Object Oriented Software Engineering

MCA Semester III

Department of Computer Science

Gujarat University

2 Requirement Engineering and Object-Oriented Software Estimation

- **Requirement Engineering:**
- Requirement engineering is a crucial initial step in software development. It lays the groundwork for the entire project’s success by understanding, documenting, and defining the expectations of users and stakeholders related to the software application.

2 Requirement Engineering and Object-Oriented Software Estimation

- **Key activities in requirement engineering include:**
- Problem recognition: Identifying the need for software.
- Evaluation and synthesis: Analyzing and combining information.
- Modeling: Creating representations of requirements.
- Specification: Documenting requirements.
- Review: Ensuring clarity and correctness.

2 Requirement Engineering and Object-Oriented Software Estimation

- **Object-Oriented Software Estimation:**
- Object-oriented estimation focuses on predicting development effort for software projects using object-oriented design paradigms.
- It involves methods to estimate complexity, quality, and size based on object-oriented properties.

2 Requirement Engineering and Object-Oriented Software Estimation

- **Common estimation techniques include:**
- Expert opinion-based: Relying on experts’ judgment.
- Top-down estimation: Starting with high-level estimates and refining them.
- Bottom-up estimation: Breaking down tasks and estimating each component.
- Parametric or algorithmic models: Using mathematical models to predict effort.

2 Requirement Engineering and Object-Oriented Software Estimation

- **Functional and Non-Functional Requirements**
- **Functional Requirements:** These are specific behaviors or functions of a system. They describe what the system should do. For example, in an online banking system, a functional requirement could be the ability for users to transfer funds between accounts. This requirement specifies the exact functionality that the system must provide.

Requirement Engineering and Object-Oriented Software Estimation

- **Non-Functional Requirements:** These define the system’s attributes such as performance, usability, reliability, and security. They describe how the system performs a function rather than the function itself. For instance, the online banking system should be able to handle 1,000 transactions per second. This ensures the system’s performance under high load conditions.
- Example: Consider a ride-sharing app like Uber. A functional requirement could be the ability for users to book a ride. A non-functional requirement could be that the app should respond to booking requests within 2 seconds. This ensures a smooth user experience.

Requirement Engineering and Object-Oriented Software Estimation

- **Requirement Elicitation Techniques**
- Interviews: This technique involves direct conversations with stakeholders to gather requirements. It is effective for understanding detailed needs and expectations. For example, interviewing doctors and nurses to gather requirements for a hospital management system.
- Surveys/Questionnaires: These are used to collect data from a large audience. They are useful when you need input from many people. For instance, sending out surveys to patients to understand their needs for an online appointment booking system.

Requirement Engineering and Object-Oriented Software Estimation

- **Requirement Elicitation Techniques**
- Workshops: These are collaborative sessions with stakeholders to gather requirements. They are effective for brainstorming and reaching a consensus. For example, conducting a workshop with teachers and students to gather requirements for an educational platform.
- Observation: This involves watching how users interact with the current system to identify requirements. It is useful for understanding real-world usage. For instance, observing customer service representatives to gather requirements for a customer support system.

Requirement Engineering and Object-Oriented Software Estimation

- **Real-Life Example:** In developing a new e-commerce platform, a combination of interviews with business owners, surveys to potential customers, workshops with the development team, and observation of current shopping behaviors can provide a comprehensive set of requirements.

Software Requirement Specification (SRS)

- **Software Requirement Specification (SRS)**
- **Definition:** An SRS is a document that describes what the software will do and how it will be expected to perform. It serves as a blueprint for the development team.
- **Components:** An SRS typically includes an introduction, overall description, specific requirements, and appendices. The introduction provides an overview of the project. The overall description gives a high-level view of the system. Specific requirements detail the functional and non-functional requirements.

