Lecture Notes: Introduction to Software Engineering (COMP2000)

Course Objective:

- Learn best practices in software development.
- Explore various programming paradigms used in mobile development.
- Understand and implement standardised design patterns.

Assessments:

- Coursework (30%) and a Report (70%).
- The coursework involves developing an Android app using Java and applying lecture concepts.

Key Topics Covered:

1. **Software Engineering Overview:**
    - Software engineering involves analysing user requirements, designing, building, and testing software applications to meet those needs.
    - This process follows a structured approach known as the **Software Development Life Cycle (SDLC)**.
2. **Software Development Life Cycle (SDLC):**
    - Divided into seven key steps:
        - Planning & Requirement Analysis
        - Defining Requirements
        - Designing the Product Architecture
        - Building/Developing the Product
        - Testing the Product
        - Deployment & Maintenance
    - Each phase includes detailed activities like feasibility analysis, risk management, architecture design (high-level and low-level), and unit testing.
3. **SDLC Models:**
    - **Waterfall Model:** Sequential process where each phase depends on the previous one, suitable for small or critical projects.
    - **Agile Model:** Iterative approach with short development cycles (1-4 weeks), reducing project risk and delivery time.
4. **Design Activities:**
    - Start once some requirements are gathered.
    - **Prototyping** and iterative cycles (design-evaluation-redesign) involving users help refine the design.
    - Two types of design:
        - **Conceptual Design:** Focuses on what the product will do and how.
        - **Concrete Design:** Includes details such as menu structures and physical feedback.
5. **Prototyping:**
    - Prototyping is crucial for evaluating design ideas.
    - There are two types:

        - **Low Fidelity Prototyping:** Cheap, quick, and flexible, useful for early exploration (e.g., paper prototypes).
        - **High Fidelity Prototyping:** Closer to the final product, used for testing functionality and technical feasibility.
6. **Storyboarding:**
    - A technique used for low fidelity prototyping that visually represents user interaction with the product through sketches or screens.
    - Helps in early-stage design to illustrate user flow and potential scenarios.
7. **Compromises in Prototyping:**
    - Prototypes involve trade-offs between breadth (range of functions) and depth (detail of functionality).
    - Some compromises are evident (e.g., non-functional prototypes), while others may not be (e.g., code quality or internal structure).
8. **Conceptual Design:**
    - Involves transforming user requirements into a conceptual model that outlines what the user can do and the needed concepts for interaction.
    - Tools like scenarios and prototypes are useful in this process to experiment and gather feedback.

Further Reading: Interaction Design: Beyond Human-Computer Interaction by Sharp, Rogers, and Preece

Overview: This book is a comprehensive guide to understanding the principles of interaction design, especially as it pertains to human-computer interaction (HCI). It bridges the gap between academic research and practical application in the field of interaction design, making it an invaluable resource for software engineers, UX/UI designers, and anyone interested in developing user-friendly digital interfaces.

Key Topics Covered:

1. **Human-Centered Design**:
    - Emphasises designing products with the user at the core.
    - Understand the needs, abilities, and limitations of users to create systems that are easy to use and accessible.
2. **Design Processes**:
    - Iterative design approach that involves cycles of design, evaluation, and redesign.
    - Prototyping as an essential part of testing design concepts and refining them based on user feedback.
3. **Cognitive Models**:
    - Insights into how users process information and interact with technology.

    - Design strategies that align with human cognitive processes, making interactions more intuitive.
4. **Prototyping and Evaluation**:
    - The role of low- and high-fidelity prototypes in the design process.
    - Techniques for evaluating designs with users, including usability testing and scenario-based evaluations.
5. **Interface Design**:
    - Guidelines for creating user interfaces that are not only functional but also aesthetically pleasing.
    - Focus on aspects such as navigation, layout, and visual elements that contribute to an effective user experience.
6. **Emerging Technologies**:
    - Exploration of new technologies, including voice interaction, augmented reality, and wearable devices.
    - Challenges and opportunities in designing for these evolving platforms.

Why It's Important for Software Engineering:

- This book provides essential knowledge for software engineers, particularly in understanding how to design user-friendly interfaces and systems.
- It aligns well with concepts like **Agile Development**, where frequent user testing and iterative design are critical.
- Applying the principles of interaction design can lead to software that not only meets functional requirements but also offers a seamless user experience.

Suggested Application:

- Use this book as a guide during the prototyping and design phases of your software development process.
- Implement user-centered design practices and conduct usability tests to ensure your software meets the needs of its target audience.

This reading complements the topics covered in the lecture on software engineering and will provide deeper insights into how to improve the interaction and usability of software products.

-

Summary:

- Introduction to SDLC, models like Waterfall and Agile.
- Importance of prototyping in design.
- Differences between conceptual and concrete designs.
- Techniques like sketching and storyboarding for early design stage