Trường Đại Học Văn Lang
Khoa Kỹ Thuật Cơ Điện và Máy Tính
Bộ Môn Kỹ Thuật Phần Mềm

HƯỚNG DẪN

ĐỒ ÁN THIẾT KẾ PHẦN MỀM

1. Giới thiệu tổng quan về đồ án Kỹ Thuật Phần Mềm

Đồ án tốt nghiệp Kỹ Thuật Phần Mềm (Software Engineering Capstone Project) là một dự án thực tiễn nhằm giúp sinh viên áp dụng kiến thức đã học vào việc thiết kế, phát triển và đánh giá một hệ thống phần mềm hoàn chỉnh. Đồ án đòi hỏi sinh viên phải hiểu rõ quy trình phát triển phần mềm, từ phân tích yêu cầu đến triển khai và bảo trì. Ngoài ra, đồ án cũng giúp sinh viên nâng cao kỹ năng làm việc nhóm, lập kế hoạch dự án và tuân thủ các tiêu chuẩn công nghiệp.

2. Phân tích nghiệp vụ kinh doanh (Business Analysis) và ConOps

- **Business Analysis**: Xác định vấn đề kinh doanh, đối tượng người dùng và các yêu cầu tổng quan. Đây là bước quan trọng để đảm bảo phần mềm đáp ứng nhu cầu thực tế của doanh nghiệp.
- **Concept of Operations (ConOps)**: Mô tả cách hệ thống phần mềm sẽ hoạt động trong môi trường thực tế, bao gồm các luồng công việc, kịch bản sử dụng chính, và cách người dùng tương tác với hệ thống.

3. Kỹ thuật lấy yêu cầu (Requirements Engineering) và SRS

Quy trình kỹ thuật lấy yêu cầu giúp xác định các thông tin cần thiết để phát triển phần mềm một cách chính xác và đầy đủ. Tài liệu SRS (Software Requirement Specification) đóng vai trò là nền tảng cho việc thiết kế và phát triển phần mềm.

- **Business Requirements**: Xác định các mục tiêu và yêu cầu kinh doanh dựa trên nhu cầu của khách hàng và thị trường.
- **User Requirements**: Thu thập các yêu cầu từ người dùng thông qua khảo sát, phỏng vấn, và phân tích Use Cases hoặc User Stories.
- **System Requirements**: Định nghĩa các yêu cầu hệ thống bao gồm hiệu suất, bảo mật, tính mở rộng và tích hợp.
- **Software Requirements**: Viết tài liệu SRS để làm cơ sở cho quá trình phát triển phần mềm, bao gồm các ràng buộc kỹ thuật và chức năng cụ thể.

4. Software Architecture and Design với SDS

4.1. Software Architecture

Software Architecture đóng vai trò quyết định đến khả năng mở rộng, bảo trì và hiệu suất của hệ thống.

- **Architectural Styles**: Các phong cách kiến trúc phổ biến như Layered, Microservices, Client-Server, Event-Driven.
- **Architectural Patterns**: Các mẫu kiến trúc như MVC (Model-View-Controller), MVVM, CQRS (Command Query Responsibility Segregation).
- **Quality Tactics**: Các chiến thuật để cải thiện chất lượng phần mềm, bao gồm Performance Optimization, Security Enhancement, Scalability Tactics.
- **Design Patterns**: Áp dụng các mẫu thiết kế như Singleton, Factory, Observer, Strategy để tối ưu hóa phát triển phần mềm.
- **Technology Stack and Tools Idioms**: Lựa chọn công nghệ phù hợp cho dự án, bao gồm ngôn ngữ lập trình, framework, cơ sở dữ liệu, công cụ CI/CD.

4.2. Software Design

Software Design xác định cách hệ thống được tổ chức và hoạt động.

- **Design Methodology**: So sánh các phương pháp phát triển phần mềm như Plan Driven (Waterfall) và Agile (Scrum, Kanban).
- **Design Approach**: Sử dụng phương pháp tiếp cận Top-Down hoặc Bottom-Up để thiết kế hệ thống.
- **Design Paradigm**: So sánh lập trình hướng đối tượng (OOP) với lập trình cấu trúc để đưa ra lựa chọn phù hợp.
- **Design Method**: Các phương pháp thiết kế phần mềm như ADD (Attribute-Driven Design), ACDM, COMET.

4.3. Software Documents

Tài liệu phần mềm giúp đảm bảo tính nhất quán và dễ bảo trì.

- **Architectural Drivers**: Liên kết các yêu cầu chức năng, phi chức năng với kiến trúc phần mềm.
- **Architectural Structure Perspectives**: Phân tích các góc nhìn của kiến trúc, bao gồm Static View, Dynamic View, Physical View.
- **Software Documents**:
    - **Module View**: Biểu đồ phân rã hệ thống thành các mô-đun nhỏ.
    - **C&C View**: Biểu đồ thành phần và kết nối (Component and Connector View).
    - **Allocation View**: Sơ đồ triển khai phần mềm (Deployment Diagram) thể hiện cách hệ thống phân bố tài nguyên.

- **Use Cases Documents and ADL**: Sử dụng UML để biểu diễn luồng công việc và ADL để mô tả kiến trúc hệ thống.

5. Software Architecture Evaluation Method trade-offs

So sánh và đánh giá các phương pháp kiểm tra kiến trúc phần mềm như:

- **ATAM (Architecture Tradeoff Analysis Method)**: Đánh giá kiến trúc dựa trên các yếu tố chất lượng.
- **SAAM (Software Architecture Analysis Method)**: Đánh giá kiến trúc dựa trên các kịch bản sử dụng thực tế.

6. Software Construction

- **Software Technology Stacks**: Lựa chọn các công nghệ phù hợp để xây dựng phần mềm, bao gồm ngôn ngữ lập trình, framework, cơ sở dữ liệu, và các công cụ hỗ trợ khác.
- **Tools**: Sử dụng các công cụ phát triển phần mềm để hỗ trợ quá trình xây dựng, bao gồm IDE (Integrated Development Environment), code editor, version control system (VCS), và các công cụ testing.

7. Project Charter

Tài liệu Project Charter giúp xác định:

- Phạm vi dự án
- Mục tiêu dự án
- Nhóm phát triển
- Các bên liên quan

8. Project Plan

- **Management Methodologies**: Agile (Scrum, Kanban), Waterfall.
- **Tasks**: Phân chia công việc theo WBS (Work Breakdown Structure).
- **Resources**: Quản lý tài nguyên và phân bổ nhân lực hợp lý.
- **Estimation Methods**: Sử dụng Function Point, COCOMO để ước lượng thời gian và chi phí phát triển phần mềm.

9. Các tài liệu tham khảo