requirements**, and general constraints and qualities are called **supplementary requirements**
- **Supplementary requirements** are all types of constraints and qualities related to the system as a whole and not only to individual functions.
- The document that contains the words, drawings, and pictures is often called the **user requirements specification document**. It becomes the blueprint for the information system waiting to be built or modified.



Requirement Classification



- **Classification of requirement**

  B. Nonfunctional

- **Nonfunctional requirements** specify **non-behavioral properties** of the system and **the constraints on those properties**.
- **Categories of nonfunctional requirements vary from product to product**. For a software system, the following categories are the most common:
  - **Usability**
  - **Reliability**
  - **Performance**
  - **Maintainability**
  - **Security**

Requirement Classification

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- **Classification of requirement**

  B. Nonfunctional

- **Usability:-** Usability defines how the behavior of the system must be shaped to fit the users and their work environment
- **Usability** is about the behavior of the system, but it is not the behavior itself.
- It specifies how the behavior of the system must be tailored to the level of its users' expertise, the environment in which a system operates, the volume of input and output, etc.
- **Example:** The interface must be designed for a touch-sensitive screen, not a keyboard

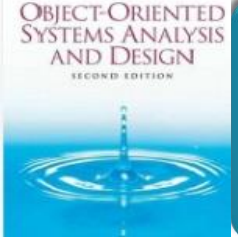
Requirement Classification

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN
SECOND EDITION

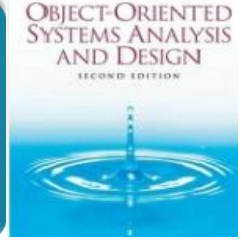
- **Classification of requirement**

  B. Nonfunctional

- **Reliability:-** requirements define the dependability of the system both in time and in the fulfillment of its functions.
- **Reliability requirements** generally relate to issues such as availability (in terms of time), mean time between failures (MTBF), and accuracy.
- **Performance:-**Performance requirements specify the response time and the input-output volume that the system can handle within a particular time frame.
- **when "good" performance is required**, the analyst must make sure that the requirement is defined in quantitative terms



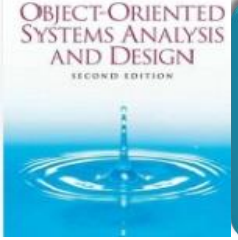
Requirement Classification



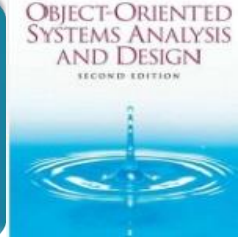
- **Classification of requirement**

  B. Nonfunctional

- **Maintainability:-** Maintainability requirements specify the ability of the software to be **modified and enhanced**
- **One architectural approach to better maintainability is to construct the software system from components with distinct responsibilities and relative independence (or loosely coupled components).**
- This approach is an important outgrowth of the object-oriented idea.



Requirement Classification



➢ Classification of requirement

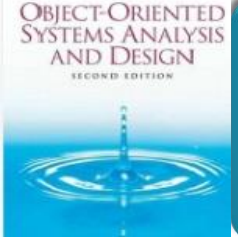
B. Nonfunctional

- **Security:-**Security requirements specify **the rights of access to the services of a system, the manner of access to those services, and the tracing of interaction with the system.**
- There is, however, a wide range of nonfunctional requirements that are related to security. They include:

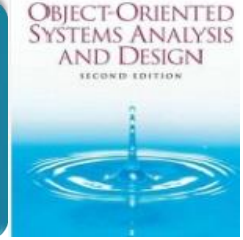
i. **Does data need to be encrypted?** We do not want to send or receive unencrypted data for a credit card transaction.

ii. **Do we need to log certain events?** If a patient's appointment with a hospital is cancelled, most likely management would want to know the reason. Did the patient cancel the appointment? Was the medical staff absent? Did lab equipment break down?

iii. **Do we need an audit trail?** While logs record an event, audit trails record changes to the value of one attribute or a set of attributes. If the price of merchandise in a supermarket is changed, we may want to see what the price was before it was changed and who changed it

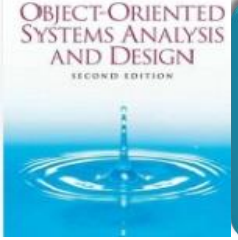


What Is Requirements Gathering?

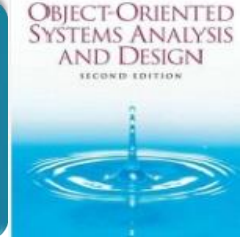


➢ What Is Requirements Gathering?

- **Requirements gathering** is a step in the **requirements management process**, which consists of:
  - **Gathering,**
  - **Documenting and**
  - **Analyzing project requirements.**
- Requirements gathering, or requirements elicitation, is **the process of determining all the requirements of a project.**

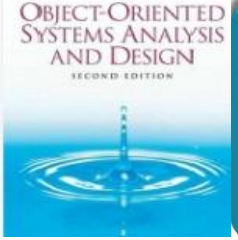


What Is Requirements Gathering?

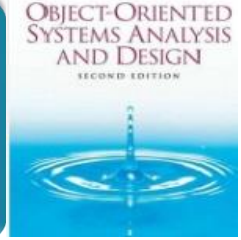


What Is Requirements Gathering?

- There are two main types of project requirements, **business and technical requirements.**
- **Business requirements** define **what an organization will accomplish** with the project, while **technical requirements** explain **how the project must be executed.**



What Is Requirements Gathering?



- Truly effective requirements gathering and management is started at the very beginning of the project, and **must answer the following questions**:
  1. **How long will the Project timeline be?**
  2. **Who will be involved in the project?**
  3. **What are the risks for the requirements gathering process?**
  4. **What is our ultimate goal in understanding our project requirements?**
  5. **What are our technical and business requirements?**

What Is Requirements Gathering?

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

Requirement Gathering vs Requirement Elicitation

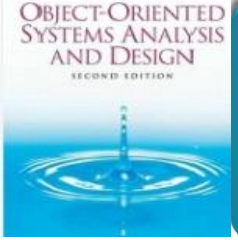
Aspect	Requirement Gathering	Requirement Elicitation
Definition	The process of **collecting** requirements, often passively or through available documents and inputs.	The process of **actively engaging stakeholders** to discover, uncover, and refine their needs.
Approach	More **passive**—assumes requirements already exist somewhere to be gathered.	More **active and interactive**—assumes requirements must be uncovered through discussion and analysis.
Methods Used	Document analysis, templates, forms	Interviews, workshops, brainstorming, prototyping, observations
Stakeholder Involvement	Minimal—may rely more on documentation	High—relies heavily on communication with stakeholders
Outcome	A list of collected requirements	A deeper understanding of **needs, goals, constraints**, and priorities
Focus	**What** the stakeholders say they want	**Why** stakeholders need it and **how** it fits the bigger picture

Requirements Gathering Techniques

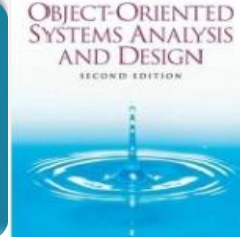
OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- **Types of Requirements**

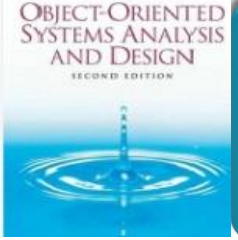
Type	Explanation
Functional requirements	Things that the system should do—may be represented by system use cases
Nonfunctional requirements	Constraints about how a functional requirement may work—may be represented by annotations to a system use case
Supplementary requirements	Constraints about the whole system, not necessarily a single requirement or use case—usually represented in



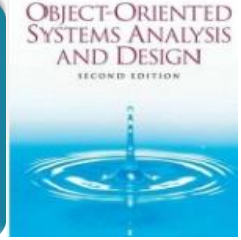
Requirements Gathering Techniques



- **Requirements** are not self-evident.
- They must be captured by **a set of tools and techniques** that have varying degrees of effectiveness and must be wielded differently depending on the situation.
- Some tools need more skills than others, but some skills in eliciting requirements can be sharpened by **learning and experience.**



Requirements Gathering Techniques



- They are techniques used to collect relevant requirements from relevant sources. They include **interviews**, **questionnaires**, **workshops, field trips** and **observation**, **document analysis**, and **modeling for elicitation.**

1. Interviews

- **Interviews** are the most flexible and direct tool for eliciting requirements. They are also more prone to misunderstanding and failure than any other technique.
- Early interviews in the requirements gathering process have a broad scope, while later ones are defined by more focus on detail.

2. Questionnaires

- **Questionnaires** are the second choice for requirements elicitation and the first choice for **requirements verification.**
- The structure of questionnaires as elicitation tools is generally the same as in interviews, but the flow is inflexible.
- As a verification tool, **questionnaires are the most traceable**
-

Requirements Gathering Techniques

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

3. Elicitation workshops

- **Elicitation workshops** are the most powerful but also the most expensive tool for requirements elicitation.
- Since they are very expensive, **you must select participants carefully and help them to help the workshop**

4. Field trips and observation

- provide valuable requirements where workflow is rich in action and interaction.
- But you **must be aware that the value of observing a workflow** also depends on the goal of the observation

Requirements Gathering Techniques

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

5. Models

- **Models** can be used to verify requirements.
- These models **need not be formal blueprints for the construction of the system,** but can be any drawing or word chart that the stakeholders can understand.
- **Mock-ups a**re one such modeling technique; **they are approximations of the system's user interface to elicit comments and requirements.**
- **Wireframes** provides a blueprint of the interface of a website or app by visualizing its navigation and layout.

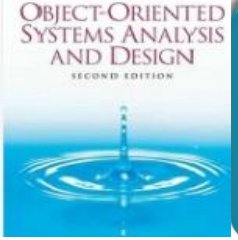
Requirements Gathering Techniques

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

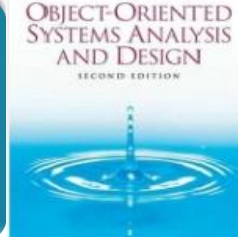
- It helps you understand how the app or website will work and identify if there are any errors in the design.
- Using a **Wireframe template** like the one below you and your team can understand how your system works.

6. Document Analysis

- **Reviewing the documentation** of an existing system can help when creating AS–IS process document, as well as **driving gap analysis** for scoping of migration projects.



Requirements Gathering Techniques



- In an ideal world, we would even be reviewing the requirements that drove creation of the existing system – a starting point for documenting current requirements.
- Nuggets of information are often buried in existing documents that help us ask questions as part of validating requirement completeness.

7. Focus Group

- **A focus group** is a gathering of people who are representative of the users or customers of a product to get feedback.
- The feedback can be gathered about n**eeds/opportunities/ problems to identify requirements, or can be gathered to validate and refine already elicited requirements**.

