Review sách chapter 15: Implementation

Chương này tập trung vào quy trình triển khai (implementation) trong phát triển phần mềm, nhấn mạnh các khía cạnh liên quan đến việc chuyển đổi thiết kế chi tiết thành mã nguồn, thử nghiệm và tích hợp

I. Nội dung:

1. Lựa chọn ngôn ngữ lập trình:

- Việc chọn ngôn ngữ lập trình thường bị ràng buộc bởi yêu cầu của khách hàng hoặc hạn chế phần cứng. Ví dụ, nếu chỉ có trình biên dịch COBOL, thì không có lựa chọn nào khác ngoài việc sử dụng COBOL.
- Khi được yêu cầu chọn ngôn ngữ "phù hợp nhất", quyết định nên dựa trên phân tích chi phí-lợi ích hoặc phân tích rủi ro. Ví dụ, một công ty có nhiều kinh nghiệm với COBOL có thể gặp rủi ro lớn về tài chính và tinh thần nhân viên nếu chuyển sang Java.
- COBOL phù hợp cho ứng dụng xử lý dữ liệu, nhưng không phù hợp cho các ứng dụng như trí tuệ nhân tạo (AI) hoặc phần mềm nhúng thời gian thực. Các ngôn ngữ như Java hoặc C++ được ưu tiên cho các ứng dụng truyền thông hoặc hệ thống.

2. Ngôn ngữ thế hệ thứ tư (4GLs):

- 4GLs (như Focus, Natural) được thiết kế để tăng năng suất bằng cách giảm số lượng mã cần viết (mỗi câu lệnh 4GL có thể tương đương 30-50 câu lệnh mã máy).
- 4GLs thường không thủ tục (nonprocedural), giúp lập trình dễ dàng hơn, nhưng không phải 4GL nào cũng phù hợp cho mọi sản phẩm. Việc chọn sai 4GL hoặc sử dụng môi trường CASE không phù hợp có thể dẫn đến thất bại.
- Lập trình bởi người dùng cuối (end-user programming) với 4GL có thể nguy hiểm nếu người dùng thiếu kinh nghiệm, dẫn đến lỗi mã hoặc hỏng dữ liệu.

3. Thực hành lập trình tốt:

- Tên biến nhất quán và có ý nghĩa: Tên biến cần dễ hiểu cho các lập trình viên bảo trì trong tương lai, tránh sử dụng các tên chỉ có ý nghĩa với người viết mã ban đầu (ví dụ: sử dụng frequencyAverage thay vì avgFreq).
- Mã tự ghi chú (self-documenting code): Mã cần có chú thích mở đầu (prologue comments) và chú thích trong dòng (inline comments) để hỗ trợ bảo trì. Tự ghi chú không nên là cái cớ để bỏ qua chú thích.
- Sử dụng tham số: Các hằng số nên được định nghĩa là tham số (ví dụ: const float salesTaxRate = 6.0;) để dễ dàng thay đổi mà không cần sửa nhiều chỗ trong mã.
- Bố cục mã: Sử dụng thụt đầu dòng (indentation) và khoảng trắng để tăng tính dễ đọc.
- Tránh lồng if quá sâu: Các câu lệnh if lồng nhau phức tạp nên được đơn giản hóa để dễ hiểu và bảo trì.

4. Kiểm thử và tích hợp:

- Kiểm thử đơn vị (unit testing): Bao gồm kiểm thử hộp đen (black-box), hộp trắng (glass-box), và không dựa trên thực thi (non-execution-based). Các công cụ như JUnit, CppUnit hỗ trợ kiểm thử tự động.

- Kiểm thử tích hợp (integration testing): Bao gồm các phương pháp top-down, bottom-up, và sandwich integration. Công cụ như SilkTest, IBM Rational Functional Tester hỗ trợ.
- Kiểm thử sản phẩm và chấp nhận (product and acceptance testing): Kiểm thử sản phẩm đảm bảo sản phẩm không lỗi, trong khi kiểm thử chấp nhận sử dụng dữ liệu thực để xác minh sản phẩm đáp ứng yêu cầu khách hàng.
- Nhóm đảm bảo chất lượng phần mềm (SQA) đóng vai trò quan trọng trong việc đảm bảo sản phẩm vượt qua kiểm thử chấp nhận.

5. Công cụ CASE và quản lý quy trình:

- Các công cụ CASE (như PVCS, Subversion) hỗ trợ kiểm soát phiên bản và tích hợp. Môi trường phát triển tích hợp (IDE) như Eclipse cung cấp giao diện người dùng nhất quán và tích hợp công cụ.
- Môi trường dựa trên kỹ thuật (technique-based environments) như IBM Rational Rose hỗ trợ các phương pháp phát triển cụ thể (ví dụ: UML).
- Việc chọn môi trường CASE không phù hợp hoặc sử dụng ở tổ chức có mức độ trưởng thành thấp (CMM level 1 hoặc 2) có thể dẫn đến hỗn loạn.

6. Thách thức của triển khai:

- Tái sử dụng mã (code reuse) cần được lên kế hoạch từ các giai đoạn yêu cầu và thiết kế, không chỉ trong giai đoạn triển khai.
- Quản lý tích hợp và kiểm thử là yếu tố quyết định thành công, bao gồm việc sử dụng công cụ CASE phù hợp, lập kế hoạch kiểm thử sớm, và truyền đạt thay đổi thiết kế kịp thời.

II. <u>Bài học:</u>

- Lựa chọn ngôn ngữ lập trình cần cân nhắc kỹ lưỡng
- 4GLs mang lại năng suất cao nhưng cần chọn đúng
- Thực hành lập trình tốt là nền tảng cho bảo trì
- Kiểm thử toàn diện là bắt buộc
- Công cụ CASE cần phù hợp với quy trình và mức độ trưởng thành
- Tái sử dụng mã cần được lên kế hoạch từ đầu
- Quản lý tích hợp và truyền thông là yếu tố then chốt