Review Chương 3:
Quy trình phát triển phần mềm

I. Tóm tắt nội dung:

❖ Quy trình phát triển phần mềm:

- Quy trình phát triển phần mềm là cách phần mềm được tạo ra, kết hợp mô hìnhvòng đời, kỹ thuật phát triển, công cụ sử dụng, và yếu tố con người.
- Các tổ chức khác nhau có cách tiếp cận khác nhau:
  + Một số tổ chức coi mã nguồn là tài liệu tự giải thích.
  + Một số khác yêu cầu tài liệu chi tiết và kiểm tra kỹ lưỡng từng giai đoạn. + Một số tổ chức dành 50% ngân sách phần mềm cho kiểm thử, trong khi sốkhác để người dùng phát hiện lỗi.

❖ Mô hình Quy trình Hợp nhất (Unified Process - UP):

- UP là một phương pháp hướng đối tượng có tính linh hoạt, áp dụng được cho nhiềuloại phần mềm khác nhau.
- UP không phải là một quy trình cố định mà có thể tùy chỉnh dựa trên yêu cầu dựán. - Dựa trên các nguyên tắc lặp (iteration) và gia tăng (incremental) để giảmrủi ro. - Mô hình này thay thế các phương pháp hướng đối tượng trước đây như OMT, Booch và Objectory.

❖ Năm quy trình công việc chính (Core Workflows):

` - Thu thập yêu cầu (Requirements workflow):
  + Hiểu rõ yêu cầu khách hàng, ràng buộc và mục tiêu phần mềm.
  + Sử dụng UML để mô tả trực quan hệ thống.
- Phân tích (Analysis workflow):
  + Xác định chi tiết chức năng và khả năng hiện thực hóa.
  + Giảm thiểu mơ hồ và thiếu sót trong đặc tả yêu cầu.
- Thiết kế (Design workflow):
  + Xây dựng cấu trúc hệ thống, chia nhỏ thành module hoặc lớp.
  + Cần ghi chép lại quyết định thiết kế để hỗ trợ bảo trì.
- Triển khai (Implementation workflow):
  + Viết mã theo thiết kế đã duyệt, kiểm tra tích hợp.
  + Một số lỗi chỉ xuất hiện khi các module được tích hợp.
- Kiểm thử (Testing workflow):
  + Kiểm tra từng thành phần và hệ thống tổng thể.
  + Gồm kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử sản phẩm, kiểmthửchấp nhận (acceptance test).

❖ Bốn giai đoạn của UP (Four Phases):

- Giai đoạn Khởi tạo (Inception Phase):
  + Xác định phạm vi dự án, xây dựng mô hình kinh doanh.
  + Phân tích rủi ro, lập kế hoạch ban đầu.
- Giai đoạn Mở rộng (Elaboration Phase):
  + Hoàn thiện yêu cầu và kiến trúc phần mềm.
  + Xác định chi phí và lên kế hoạch chi tiết.
- Giai đoạn Xây dựng (Construction Phase):
  + Phát triển phần mềm, kiểm thử tích hợp.
  + Xuất bản phiên bản beta để kiểm tra thực tế.
- Giai đoạn Chuyển giao (Transition Phase):
  + Kiểm thử tại môi trường thực tế, sửa lỗi cuối cùng.
  + Hoàn thiện tài liệu, phát hành phiên bản chính thức.

II. Những bài học được rút ra:

- Tầm quan trọng của kiểm thử sớm và liên tục:
  + Nếu lỗi phát hiện sau khi triển khai, chi phí sửa lỗi có thể cao gấp 100 lần so với việc phát hiện trong giai đoạn thiết kế.

+ Ví dụ: Một công ty phát triển phần mềm ngân hàng nhưng không kiểmtra khả năngxử lý số thập phân, gây sai lệch số dư tài khoản khách hàng.

- Kiểm thử tự động giúp tiết kiệm thời gian:

+ Việc sử dụng Test-Driven Development (TDD) trong Agile giúp phát hiện lỗi sớm. + Ví dụ: Một nhóm phát triển phần mềm thương mại điện tử dùng kiểmthử tự độngđể phát hiện lỗi ngay khi thêm tính năng mới.

+ Mô hình hai chiều giúp quản lý dự án tốt hơn

+ Thay vì phát triển theo kiểu tuyến tính, mô hình hai chiều (Two-Dimensional Life Cycle Model) giúp chia nhỏ dự án thành nhiều phần dễ quản lý hơn. + Ví dụ: Một công ty phát triển phần mềm quản lý bệnh viện theo mô hình lặp, giúptích hợp phản hồi từ bác sĩ ngay từ đầu.

- Quản lý rủi ro trong phát triển phần mềm:

+ Một số dự án phần mềm thất bại do xác định yêu cầu sai hoặc thay đổi yêu cầu liêntục.

+ Ví dụ: Một công ty phần mềm phát triển ứng dụng cho chính phủ nhưng không tínhđến yêu cầu bảo mật mới, dẫn đến phải làm lại từ đầu.

- Ứng dụng mô hình CMM để cải tiến quy trình:

+ CMM (Capability Maturity Model) giúp đánh giá mức độ trưởng thành của quy trình phần mềm.

+ Ví dụ: Một công ty phần mềm nhỏ muốn đạt CMM Level 3 cần chuẩn hóa quy trình phát triển và kiểm thử phần mềm.

- UP và mô hình lặp giúp phát triển phần mềm linh hoạt hơn so với mô hình thác nước truyềnthống.
- Kiểm thử liên tục giúp phát hiện lỗi sớm và giảm chi phí bảo trì. - Quản lý rủi ro và cải tiến quy trình phần mềm (CMM) giúp dự án đạt hiệu suất cao hơn. - Sử dụng UML giúp mô hình hóa phần mềm hiệu quả hơn và giảm sai sót trong giao tiếp giữa các nhóm phát triển.