Requirements Gathering Techniques

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- This form of market research is **distinct from brainstorming in that it is a managed process with specific participants. Focus groups typically involve external stakeholders**

8. Use Cases and Scenarios

- Once you have **high-level functional requirements** in place, it is a good idea to explore a variety of use cases and scenarios.
- **Use cases** are the **specific, individual business objectives** the system must accomplish and the various situations in which **a given feature or functionality will be used.**

Requirements Gathering Techniques

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

8. Use Cases and Scenarios

- They describe the **various external entities that act on the system** and the **specific interactions they have with the system to accomplish the business objective**.
- **Use cases** are expressed **as step-by-step lists of tasks performed during a process.**
- **Scenarios**, also called **user stories**, are similar to use cases in that they describe **how the system will carry out a process to fulfill a business objective.**

8. Use Cases and Scenarios

- Their form, however, is **a narrative, rather than an enumerated list**.
- They are **essentially short stories** with the user in the role of the protagonist. Scenarios describe:
  - **Tasks users perform**
  - **Information users see**
  - **How users interact with the system**
- **Use cases and scenarios** can be used **to validate the features and functional requirements of the system across a wide range of situations**. They can also **help you discover exceptions and boundary cases that need consideration.**

Requirements Gathering Techniques

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

9. Role-play

- Some systems—certain kinds of enterprise software, like ERP, for example—must meet the needs of a variety of user types.
- **Role-play** can help to ensure that the needs of all users are being met.
- In a role-play session, **different people take the roles of the different user types.** Having the various roles interact with one another helps examine individual system requirements from different perspectives and generates discussions and new ideas.
- In effect, **role-play is an additional brainstorming technique**.
- It is a **good way to gain a solid understanding of how the various parts of the system need to function to support the overall process**

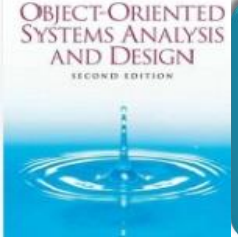
Requirements Gathering Techniques

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

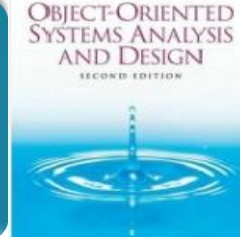
10. Brainstorming

- **Brainstorming** can be performed as part of a workshop .
- In your **brainstorming session,** consider different parts of the system individually.
- Explore various **what-if scenarios and blue-sky ideas**.
- The general idea is **to break away from existing conventions**.
- Consider visionary ideas **for pushing current boundaries**.
- Useful tools for brainstorming sessions include **whiteboards, mind mapping software, and empathy maps (the latter for exploring user needs).**

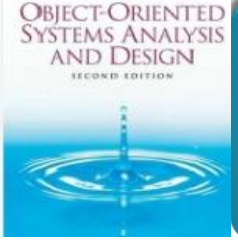
- **Assignment: Try to select suitable requirement gathering techniques for your Course project and Explain why you select the methods.**



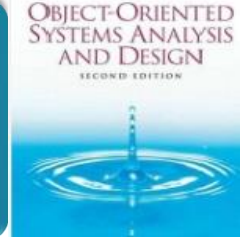
Requirements Gathering Techniques



- **Summary**
- **Requirements gathering** is the process of identifying your project's exact requirements from start to finish. This process occurs during the project initiation phase, but you'll continue to manage your project requirements throughout the project timeline. In this piece, we'll outline the requirements gathering process and explain how taking time to focus on requirements gathering can lead to successful project outcomes.

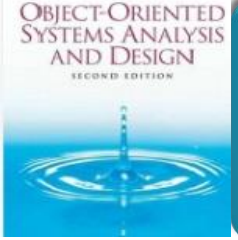


Requirements Gathering Techniques

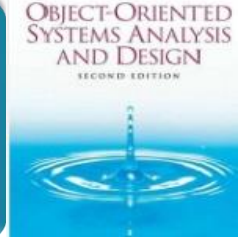


Fundamental Techniques for Requirements Elicitation

Technique	Description	Best For
1. Interviews	One-on-one or group sessions where stakeholders are asked open or structured questions.	Getting deep insights from key users or experts.
2. Questionnaires / Surveys	Written sets of questions distributed to stakeholders.	Gathering data from a large group quickly.
3. Workshops	Facilitated sessions where multiple stakeholders brainstorm and discuss requirements together.	Building consensus and prioritizing features.
4. Observation	Watching users interact with existing systems or processes to identify pain points or needs.	Discovering requirements that users may not explicitly mention.
5. Document Analysis	Reviewing existing documents, like user manuals, process docs, or system logs.	Understanding current systems and identifying gaps.
6. Prototyping	Creating mock-ups or simulations of the system to prompt user feedback.	Clarifying vague requirements and uncovering hidden needs.
7. Brainstorming	Generating a wide range of ideas from stakeholders in a short period.	Exploring solutions or uncovering innovative features.
8. Focus Groups	Discussions with selected groups of users to gather opinions, reactions, and expectations.	Understanding user expectations and emotional reactions.
9. Use Case / Scenario Analysis	Describing how users interact with a system in specific scenarios.	Identifying functional requirements and user workflows.
10. Interface Analysis	Reviewing how the new system will interact with users, other systems, or hardware.	Identifying inputs, outputs, and integration points.



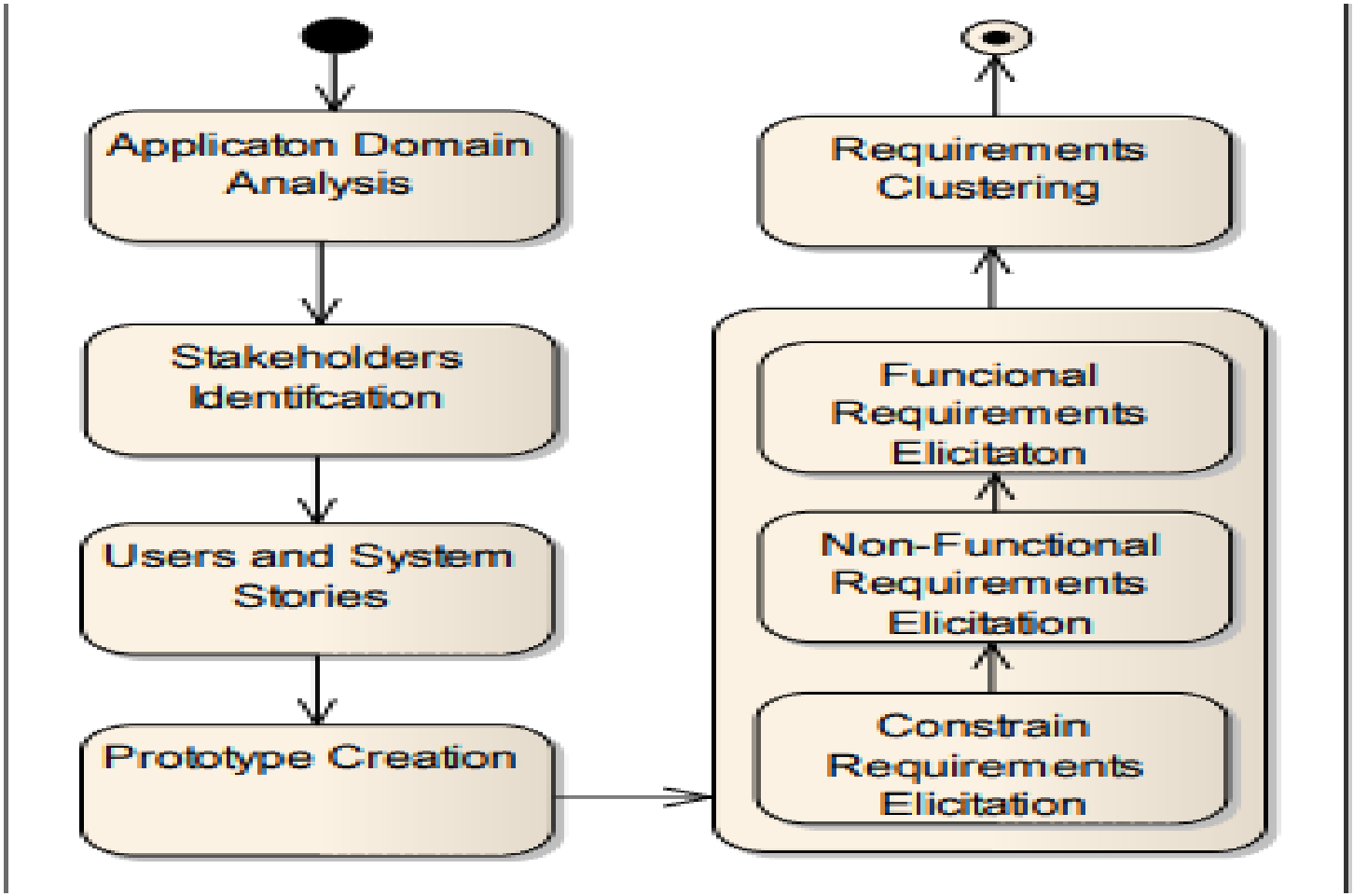
What Is Requirements Gathering?



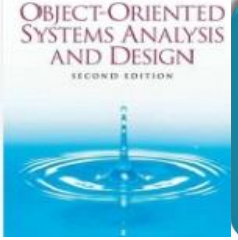
- The requirements elicitation process contains these steps:
  1. **Application Domain Analysis**
  2. **Stakeholders Identification**
  3. **Users and System Stories**
  4. **Prototype Creation**
  5. **Functional Requirements Elicitation**
  6. **Non-Functional Requirements Elicitation**
  7. **Constrain Requirements Elicitation**
  8. **Requirements Clustering**

- **Assignment: Read and try to remind the above process steps apply to the course project work**

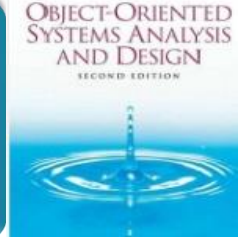
What Is Requirements Gathering?



Requirements Gathering Processes

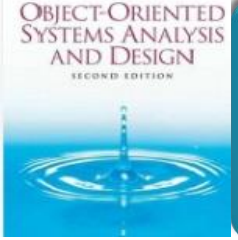


What Is Requirements Gathering?

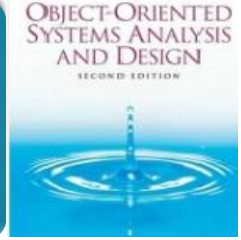


- **Managing requirements**
- **Managing requirements is as crucial to system development as is gathering requirements itself.**

1. **Document & update requirements.**
2. **Document sources.**
3. **Separate requirements into distinct units.**
4. **Uniquely identify each requirement.**
5. **Verify requirements & document verifications.**
6. **Prioritize requirements.**
7. **Classify requirements meaningfully.**



Requirements Validation



➢ Requirements Validation

❖ **Validation** denotes checking whether **inputs, performed activities, and created outputs (requirements artifacts) of the requirements engineering core activities fulfill defined quality criteria**.

❖ **Validation** is performed by involving relevant **stakeholders, other requirement sources (standards, laws, etc.) as well as external reviewers, if necessary**.

➢ Quality Criteria

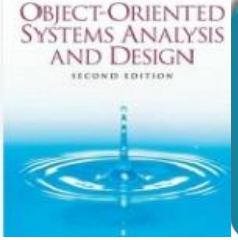
1. **Completeness** -The requirement must contain all relevant information (template).

Requirements Validation

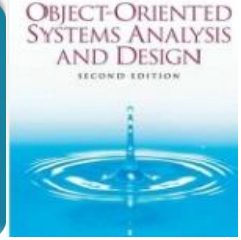
OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- **Requirements Validation**

2. **Consistency** - The requirements must be compatible with each other.
3. **Adequacy** - The requirements must address the actual needs of the system.
4. **Un ambiguity** -Every requirement must be described in a way that precludes different interpretations.
5. **Comprehensibility** - The requirements must be understandable by the stakeholders.
6. **Importance** -Each requirement must indicate how essential it is for the success of the project.



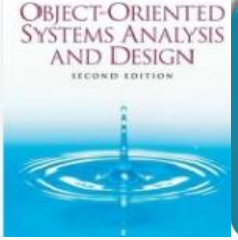
Requirements Validation



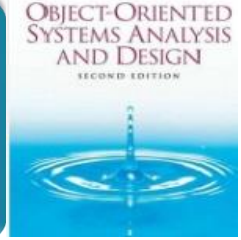
➢ **Requirements Validation**

7. **Measurability** - The requirement must be formulated at a level of precision that enables to evaluate its satisfaction.
8. **Necessity** - The requirements must all contribute to the satisfaction of the project goals.
9. **Viability** -All requirements can be implemented with the available technology, human resources and budget.
10. **Traceability** - The context in which a requirement was created should be easy to retrieve.