Software Requirement Specification (SRS)

- Example: For an e-commerce website, the SRS might include user roles (admin, customer), functional requirements (product search, shopping cart), and non-functional requirements (system should handle 10,000 concurrent users).
- Importance: An SRS ensures that all stakeholders have a clear understanding of the system requirements. It helps in avoiding misunderstandings and ensures that the development team knows exactly what to build.

Requirements Change Management

- Definition: This is the process of managing changes to the requirements throughout the project lifecycle. It ensures that changes are evaluated, approved, and implemented systematically.
- Steps: The steps in change management include identification, analysis, approval, implementation, and verification. Identification involves recognizing the need for a change. Analysis assesses the impact of the change. Approval involves getting the necessary sign-offs. Implementation is the actual change, and verification ensures the change meets the requirements.
- Example: In a software project, a change request might be to add a new feature based on user feedback. This request would go through the change management process to ensure it is feasible and beneficial.

Need for Cost and Schedule Estimation

- Importance: Cost and schedule estimation are critical for planning, budgeting, and resource allocation. Accurate estimates help in setting realistic expectations and ensuring that the project is completed on time and within budget.
- Methods: Common estimation methods include expert judgment, analogous estimating, and parametric estimating. Expert judgment relies on the experience of experts. Analogous estimating uses historical data from similar projects. Parametric estimating uses mathematical models to estimate costs and time.
- Example: In a software development project, expert judgment might be used to estimate the effort required for a new feature based on the experience of senior developers.

Need for Cost and Schedule Estimation

- Challenges: Estimating costs and schedules can be challenging due to uncertainties and complexities in projects. It requires careful analysis and consideration of various factors such as scope, resources, and risks.
- Real-Life Example: In a project to develop a new mobile app, cost and schedule estimation would involve analyzing the scope of work, identifying the required resources, and estimating the time and cost for each phase of the project.

Lorenz and Kidd Estimation Method

- The Lorenz and Kidd Estimation Method provides a structured approach to estimating the size and effort of object-oriented software projects.
- Key Concepts of Lorenz and Kidd Estimation Method
- Scenario Scripts (Use Cases):
- Definition: Scenario scripts, also known as use cases, describe the interactions between users and the system to achieve specific goals.

Lorenz and Kidd Estimation Method

- Estimation: The number of scenario scripts is used to estimate the number of classes in the system. According to Lorenz and Kidd, the number of classes can be estimated as:
- Number of Classes=17×Number of Scenario Scripts
- Example: If a library management system has 10 scenario scripts (e.g., Borrow Book, Return Book, Search Catalog), the estimated number of classes would be (17 \times 10 = 170) classes.

Lorenz and Kidd Estimation Method

Class Estimation:

- Definition: Classes are the fundamental building blocks of object-oriented software. Each class represents a blueprint for objects.
- Estimation: The method involves estimating the number of methods per class and the complexity of these methods.
- Example: For a class Book in a library management system, methods might include borrow(), return(), and search(). The complexity of these methods is assessed to estimate the effort required for development.

Lorenz and Kidd Estimation Method

Effort Calculation:

- Definition: Effort calculation involves determining the total effort required to develop the software based on the estimated number of classes and methods.
- Steps:
- Identify the classes and methods.
- Estimate the complexity of each method.
- Calculate the total effort based on the complexity and number of methods.
- Example: If the Book class has 3 methods with medium complexity, and each method requires 5 person-days, the total effort for the Book class would be (3 \times 5 = 15) person-days.

Lorenz and Kidd Estimation Method

Advantages:

- Early Estimation: This method allows for early estimation of the size and effort, which is crucial for planning and budgeting.
- Object-Oriented Focus: It is specifically designed for object-oriented software, making it more accurate for such projects.
- Scalability: The method can be scaled to estimate larger systems by breaking down the system into smaller components.

