REVIEW SÁCH CHƯƠNG 5: THE TOOLS OF THE TRADE

Chương này trình bày các công cụ lý thuyết và phần mềm giúp cải thiện quy trình phát triển phần mềm.

1. Công cụ lý thuyết (Analytical Tools)

- **Stepwise Refinement**: Chia nhỏ vấn đề và giải quyết từng bước, tập trung vào các yếu tố quan trọng trước khi xử lý chi tiết.
- **Cost-Benefit Analysis**: Đánh giá chi phí và lợi ích để ra quyết định tối ưu.
- **Divide-and-Conquer**: Chia một bài toán lớn thành các bài toán con dễ xử lý hơn.
- **Separation of Concerns**: Tách riêng các chức năng để giảm sự phụ thuộc giữa các thành phần phần mềm.
- **Software Metrics**: Đo lường chất lượng phần mềm thông qua kích thước, chi phí, thời gian, công sức và số lỗi.

2. Công cụ phần mềm (Software Tools - CASE Tools)

- **UpperCASE & LowerCASE**: Công cụ hỗ trợ phân tích, thiết kế (UpperCASE) và lập trình, kiểm thử (LowerCASE).
- **Version Control Tools**: Quản lý các phiên bản phần mềm, giúp theo dõi thay đổi (Git, Subversion).
- **Configuration Control Tools**: Kiểm soát các phiên bản khác nhau của phần mềm, tránh xung đột khi làm việc nhóm.
- **Build Tools**: Tự động hóa quá trình biên dịch, liên kết mã nguồn (make, Ant).

3. Lợi ích của CASE Tools

- Tăng năng suất làm việc, giảm lỗi phần mềm.
- Hỗ trợ làm việc nhóm và quản lý dự án hiệu quả hơn.
- Giảm thời gian phát triển phần mềm.
- Cần đào tạo để sử dụng hiệu quả.

NHỮNG BÀI HỌC RÚT RA

1. **Tư duy chia nhỏ vấn đề rất quan trọng**
    - **Stepwise Refinement** và **Divide-and-Conquer** giúp giải quyết vấn đề phức tạp một cách có hệ thống.
    - Tránh xử lý quá nhiều chi tiết ngay từ đầu, hãy tập trung vào những yếu tố cốt lõi trước.
2. **Cần đánh giá chi phí - lợi ích trước khi ra quyết định**
    - Áp dụng **Cost-Benefit Analysis** để đưa ra quyết định tối ưu, tránh lãng phí tài nguyên.
    - Không phải lúc nào cũng nên sử dụng công nghệ mới nếu lợi ích mang lại không đáng kể.
3. **Phân tách các thành phần giúp phần mềm dễ bảo trì hơn**
    - **Separation of Concerns** giúp tái sử dụng mã nguồn, giảm lỗi khi nâng cấp phần mềm.
    - Sử dụng nguyên tắc **high cohesion - low coupling** để thiết kế hệ thống hiệu quả hơn.
4. **Đo lường là chìa khóa để cải thiện chất lượng phần mềm**
    - Sử dụng **Software Metrics** để theo dõi hiệu suất và chất lượng phần mềm.
    - Chỉ số như LOC, số lỗi, thời gian phát triển giúp đánh giá tiến độ và hiệu quả làm việc.
5. **Công cụ quản lý phiên bản là bắt buộc khi làm việc nhóm**
    - **Version Control Tools** như Git giúp kiểm soát thay đổi mã nguồn, tránh ghi đè mất dữ liệu.
    - **Configuration Control Tools** giúp theo dõi các phiên bản phần mềm và khắc phục lỗi dễ dàng hơn.
6. **Tự động hóa giúp tiết kiệm thời gian và công sức**
    - **Build Tools** như make, Ant giúp tự động hóa việc biên dịch, giảm lỗi khi build phần mềm.
    - CASE Tools giúp tăng năng suất nhưng cần được sử dụng đúng cách.

7. **Hiệu quả của CASE Tools không chỉ nằm ở năng suất**
    - Ngoài tăng tốc độ phát triển, CASE Tools còn giúp cải thiện chất lượng phần mềm, giảm lỗi và nâng cao khả năng bảo trì.
    - Tuy nhiên, cần có đào tạo bài bản để sử dụng chúng đúng cách.