☞ **In System development the earlier an error discovered, the Cheaper it is to correct.**

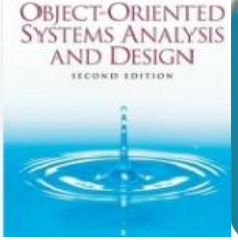


Requirements Validation

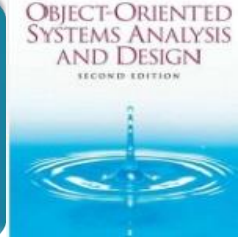


➢ The 6 Principles of Validation

1. **Involving the Right Stakeholders:-** **Ensure that relevant company-internal as well as relevant external stakeholders participate in validation**. Pay attention to the reviewers' independence and appoint external, independent stakeholders, if necessary.
2. **Defect Detection vs. Defect Correction:-** Separate defect detection from the correction of the detected defects.
3. **Leveraging Multiple Independent Views:-** Whenever possible, try to obtain independent views that can be integrated during requirements validation in order to detect defects more reliably.
4. **Use of Appropriate Documentation Formats:-** Consider changing the documentation format of the requirements into a format that matches the validation goal and the preference s of the stakeholders who actually perform the validation.

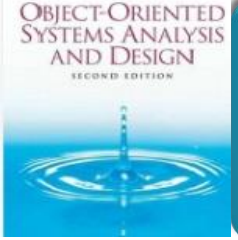


Requirements Validation

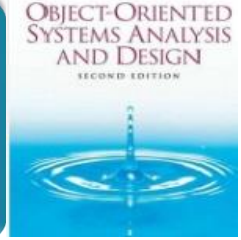


- **The 6 Principles of Validation**

5. **Creation of Development Artifacts during Validation:-** If your validation approach generates poor results, try to support defect detection by creating development artifacts such as architectural artifacts, test artifacts, user manuals, or goals and scenarios during validation.
6. **Repeated Validation:-** Establish guidelines that clearly determine when or under what conditions an already released requirements artifact has to b e validated aga



Requirements Validation



➢ Validation Techniques

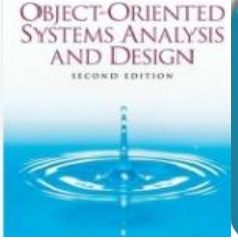
1. **Inspections:-** an organized examination process of the requirements
2. **Desk-Checks:-**
   - ❑ The author of a requirement artifact distributes the artifact to a set of stakeholders.
   - ❑ The stakeholders check the artifact individually.
   - ❑ The stakeholders report the identified defects to the author.
   - ❑ The collected issues are discussed in a group session (optional)

Requirements Validation

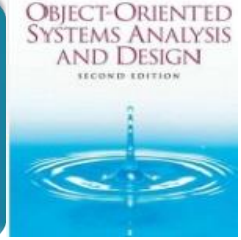
OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

Validation Techniques

3. **Walkthroughs:-** A walkthrough does not have formally defined procedure and does not require a differentiated role assignment. -Checking early whether an idea is feasible or not.
4. **Prototypes:-** A prototype allows the stakeholders to try out the requirements for the system and experience them thereby.
   - Develop the prototype (tool support).
   - Training of the stakeholders.
   - Observation of prototype usage.
   - Collect issues.



Requirements Validation

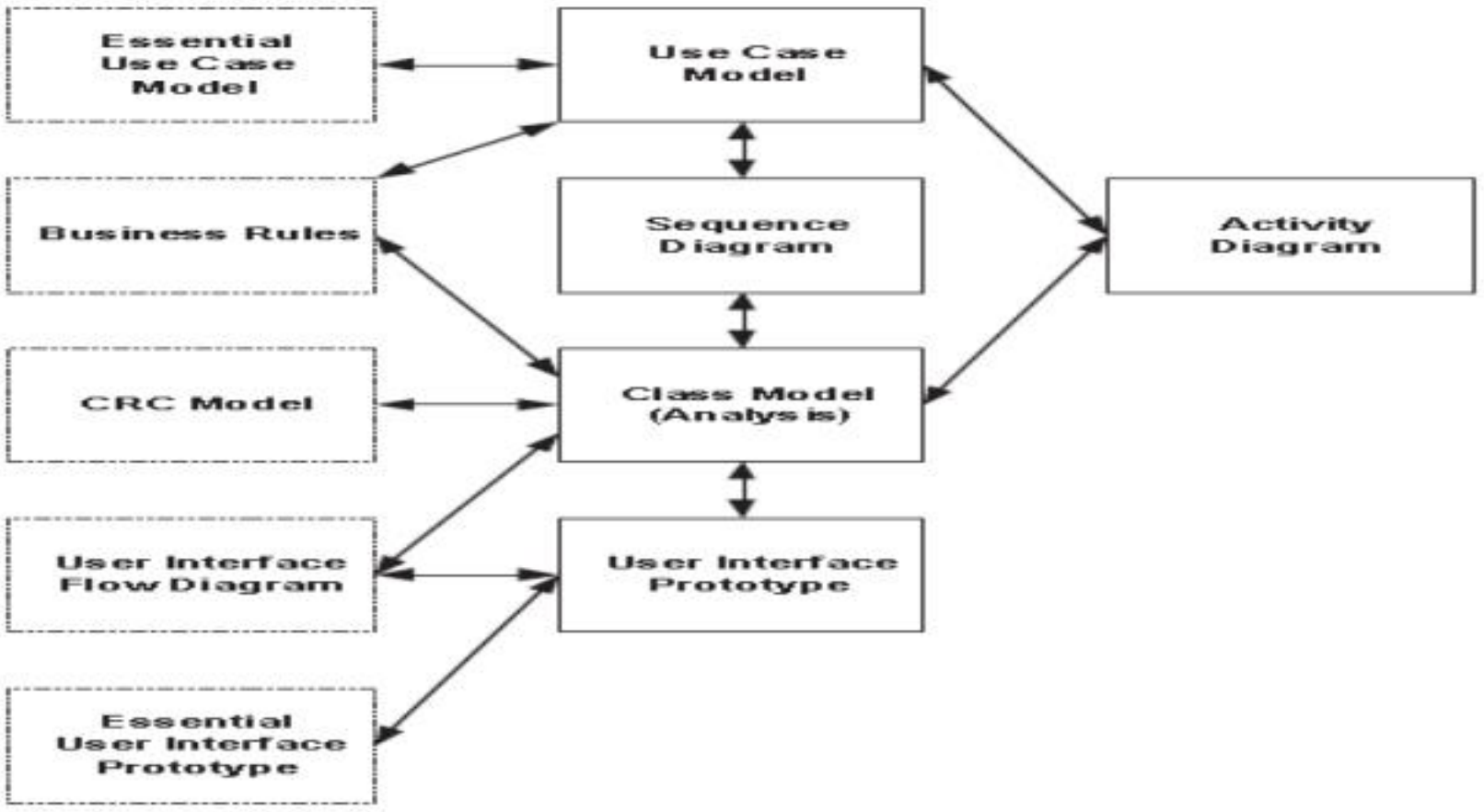


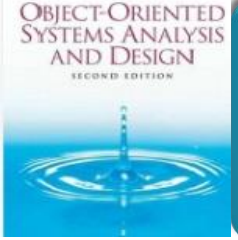
- **The challenges regarding requirements are, at least, the following:**

1. How to **discover** requirements.
2. How to **communicate** requirements to the other phases and teams of the project.
3. How to **recall** requirements during development to **verify** if they have all been implemented.
4. How to **manage** requirements change.

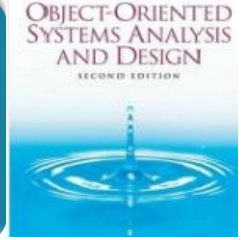
Essential Use Case Modeling

- Overview of Analysis artefacts and their Relationships





Essential Use Case Modeling



Topics

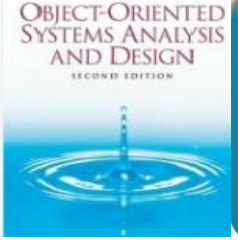
- What use case modeling is and is not?
- The four components of a use case.
- The basic elements of use case diagram
- Various flows in the narrative of a use case.
- How to transform concepts from domain analysis into use cases?
- Identifying prominent actors.
- Identifying major use cases.
- The context diagram

Essential Use Case Modeling

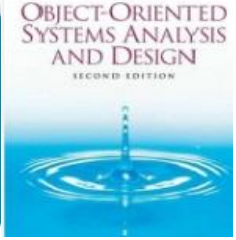
OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

- **Use case diagrams** help visualize the interaction between **the user and the system**, or in other words, **the user actions and the system responses**. It helps keep the focus on the requirements of the end user throughout the development of the system.
- **Use case diagrams d**escribes the **behavior of a system** from a user's standpoint
  - Functional description of a system and its major processes
  - Provides a graphic description of who will use a system and what kinds of interactions to expect within that system
  - Processes that occur within the application area are called use cases
  - Entities outside the area that are going to use the application are called actor



Essential Use Case Modeling



➢ What is Use Case Modeling?

- **A use case** is a compound entity. Understanding it requires a good understanding of its components and how they affect each other.
- It also requires a clear understanding of what use case modeling is not. Without this negative definition, use cases are liable to miss their mission and turn into a grab bag of irrelevant and misguiding details.
- **Use cases model** the behavior of a system. A use case is a unit of system behavior.
- **A use case details** the interaction of entities outside a system, called **actors**, with the system to achieve a specific goal by following a set of steps called **a scenario.**

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

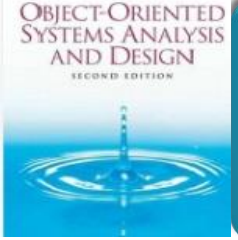
- **A use case** is a document that explains how users will perform tasks on your product. It is written from a user's point of view and done in steps that include:
  - **Who's using the product,**
  - **What they want from the product, the user's goal, the steps they take to accomplish their task and**
  - **How the product responds to their action.**
- **The significance of use case modeling** is that **it provides the framework for the most important building blocks of modeling and beyond**.
- **Use cases** are the indispensable guideposts from one end of system development to the other: from gathering requirements and communicating upstream with stakeholders to exchanging information downstream with designers and programmers to testing the product and training the users.

Essential Use Case Modeling

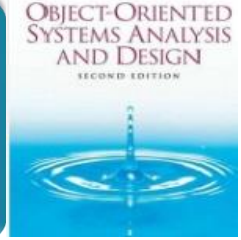
OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

What is Use Case Modeling?