Lorenz and Kidd Estimation Method

Real-Life Example:

- Project: Developing an online shopping platform.
- Scenario Scripts: Add to Cart, Checkout, View Order History, etc.
- Estimation: If there are 15 scenario scripts, the estimated number of classes would be (17 \times 15 = 255) classes.
- Effort Calculation: For a class Product, with methods like addToCart(), removeFromCart(), and viewDetails(), if each method is of medium complexity requiring 4 person-days, the total effort for the Product class would be (3 \times 4 = 12) person-days.

Use case Points method

The Use Case Points (UCP) Method is a software estimation technique used to measure the size of a software project based on its use cases. Developed by Gustav Karner in 1993, this method helps in estimating the effort required for a project by analyzing the use cases, actors, and various technical and environmental factors.

- Key Concepts of Use Case Points Method
- Unadjusted Use Case Weight (UUCW):
- Definition: This is the sum of the weights of all use cases in the system. Use cases are classified as simple, average, or complex based on the number of transactions they contain.

Use case Points method

- Classification:
- Simple: ≤ 3 transactions, weight = 5
- Average: 4 to 7 transactions, weight = 10
- Complex: > 7 transactions, weight = 15
- Example: For an online shopping system, if there are 3 simple use cases, 5 average use cases, and 2 complex use cases, the UUCW would be calculated as:
- $UUCW=(3\times 5)+(5\times 10)+(2\times 15)=15+50+30=95$

Use case Points method

Unadjusted Actor Weight (UAW):

- Definition: This is the sum of the weights of all actors interacting with the system. Actors are classified as simple, average, or complex based on their interaction with the system.
- Classification:
- Simple: External system with a defined API, weight = 1
- Average: External system interacting through a protocol, weight = 2
- Complex: Human user interacting through a GUI, weight = 3
- Example: For the same online shopping system, if there are 2 simple actors, 3 average actors, and 1 complex actor, the UAW would be calculated as:
- $UAW=(2\times1)+(3\times2)+(1\times3)=2+6+3=11$

Use case Points method

Technical Complexity Factor (TCF):

- Definition: This factor adjusts the UUCW and UAW based on technical considerations. It is calculated using 13 technical factors, each rated on a scale from 0 to 5.
- Formula:
- TCF=0.6+(0.01×i=1∑13Ti)
- Example: If the sum of the ratings for all technical factors is 35, the TCF would be:
- TCF=0.6+(0.01×35)=0.6+0.35=0.95

Use case Points method

Environmental Complexity Factor (ECF):

- Definition: This factor adjusts the UUCW and UAW based on environmental considerations. It is calculated using 8 environmental factors, each rated on a scale from 0 to 5.
- Formula:
- ECF=1.4+(−0.03×i=1∑8Ei)
- Example: If the sum of the ratings for all environmental factors is 20, the ECF would be:
- ECF=1.4+(−0.03×20)=1.4−0.6=0.8

Use case Points method

Calculating Use Case Points (UCP):

- Formula:
- $UCP=(UUCW+UAW)\times TCF\times ECF$
- Example: Using the previous examples, the UCP would be calculated as:
- $UCP=(95+11)\times 0.95\times 0.8=106\times 0.95\times 0.8=80.56$

Use case Points method

Example: In a project to develop a customer relationship management (CRM) system, the classes might include Customer, Contact, and Opportunity. Each class would be assigned a weight based on its complexity, and the total effort calculated.

- Advantages: This method provides a systematic approach to estimating the size of object-oriented software. It helps in understanding the complexity of the system and planning the development effort accordingly.

Object-Oriented Function Point (OOFP)

- The Object-Oriented Function Point (OOFP) method is a technique used to measure the functional size of object-oriented software. It extends the traditional Function Point Analysis (FPA) to accommodate the unique aspects of object-oriented design.
- Key Concepts of OOFP
- Objects:
- Definition: Objects are instances of classes that encapsulate data and behavior. In OOFP, objects are the primary units of measurement.
- Example: In a library management system, objects might include Book, Member, and Loan.

