Review Chương 2:
Mô hình vòng đời phần mềm

I. Tóm tắt nội dung chính:

- Phát triển phần mềm trong quản lý thuyết và thực tế:
    + Trong lý thuyết, phần mềm được phát triển theo quy trình tuyến tính:
        1. Xác định yêu cầu
        2. Phân tích
        3. Thiết kế
        4. Triển khai
        5. Cài đặt và bảo trì
    + Trong thực tế, quá trình này phức tạp hơn do lỗi của con người và sự thay đổi yêu cầu từ khách hàng.
- Mô hình vòng đời cây tiến hóa (Evolution-tree model):
    + Ví dụ về hệ thống thu phí xe buýt tại thành phố Winburg, nơi phần mềm được cải tiến qua nhiều lần chỉnh sửa để đảm bảo hiệu suất và độ chính xác.
    + Mô hình này thể hiện sự thay đổi liên tục trong phát triển phần mềm, với nhiều phiên bản cải tiến dựa trên phản hồi thực tế.
- Mô hình vòng đời thác nước (Waterfall model):
    + Đây là mô hình cổ điển, với các giai đoạn tuyến tính từ yêu cầu đến bảo trì.
    + Nhược điểm: Không linh hoạt, khó thích nghi với thay đổi.
- Mô hình vòng đời lặp và gia tăng (Iterative-and-incremental model):
    + Dựa trên nguyên tắc:
        Lặp (Iteration): Cải thiện từng phiên bản phần mềm liên tục.
        Gia tăng (Incremental): Xây dựng từng phần chức năng của phần mềm theo từng giai đoạn.
- Áp dụng Luật Miller: Chia nhỏ hệ thống để dễ quản lý do con người chỉ có thể xử lý khoảng 7 đơn vị thông tin cùng lúc.
- Mô hình phát triển Agile & Extreme Programming:
    + Agile nhấn mạnh vào sự linh hoạt, phản hồi nhanh từ khách hàng và cải tiến liên tục.
    + Extreme Programming (XP) gồm các nguyên tắc như lập trình cặp (pair programming) và phát triển theo hướng kiểm thử (test-driven development - TDD).
    + Nhược điểm: Chưa được chứng minh hiệu quả với dự án lớn, có thể tốn kém khi bảo trì.
- Mô hình mã nguồn mở (Open-Source Life-Cycle Model):
    + Phần mềm được phát triển bởi cộng đồng, không có giai đoạn yêu cầu và thiết kế rõ ràng.
    + Ví dụ thành công: Linux, Apache, Firefox.

II. Những bài học được rút ra:

- Tầm quan trọng của mô hình lặp và gia tăng:
    + Ví dụ: Một công ty phát triển phần mềm kế toán theo mô hình thác nước có thể gặp khó khăn khi luật thuế thay đổi. Nếu sử dụng mô hình lặp và gia tăng, họ có thể điều chỉnh dễ dàng hơn.
- Phát hiện lỗi sớm giúp giảm chi phí:
    + Nếu một lỗi được phát hiện ngay từ giai đoạn thiết kế, chi phí sửa có thể thấp hơn hàng trăm lần so với khi lỗi được phát hiện sau khi phần mềm đã triển khai.
- Sự khác biệt giữa phần mềm mã nguồn mở và mã nguồn đóng:
    + Phần mềm mã nguồn mở (Linux, MySQL): Được phát triển bởi cộng đồng, phát hành liên tục, kiểm thử rộng rãi.
    + Phần mềm mã nguồn đóng (Microsoft Windows): Được phát triển bởi công ty, có quy trình kiểm thử nghiêm ngặt trước khi phát hành.
- Tác động của vấn đề "mục tiêu di động" (Moving-target problem):
    + Khi yêu cầu thay đổi liên tục, nếu phần mềm không được thiết kế linh hoạt, chi phí sửa đổi có thể rất cao.

+ Ví dụ: Một công ty bán máy kéo mở rộng thị trường sang Canada nhưng phần mềm của họ không hỗ trợ đa tiền tệ, khiến họ phải thiết kế lại toàn bộ hệ thống.

- Phát triển phần mềm không bao giờ là một quy trình tuyến tính hoàn hảo mà luôn phải thích nghi với thay đổi.
- Mô hình lặp và gia tăng giúp giảm rủi ro, tăng khả năng phản hồi với yêu cầu mới.
- Agile và Extreme Programming có thể hiệu quả cho dự án nhỏ nhưng cần đánh giá kỹ khi áp dụng cho hệ thống lớn.
- Hiểu rõ các mô hình vòng đời giúp doanh nghiệp lựa chọn phương pháp phù hợp để tối ưu chi phí và hiệu suất.