- The significance of **use case modeling** is that it provides the framework for the most important building blocks of modeling and beyond.
- **Use cases** are the indispensable guideposts from one end of system development to the other:
  - **From gathering requirements and communicating upstream with stakeholders to**
  - **exchanging information downstream with designers and programmers to testing the product and training the users.**



Essential Use Case Modeling



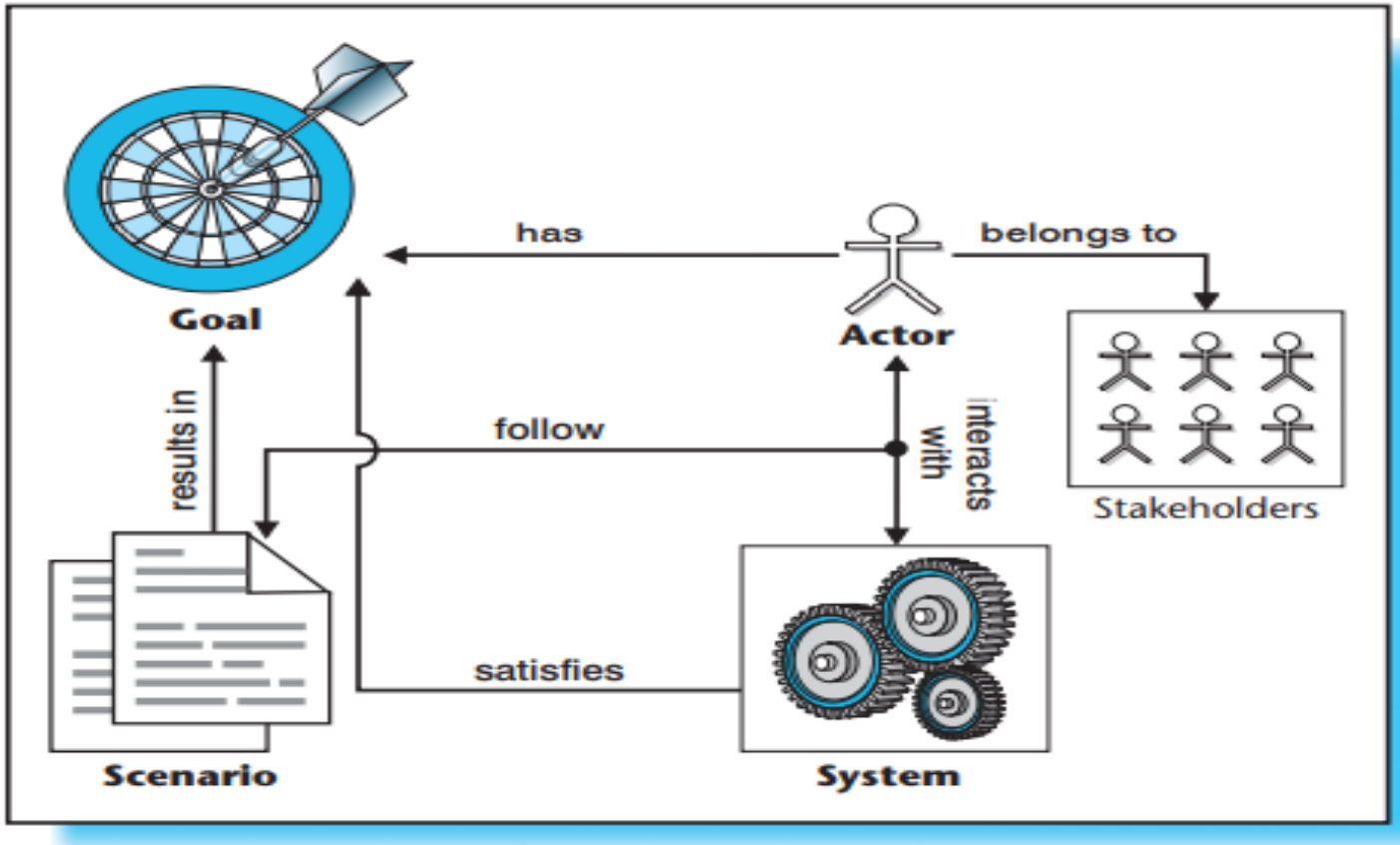
➢ What is Use Case Modeling?

- **Use case modeling** extends beyond use cases themselves.
- It is not limited to the components of the use cases but also originates a diverse set of documents that **shapes both the means and the goal of system development.**

What Use Case Modeling Is Not?

- Use case modeling is limited to a system's external behavior
  - **Use cases do not model the system from inside.**
  - **Use cases are not effective in capturing the nonfunctional requirements.**
  - **Use case modeling is not the same as functional decomposition.**
  - **Use cases are not inherently object-oriented**
  - **Use cases describe what a system accomplishes, not how.**

Essential Use Case Modeling



Components of a Use Case: **Actor(s), System, Goal, Scenario.**
The goal must result in measurable value, but the value is a matter of judgment

The four components of a use case.

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

❖ **The four components of a use case.**

❖ **A use case has four components:**

1. **A goal**

- **A goal:-** A goal, the outcome of a successful use case. This goal
  - **must be meaningful,**
  - **must be a logically complete function, and**
  - **must be of measurable value**
- A use case is **successful only if its stated goal is completely achieved.**
- Consider the use case, Purchase Groceries, that we previously described. If the customer does not select any item to buy and leaves the store, nothing useful is accomplished.

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

- Consider another example:
- If a **bank customer inserts a bank card** into an ATM and then **retrieves the card and simply walks away**, **the goal of the use case—call it Conduct ATM Transaction—is logically incomplete.**
- A goal is also shaped to a large degree by judgment.
- Suppose the **supermarket customer picks up items and then leaves the supermarket without paying for them.**
- It is undoubtedly of some "**measurable value**" to the customer, but the supermarket management **would not be amused**.

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

- Consider another example:
- **Without payment for the groceries, the use case is not successful.**
- Reaching the goal can be simple or complex but the **name of the use case**, which declares the goal, must be **clear, active, and simple.**
- **A use case's name is its goal**. The name must be **active, concise, and decisive**.
- **A use case's name must show action.**
- It must have **one transitive verb, simple or compound**, and **one grammatical object, simple or compound.**
- **It is the goal that decides the relevance of activities in a use case.**

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

❖ **The four components of a use case.**

2. Stakeholders and Actors

- **Stakeholders:-** whose interests are affected by the outcome of the use case, including actors who interact with the system to accomplish the goal.
- **Stakeholders** are those entities whose interests are affected by the success or the failure of the use case.
- A use case needs at least one stakeholder
- **An actor** is an **entity outside the system that interacts with the system to achieve a specific goal.**

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

❖ **The four components of a use case.**

2. Stakeholders and Actors

- **Stakeholders:-** whose interests are affected by the outcome of the use case, including actors who interact with the system to accomplish the goal.
- **Stakeholders** are those entities whose interests are affected by the success or the failure of the use case.
- **A use case needs at least one stakeholder**
- **An actor** is an **entity outside the system that interacts with the system to achieve a specific goal.**

➢ What is Use Case Modeling?

- **The four components of a use case.**

2. Stakeholders and Actors

- Among the actors identified in a use case, one, and only one, is **the primary actor.**
- **For example**: In the **Purchase Groceries** use case, the **primary actor** is the **customer who wants to buy groceries**.
- **The services** that the system provides in a use case **are primarily for the benefit of the primary actor, although not exclusively.**
- The goal of the primary actor is specified by the name of the use case

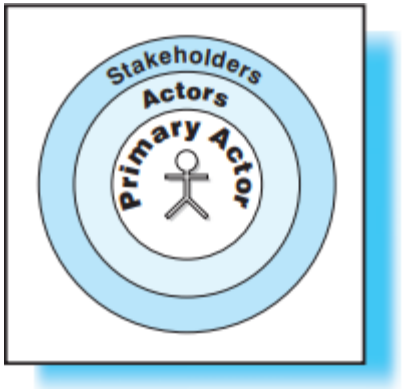
Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ **The four components of a use case.**

2. Stakeholders and Actors

- **An actor** is in fact not a specific person or group of persons, but **a role**.
- **The same person may appear in different roles**. That is to say, **a person may play one role in one use case and a different role in another use case.**
- Even in the **same use case, the same person can play multiple roles**. In Purchase Groceries, the cashier can also play the customer.



Circle of Stakeholders
Actors are stakeholders who interact with the system. The primary actor is an actor whose goal is accomplished by the use case

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

- **The four components of a use case.**

2. Stakeholders and Actors

- Other actors are **supporting (or secondary) actors**.
- **They support the primary actor in reaching** the goal of the use case. **In primary or supporting guises, the same actor may feature in multiple use cases.**
- **An actor** is any entity that interacts with the system. It can be **a person; it can also be another system or subsystem, a device, an organization, or even time.**

➢ What is Use Case Modeling?

❖ **The four components of a use case.**

2. Stakeholders and Actors

❖ Example:

❖ In the **Purchase Groceries use case**, identification of the primary actor is simple: it is the **customer**.

❖ It is the customer whose **goal is to buy groceries**; other elements in the story provide a service **to facilitate the customer including (for the moment) the lone supporting actor, the cashier.**

❖ **An actor** is a **role** that any user who has been given the part can play.

❖ A use case must enforce the interests of all stakeholders.

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

❖ **The four components of a use case.**

2. Stakeholders and Actors

❖ The name must be **unique across the whole enterprise, across the whole system, and not just within the scope of one use case, a set of use cases, or one domain or subsystem.**

❖ The commercial division of a bank may believe that a Customer is **a corporation only,** while the consumer division may consider only **an individual** as such.

❖ Both are correct within their own spheres of activity. However, an enterprise-wide information system for the bank cannot stand by double meanings.

❖ One actor must become Commercial Customer, the other Individual Customer.

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ What is Use Case Modeling?

- **The four components of a use case.**

3. A system

- **A system:-** provides the services that the actors need to reach the objective.
- **The system defines the boundaries of a use case.**
- **The scope of the system constrains the scope of the use case.**
- This constraint has a more profound effect than appears at first glance.
- In order to have **the correct actors and the right scenario**, we must **identify the correct system**.
- Precise as the goal of a use case is, **more than one system can satisfy the same goal**.

Essential Use Case Modeling

- **The four components of a use case.**

3. **A system**

❖ **Example:** Consider the following scenario for an information system that we would call **Checkout Groceries or Purchase Groceries:** One Supermarket, Two Systems:

The Real System

Purchase Groceries

1. Customer enters the supermarket.
2. Customer picks a shopping cart.
3. Customer selects groceries.
4. Customer brings groceries to cash register.
5. Customer pays for groceries.
6. Customer leaves with groceries.

The Information System

Checkout Groceries

1. Customer deposits groceries on checkout counter.
2. Cashier scans items one by one.
3. Cashier calculates the total.
4. Cashier informs customer of total.
5. Cashier receives payment from customer.
6. Cashier gives receipt to customer.

The two systems come into contact (see blue arrows), but the use cases are not the same

➢ What is Use Case Modeling?

- **The four components of a use case.**

4. **A scenario**

- **A scenario:-** is the activity that the actors and the system follow to accomplish the goal. Constructing a scenario is the bulk of the work in use case modeling.
- The general outline of the scenario can be discovered by analyzing the definition of the **concepts classified as "process" or "function."**
- With the system **to achieve a specific goal by following a set of steps** called **a scenario.**

➢ Developing Use Cases(Behavioral Modeling)

- **State is** "**the cumulative results of the behavior of an object**;
- one of the possible conditions in which **an object may exist, characterized by definite quantities that are distinct from other quantities."**
- In other words, **an object's state changes if the value of at least one of its attributes changes:**
- **For example**: a 70-year-old man is the same person, the same object, as when he was 7 years old, but is now in a different state.

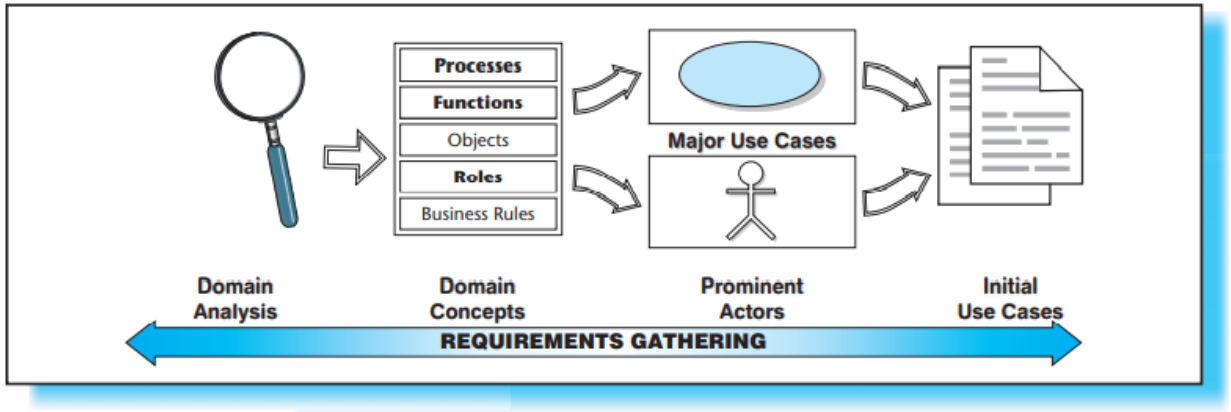
➢ Developing Use Cases(Behavioral Modeling)

- **Trigger**
- **A trigger** is the event that starts the use case. **The trigger** occurs, or can occur, only if the preconditions to the use case are met.
- **A use case may have more than one trigger**. In Make Appointment, the use case does not start unless the referral source calls the hospital.
- **Make Appointment** is a **Trigger**: **that is the referral source calls the hospital for an appointment.**

Essential Use Case Modeling

➢ Developing Use Cases(Behavioral Modeling)

❖ By defining the behavior that the users expect from the information system, use cases form the foundation of conceptual modeling.