Object-Oriented Function Point (OOFP)

- Methods:
- Definition: Methods are functions or procedures associated with objects. They define the behavior of the objects.
- Example: For the Book object, methods might include borrow(), return(), and search().
- Interactions:
- Definition: Interactions refer to the communication between objects. These interactions are crucial for understanding the system's functionality.
- Example: In the library management system, the Member object interacts with the Book object to borrow a book.

Object-Oriented Function Point (OOFP)

- Real-Life Example
- Project: Developing an online banking system.
- Identify Objects:
- Objects: Account, Transaction, Customer, Loan
- Identify Methods:
- Account: openAccount(), closeAccount(), viewBalance()
- Transaction: deposit(), withdraw(), transfer()
- Customer: register(), login(), updateProfile()

Risk Management

- Risk management is the process of identifying, assessing, and mitigating risks that could potentially affect a project's success. It involves a systematic approach to managing uncertainties and ensuring that potential issues are addressed before they become problems.
- Key Concepts of Risk Management
- Risk Identification:
- Definition: The process of identifying potential risks that could affect the project. This involves brainstorming sessions, expert consultations, and reviewing historical data.
- Example: In a software development project, risks might include technical challenges, resource shortages, and changing requirements.

Risk Management

- Risk Assessment:
- Definition: Evaluating the identified risks to determine their likelihood and impact. This helps in prioritizing risks based on their potential effect on the project.
- Example: Assessing the risk of a key team member leaving the project. The likelihood might be low, but the impact could be high if the member has specialized knowledge.

Risk Management

- Risk Mitigation:
- Definition: Developing strategies to reduce the likelihood or impact of risks. This involves planning and implementing measures to manage risks effectively.
- Example: To mitigate the risk of data breaches, a company might implement strong security protocols, conduct regular security audits, and provide training to employees.

Risk Management

- Risk Monitoring:
- Definition: Continuously tracking identified risks and monitoring new risks throughout the project lifecycle. This ensures that risk management strategies are effective and updated as needed.
- Example: Regularly reviewing project progress and conducting risk assessments during project meetings to identify any new risks or changes in existing risks.

Risk Management

- Risk Response Planning:
- Definition: Developing action plans for responding to risks if they occur. This includes contingency plans and assigning responsibilities for risk management.
- Example: In a construction project, having a contingency plan for adverse weather conditions, such as rescheduling tasks or arranging for temporary shelters.

Risk Management

- Real-Life Examples
- Compliance Risk:
- Scenario: A pharmaceutical company must comply with stringent regulatory requirements.
- Risk Management: The company conducts regular compliance audits, trains employees on regulatory standards, and implements a compliance management system to ensure adherence to regulations.

Risk Management

- Real-Life Examples
- Safety Risk:
- Scenario: A manufacturing plant faces safety risks due to the use of heavy machinery.
- Risk Management: The plant implements safety protocols, conducts regular safety drills, and provides personal protective equipment (PPE) to workers to minimize the risk of accidents.

Assignment - 2

- Define functional and non-functional requirements. Provide two real-life examples of each.
- Explain why both functional and non-functional requirements are crucial for the success of a software project.
- Describe four requirement elicitation techniques and provide a real-life example where each technique would be most effective.
- Discuss the challenges faced during requirement elicitation and suggest strategies to overcome them.
- What are the key components of a Software Requirement Specification (SRS) document? Illustrate with an example of an e-commerce website.
- Explain the importance of an SRS document in the software development lifecycle.
- Outline the steps involved in requirements change management. Provide a real-life example of a change request and how it was managed.

Assignment - 2

- Discuss the impact of poorly managed requirement changes on a software project.
- Why is cost and schedule estimation important in software projects? Describe three common estimation methods.
- Provide a detailed example of how cost and schedule estimation was conducted for a mobile app development project.
- Explain the Lorenz and Kidd Estimation Method in detail. Use a real-life example to illustrate the process of estimating the size of an object-oriented software project.
- Describe the Use Case Points Method for software estimation. Provide a detailed example of how this method is applied to an online banking system.