Precondition and Post-Condition

- **Precondition** defines the state of the system before a use case can start;
- **post-condition** defines the state of the system after a use case is complete

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- Developing Use Cases(Behavioral Modeling)

- **Example:**

- **In Register Patient**, the reception clerk is an **entity** inside the hospital system who does not directly participate in registration.

Register Patient	**Precondition:**	The reception clerk has verified that the patient has an appointment but must register.

Register Patient	**Post-Condition:**	The patient is registered and is provided with a hospital ID card.

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN
SECOND EDITION

- Developing Use Cases(Behavioral Modeling)

- **Example:**
- **In Verify Insurance Plan**

Verify Insurance Plan	**Precondition:**	Hospital needs valid health insurance plan for the patient.

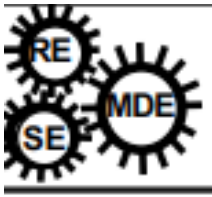
- **Describe the Post condition of the use case above**

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- Developing Use Cases(Behavioral Modeling)

- **Example 1:**



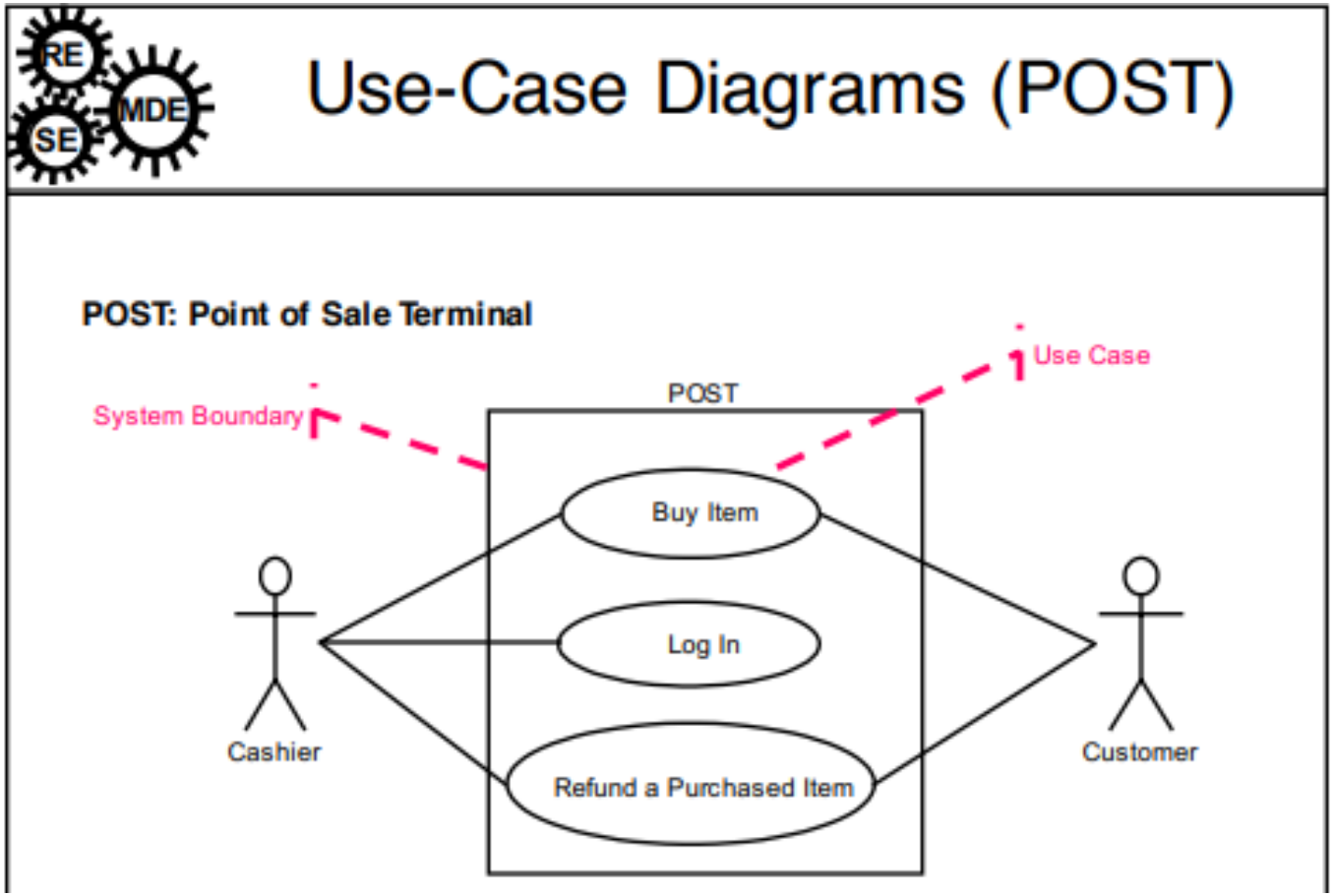
Point of Sale Terminal (POST)

- Computerized system to record sales and handle payments
- Typically used in a retail store
- Includes HW components, such as computer and bar code scanner
- Software to run the system
- Goals of system:
  - Increase checkout automation
  - fast and accurate sales analysis
  - automatic inventory control

Essential Use Case Modeling

- Developing Use Cases(Behavioral Modeling)

- **Example 1:**



Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- Developing Use Cases(Behavioral Modeling)

- **Example 2:** Patient Management system

- **Actor Dictionary**

Actor	Description	Abstract	Use Case(s)
Appointment Clerk	Makes appointment for the patient to receive medical service.		Make Appointment
Billing Clerk	Maintains patient billing.		Enter Bulk Payment
Hospital Clerk	Generalizes: • Appointment Clerk • Billing Clerk • Reception Clerk • Registration Clerk	✔	Resolve Patient Billing Issue
Reception Clerk	Receives patient on arrival at the hospital. Verifies registration. Arranges for the patient to receive medical service.		Receive Patient
Registration Clerk	Enters or updates patient's personal and payment data. Issues a hospital card, if necessary.		Register Patient

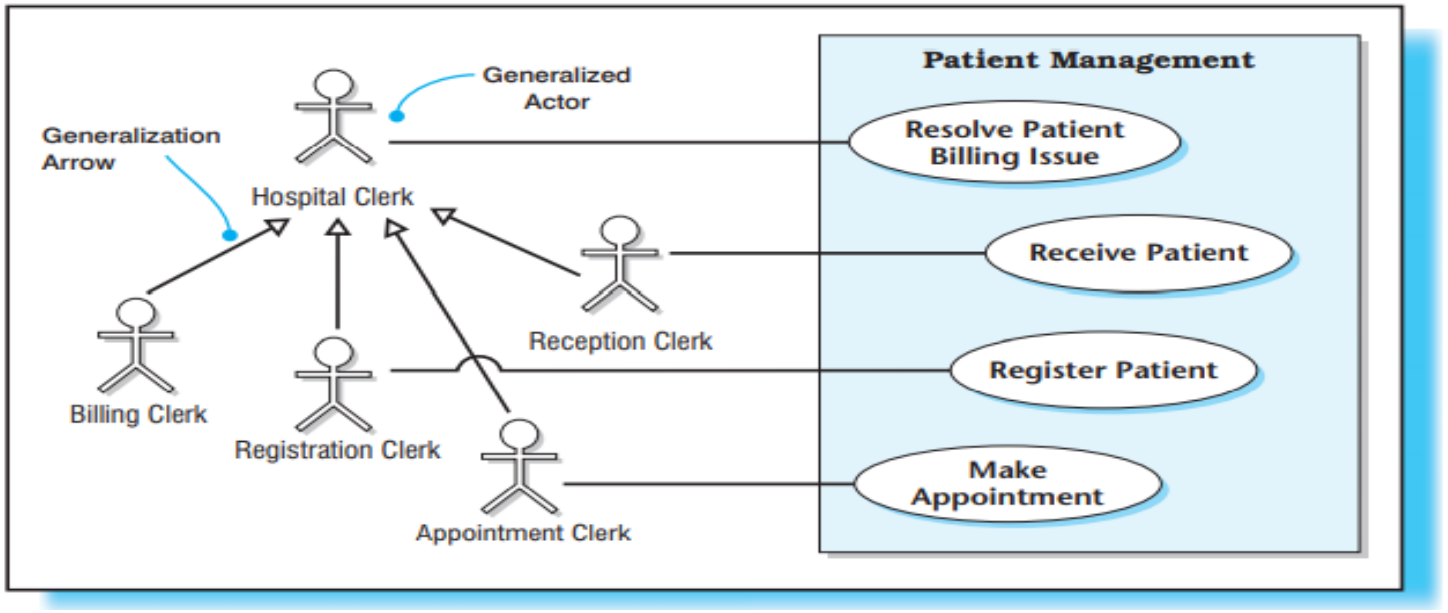
Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- Developing Use Cases(Behavioral Modeling)

- **Example 2:** Patient Management system

- **Actor Generalization: Creating a "Super-Role"**



- Billing Clerk has no other role in Patient Management, but would be **a busy actor in the Accounting subsystem**

- DEPENDENCIES: INCLUDE AND EXTEND

- Developing Use Cases(Behavioral Modeling)

Dependencies: Include and Extend

- The relationships that express and model this dependency are two: **include and extend.**
  1. **Extend relationship:-** is one in which a use case is **created to extend the functionality of a base use case. For example**
  2. **Include Relationship:-** is one in which **one use case uses the functionality of another, independent, use case**

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- Developing Use Cases(Behavioral Modeling)

- **Dependencies: Include and Extend**

- **Example:**

Register Patient	
Alternate Flow/ Exceptions:	. . .
	2.a Patient is not new and insurance data has not changed. Registration clerk does not update the insurance data by default.
	2.b Patient wants to pay the entire bill or the co-payments by a credit card. Registration clerk verifies the credit card (Extend: 142 – Verify Credit Card) and records credit card information.

- **Register Patient** needs to be **enhanced to handle credit cards**. We conclude that the **normal scenario of the use case is just fine and need not to be changed**. We only need **to extend it through its alternate flow by delegating the functionality to another use case:**

Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

- Developing Use Cases(Behavioral Modeling)

- **Dependencies: Include and Extend**

- **Example:**

Receive Patient	
Normal Flow:	. . .
	3. Reception clerk verifies that patient has been registered and registration is valid.
	. . .
Alternate Flow/ Exceptions:	**3.a** Patient is new. Reception clerk directs the patient to registration. (Include: 140 – Register Patient.)
	. . .

- **In Receive Patient**, the reception clerk must ensure that the system has the right state about the patient before **dispatching the patient to the medical service, but after verifying that the patient has an appointment.**

- Developing Use Cases(Behavioral Modeling)

Dependencies: Include and Extend

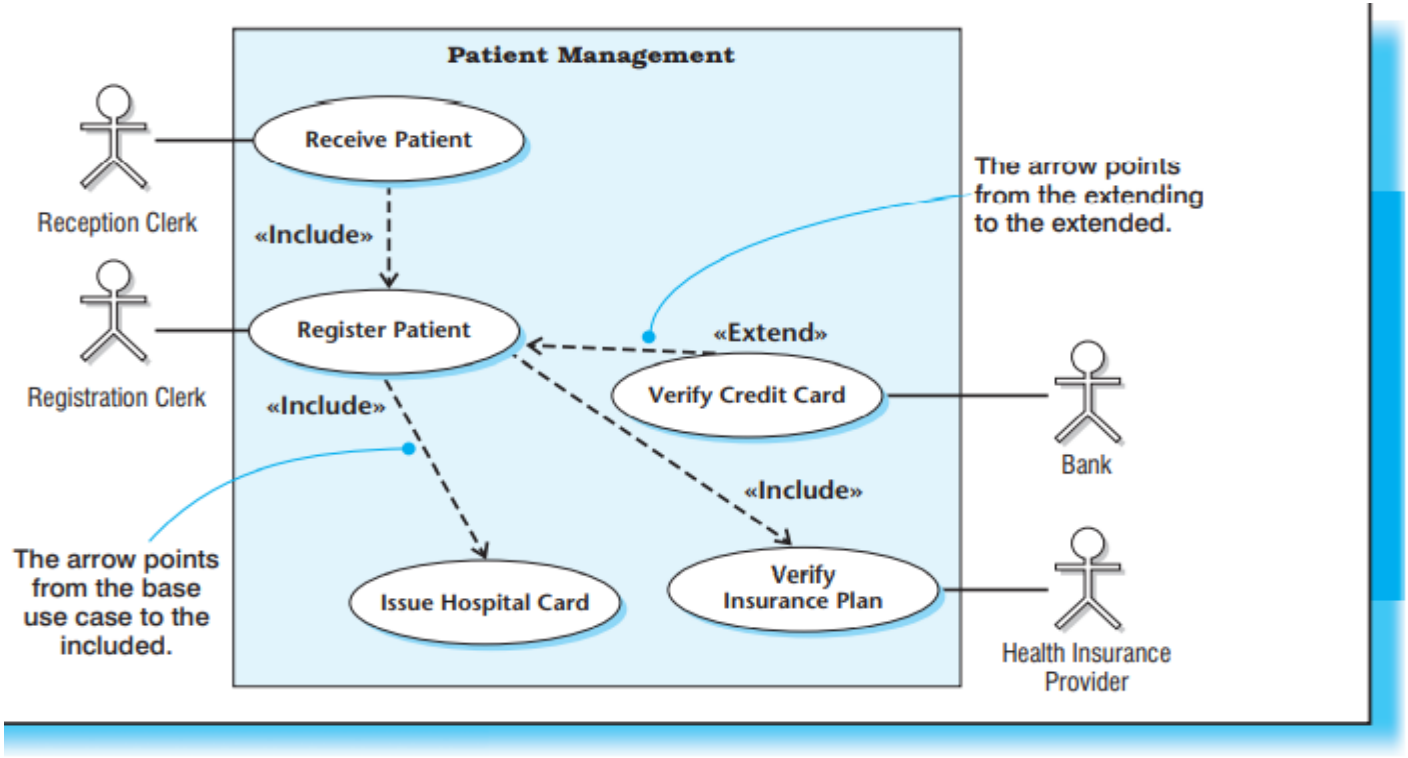
- Here**, in include relationship unlike the extend relationship**, it is the "**base" use case** that is dependent on the use case that it calls.
- In a use case diagram, dependency type is **indicated by the direction of an arrow.**
- **The base use case** is **independent of an extending use case but dependent on an including one.**

Essential Use Case Modeling

- Developing Use Cases(Behavioral Modeling)

- **Dependencies: Include and Extend**

- **Example:**



Essential Use Case Modeling

- Developing Use Cases(Behavioral Modeling)

- **Dependencies: Include and Extend**

- **Example:**

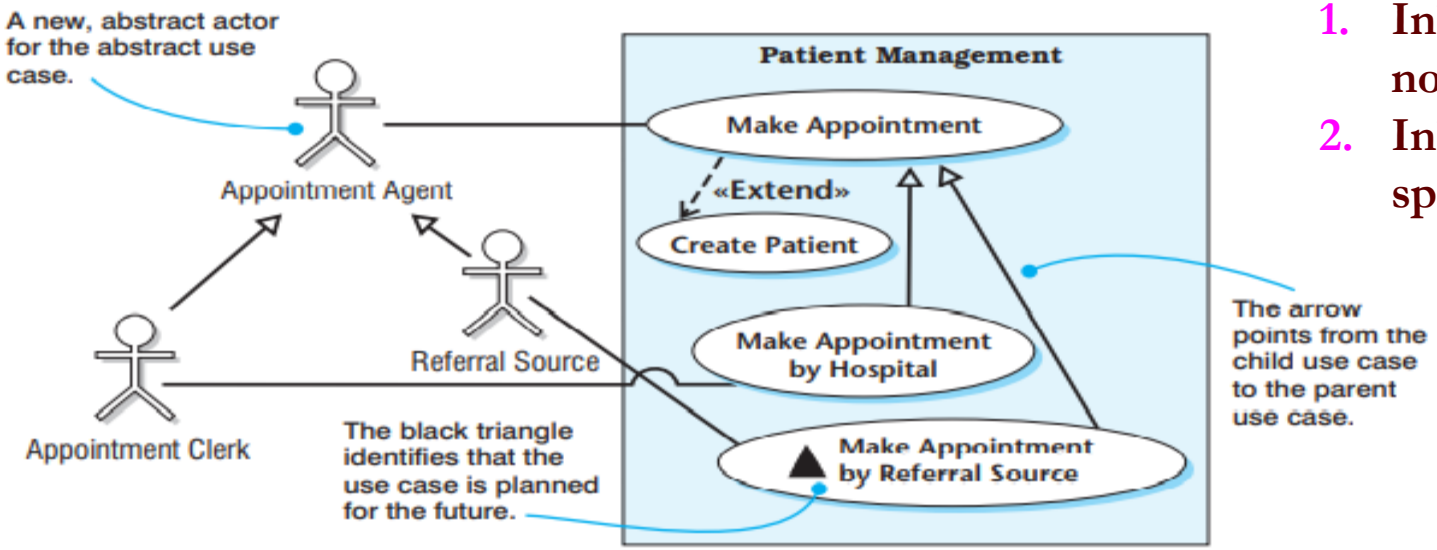
Base UC	Arrow's Direction	Referenced UC
Extended UC	←	**Extending UC**
Register Patient		Verify Credit Card
Including UC	→	**Included UC**
Receive Patient		Register Patient

Essential Use Case Modeling

- Developing Use Cases(Behavioral Modeling)

- **Dependencies: Include and Extend**

- **Example:** Use Case Generalization: When the Same Goal Is Achieved by Different Means



Read about

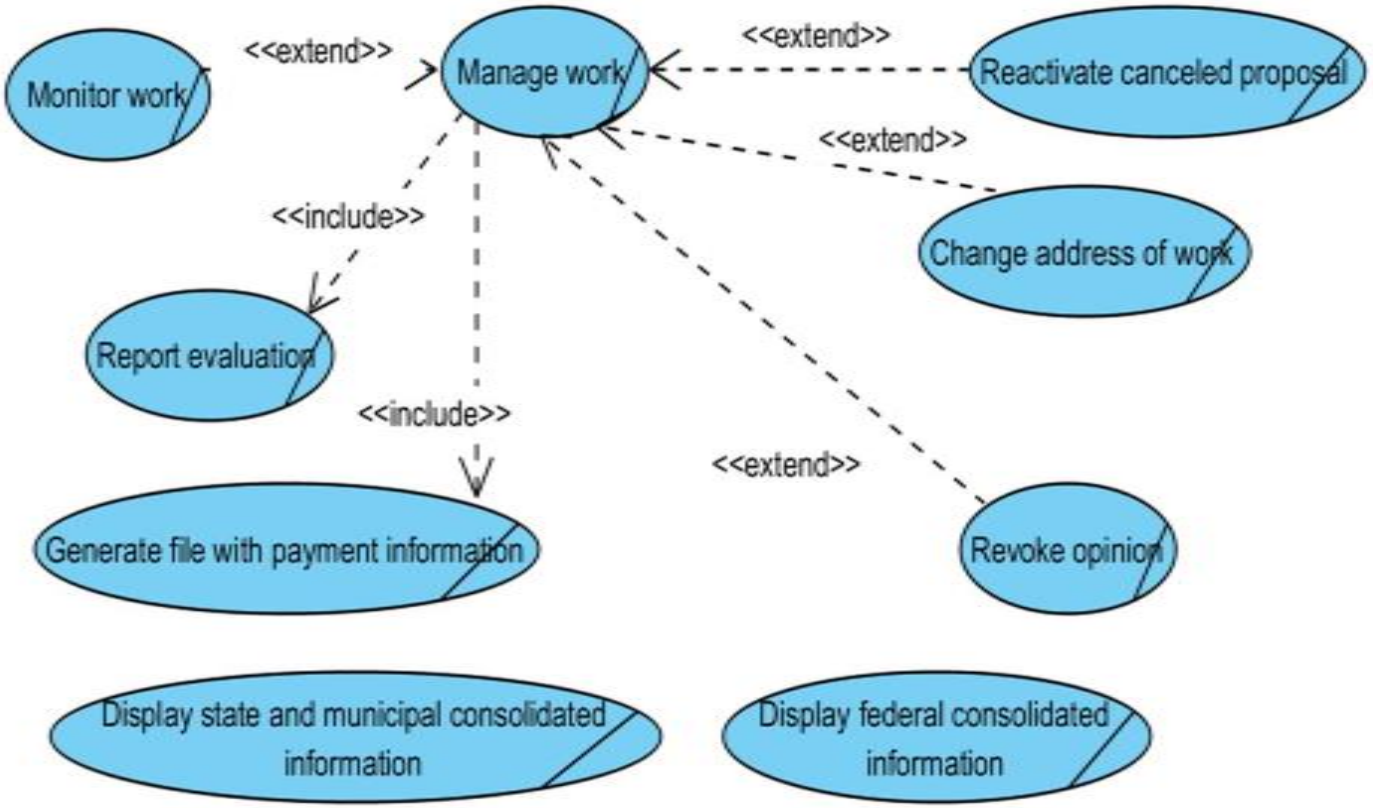
1. **Inherited with no change.**
2. **Inherited but specialized.**

Essential Use Case Modeling

- Developing Use Cases(Behavioral Modeling)

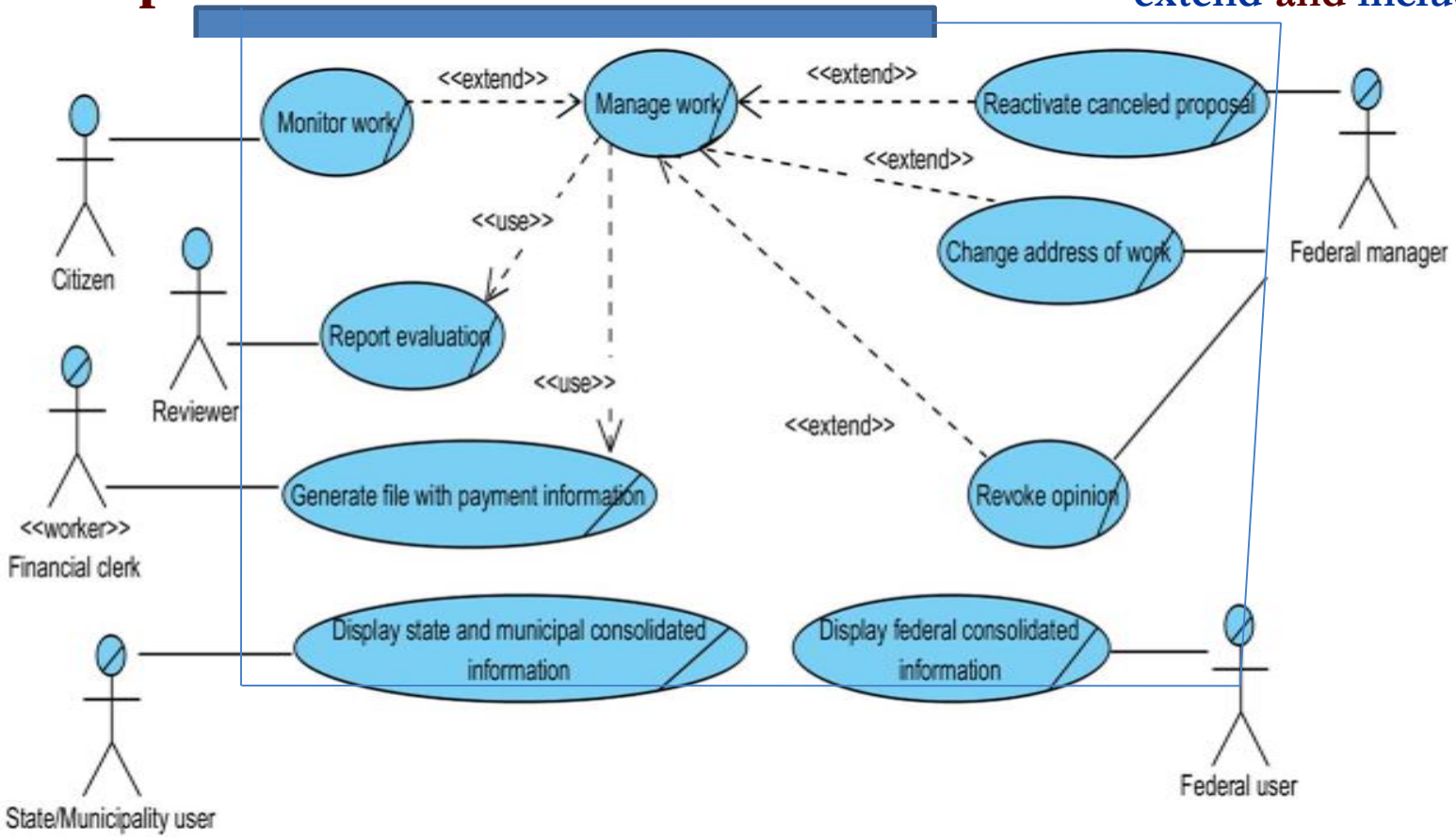
- **Dependencies: Include and Extend**

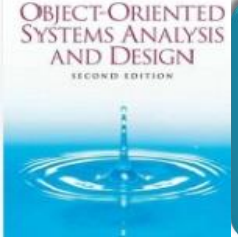
  - **Examples of the use of extend and include .**



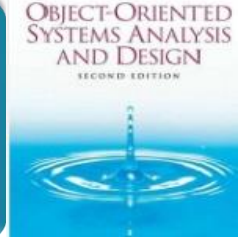
Essential Use Case Modeling

- Developing Use Cases(Behavioral Modeling)
- **Dependencies: Include and Extend**
  - **Examples of the use of extend and include .**





Essential Use Case Modeling



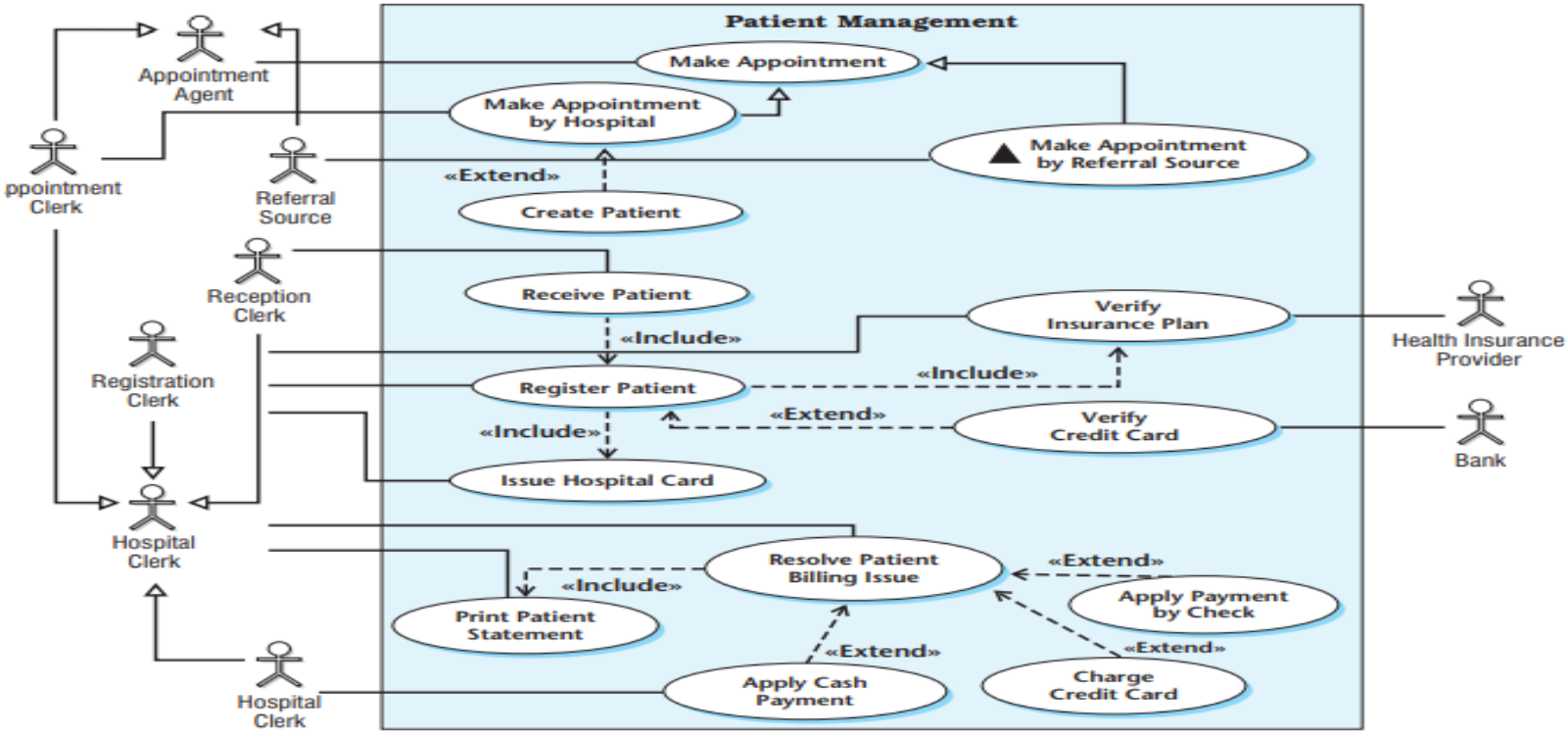
- Developing Use Cases(Behavioral Modeling)

- A use case diagram is a meta-model that portrays associations among actors, use cases, and the system.
- **Use cases** provide a crucial **framework for analysis, design, implementation, and deployment activities**

Essential Use Case Modeling

- Developing Use Cases(Behavioral Modeling)

- A Meta-Model for the "Big Picture"



Essential Use Case Modeling

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ Assignment:

1. Identify major use cases and create a use case summary for the application in your course work project.
2. Create a context diagram.
3. Refer to the online shopping system in Gathering Requirements for the following exercises.
4. Identify primary and secondary actors.
5. Identify major use cases and create a use case summary.
6. Create a context diagram for the system.(**read about context diagram)**

➢ Assignment:

7. **Identify understand the following phrases about requirements**

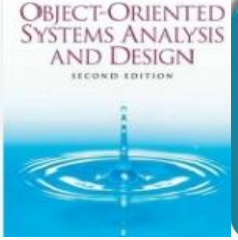
   A. Requirements Anticipation.
   B. Requirements Elicitation.
   C. Requirements Assurance.
   D. Requirements Specification.

Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.

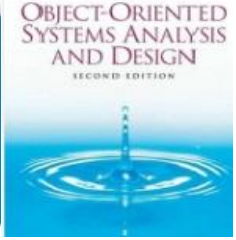
OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

➢ CRC

- ***Class* Responsibility *Collaboration (CRC) cards*** provide an effective technique for exploring the possible ways of allocating responsibilities to classes and the collaborations that are necessary to fulfill the responsibilities.
- ***CRC cards*** can be used at several different stages of a project for different purposes.
  - For example, they can be used **early in a project to aid the production of an initial class diagram** and **to develop a shared understanding of user requirements among the members of the team.**



Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.



➢ CRC

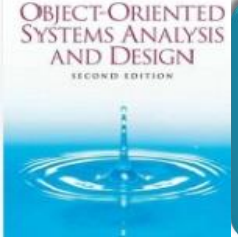
- Purpose of this part is to transform the requirement analysis represented by the use-case diagram(s) to design diagrams using static analysis.
- The result of this phase is UML class diagrams consisting of classes and relationship among them.
- We will study techniques to discover classes and the UML notations for representing class diagrams.

Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.

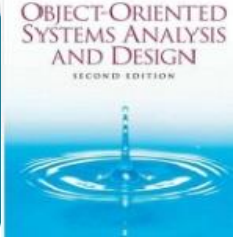
OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN
SECOND EDITION

➢ CRC

- Although proponents of the object paradigm often say that identifying objects is a simple and intuitive process.
- The solution is to use the **CRC process** to determine the classes necessary to the system as part of the design process for the application.
- **CRC (classes, responsibility, and collaboration) cards** can be used to visualize and test different class-based models during the design phase.
- It is a proven technique used and advocated by leading methodologists.

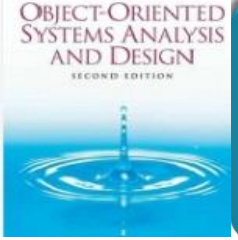


Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.

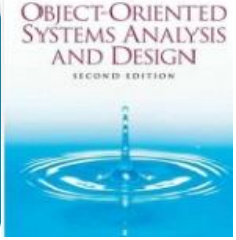


➢ CRC Card

- Design must be kept simple and no functionality should be added before it is scheduled.
- Developers should use **CRC cards** to represent objects. The objects take shape in CRC meetings where the cards are passed around to the participants.
- **CRC cards** were originally introduced as a tool for performing requirements elicitation and analysis



Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.



➢ CRC Card

- **CRC cards** are indeed a powerful tool for this purpose, but only when the **information technology professionals are familiar with the domain**; that is, the specific business environment in which the information system is to operate, such as **aero- space**, **banking**, or **clothing manufacturing**.
- However, even for information technology professionals who have no domain expertise whatsoever, **CRC cards** are an extremely effective way of **testing object-oriented analysis and design artifacts.**

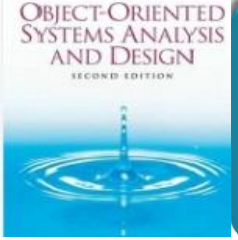
Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

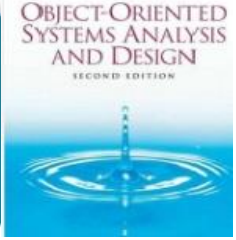
➢ CRC Card

❖ **Format of a CRC card**.

Class Name:	
Responsibilities	Collaborations
Responsibilities of a class are listed in this section	*Collaborations with other classes are listed here, together with a brief description of the purpose of the collaboration*



Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.



➢ CRC Card

- Nowadays the whole process can be automated; System Architect includes modules for creating and updating CRC "cards" on the screen.
- The strength of CRC cards is that, when utilized by a team, the interaction between the members can highlight missing or incorrect fields in a class, whether attributes or operations. Also, relationships between classes are clarified when CRC cards are used.
- One especially powerful technique is to distribute the cards among the team members who then act out the responsibilities of their classes.

➢ Example of CRC card

Class Name	ShoppingCart
Responsibilities	- Add item - Remove item - Calculate total
Collaborators	- Product - Customer

Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

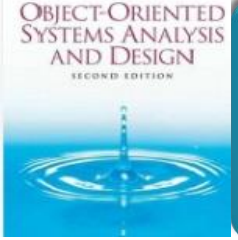
A CRC Card for Class Mortgage Class

CLASS Mortgage Class	
RESPONSIBILITY	COLLABORATION
Compute estimated grants and payments for week	Estimate Funds for Week Class
Initialize, update, and delete mortgages	Manage an Asset Class
Generate list of mortgages	User Interface Class
Print list of mortgages	Mortgages Report Class

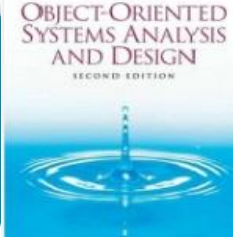
➢ CRC Card

- The CRC card technique has subsequently been extended.

1. **A CRC card nowadays often explicitly contains the attributes and operations of the class, rather than just its "responsibility" expressed in some natural language like English.** That is, the CRC card essentially contains the information of the complete class diagram
2. **The technology has changed**. Instead of using cards, some organizations put the names of the classes on Post-it notes that they then move around on a white board; lines are drawn between the Post-it notes to denote collaboration.



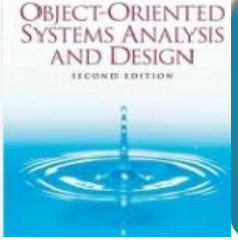
Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.



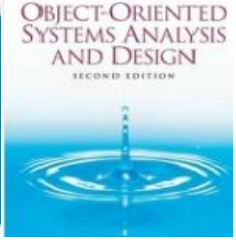
➢ CRC Card

❖ Why CRC is Useful:

1. It helps in **brainstorming** object-oriented designs.
2. Encourages **encapsulation** by focusing on responsibilities.
3. Promotes **collaboration** by making you think about which classes should interact.
4. Lightweight and **easy to understand** — good for team discussions



Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.



➢ Assignment on CRC Card

1. Identify the key concepts(Classes) for the course work project system
2. Create CRC Cards for your system

➢ Essential User Interface Prototyping

- **User interface (UI) prototyping** is a crucial phase in the design and development of software applications, websites, and other interactive systems.
- It involves creating ***mockups or prototypes of the user interface to visualize and test the layout, functionality, and usability before proceeding with full-scale development***.

Here are some key points about UI prototyping:

1. **Visualization of Design Concepts**: UI prototypes allow designers and stakeholders to visualize design concepts and ideas in a tangible form.
   - They provide a concrete representation of the user interface, including layout, navigation, controls, and visual elements.

➢ Essential User Interface Prototyping

2. **Iterative Design Process**: UI prototyping is often an iterative process, where multiple iterations of prototypes are created and refined based on feedback from users, stakeholders, and usability testing.

❖ This iterative approach helps to refine the design, improve usability, and address potential issues early in the development process.

3. **Low-Fidelity vs. High-Fidelity Prototypes:** UI prototypes can be categorized as low-fidelity or high-fidelity based on the level of detail and fidelity to the final design.

❖ **Low-fidelity prototypes** are quick and simple representations that focus on basic layout and functionality, while high-fidelity prototypes closely resemble the final product in terms of appearance and interaction.

➢ Essential User Interface Prototyping

4. **Tools and Techniques**: Various tools and techniques are available for creating UI prototypes, ranging from paper sketches and wireframes to interactive mockups and clickable prototypes. Common tools include Adobe XD, Sketch, Figma, InVision, Axure RP, and Balsamiq, among others.
5. **User Feedback and Testing**: UI prototypes are used to gather feedback from users through usability testing, interviews, and surveys. By observing how users interact with the prototype, designers can identify usability issues, navigation problems, and areas for improvement.

➢ Essential User Interface Prototyping

6. **Communication and Collaboration**: UI prototypes serve as a communication tool for designers, developers, and stakeholders to collaborate and align on the design vision and requirements. They help to ensure that everyone involved in the project has a clear understanding of the intended user experience and interface design.
7. **Risk Reduction:** UI prototyping helps to mitigate risks associated with design decisions by allowing designers to experiment with different layouts, features, and interactions in a low-risk environment. By identifying and addressing potential issues early, UI prototypes can help to avoid costly redesigns and rework later in the development process.

➢ Essential User Interface Prototyping

- **UI prototyping** is an essential part of the design and development process, enabling designers to create user-centered interfaces that are **intuitive**, **efficient**, and **enjoyable** to use.
- By iteratively refining and testing prototypes, designers can ensure that the final product meets the needs and expectations of users while aligning with **business goals and technical constraints.**

➢ Essential User Interface Prototyping

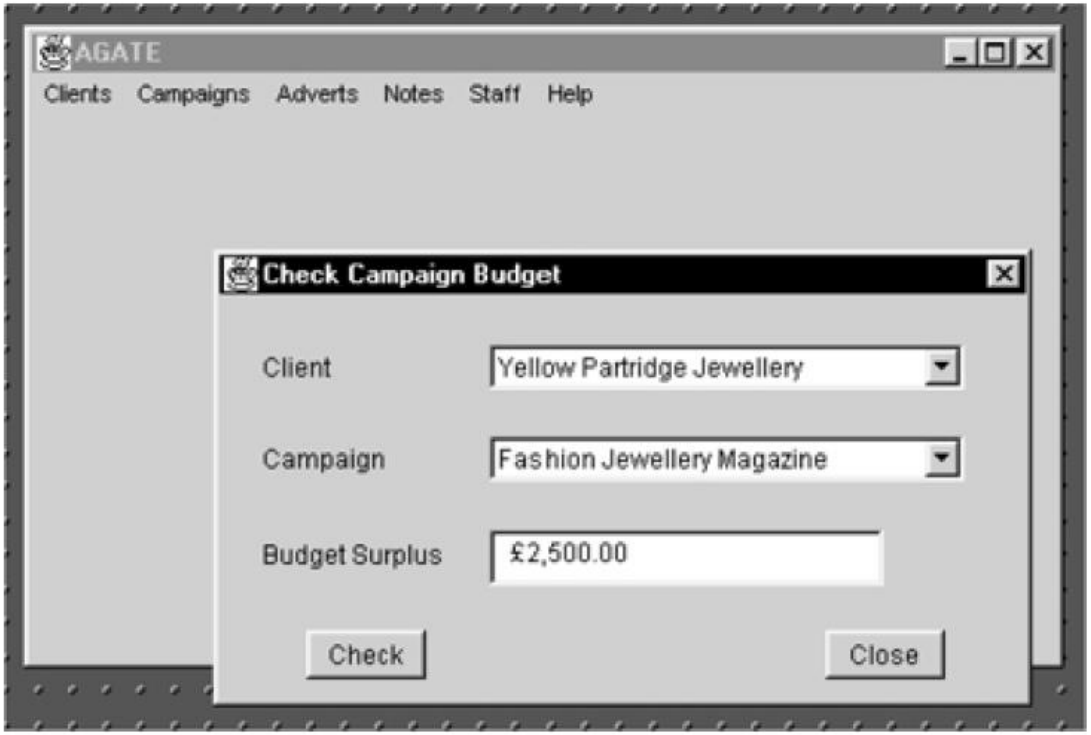
❖ Purpose of Essential UI Prototyping:

1. To clarify the user's goals and needs.
2. To identify and define the main interactions between the user and the system.
3. To avoid premature decisions about design or technology.
4. Helps in communicating the system's functionality to stakeholders

Classes, Responsibility, Collaboration (CRC) method for discovering classes and assigning responsibility.

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN SECOND EDITION

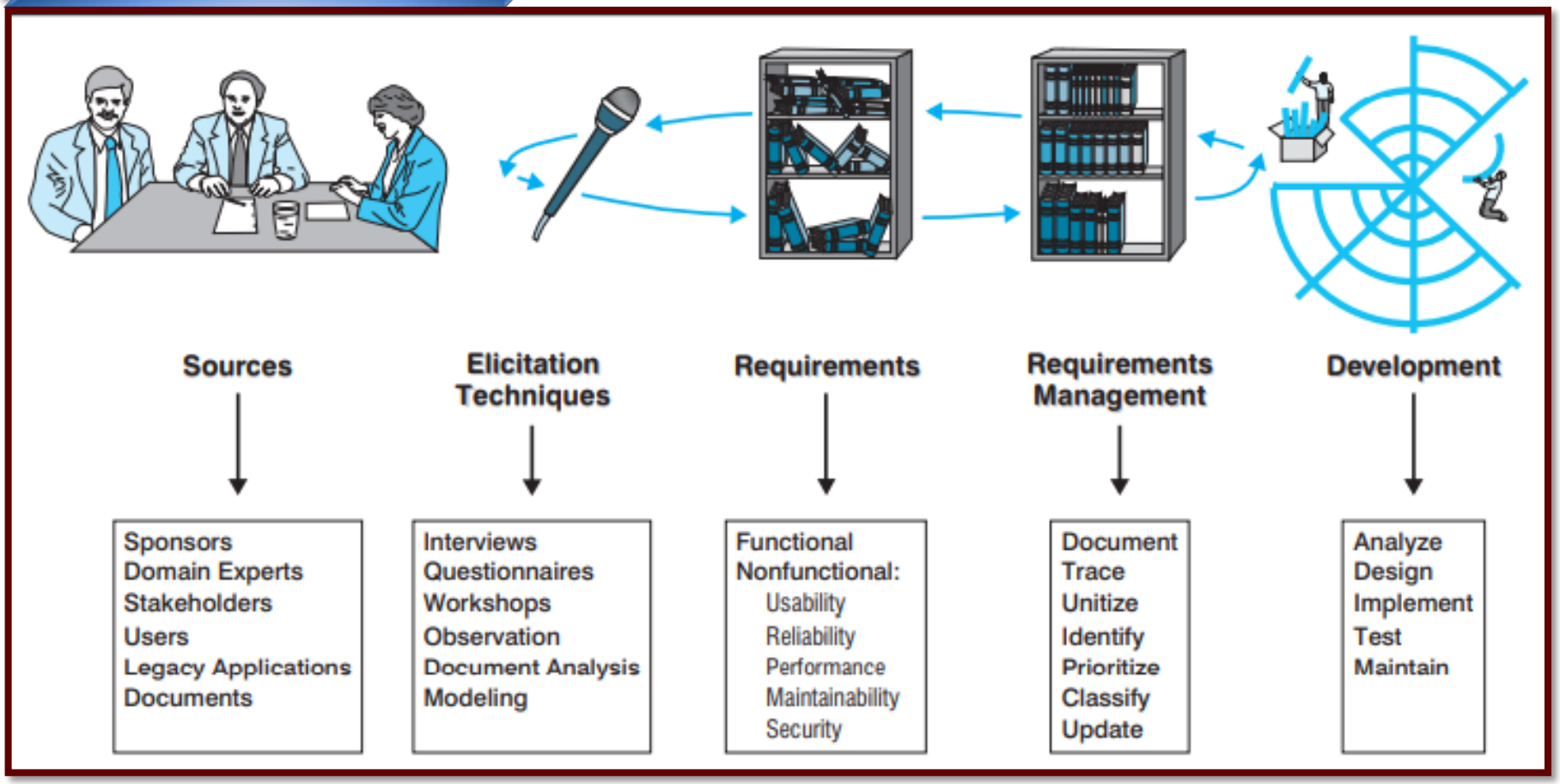
Essential User Interface Prototyping



- *Interface for the use case Check Campaign Budget developed in Java*

- **End of Chapter IV**

The End of Unit IV



Agile Methods in General