HUTECH
Đại học Công nghệ Tp.HCM

KHOA CÔNG NGHỆ THÔNG TIN

CÔNG NGHỆ PHẦN MỀM

Bài 6:

TÁC VỤ XÂY DỰNG CHƯƠNG TRÌNH

Thời gian: 3 tiết

Giảng viên: ThS. Dương Thành Phết

Email: phetcm@gmail.com

Website: http://www.thayphet.net

Tel: 0918158670

facebook.com/DuongThanhPhet

NỘI DUNG

1. Tổng quan
2. Môi trường lập trình
3. Phong cách lập trình
4. Đánh giá chất lượng công việc
5. Ví dụ minh họa

6.1. TỔNG QUAN

- ✓ Trong quá trình thiết kế, ta đã chuẩn bị đầy đủ kiến trúc hệ thống cùng những thành phần thiết kế để từ đó có thể hiện thực được hệ thống dưới dạng các thành phần như mã nguồn, kịch bản, tập tin nhị phân, tập tin thực thi, thư viện, bảng, dữ liệu …
- ✓ Hơn nữa, việc xây dựng một chương trình có chất lượng tốt sẽ phản ánh những quyết định của thiết kế. *(Xem thêm Phụ lục C – Phần D)*
- ✓ Mục tiêu của bước xây dựng chương trình này là bổ sung thêm các thành phần giúp kiến trúc và hệ thống trở thành một khối hoàn chỉnh, cụ thể là:
- ✓ *Lên kế hoạch tăng cường tích hợp hệ thống (system integration) trong mỗi bước phát triển lặp lại*. Điều này có nghĩa là một hệ thống được cài đặt bởi một dãy các bước nhỏ liên tiếp và có thể quản lý được.

6.1. TỔNG QUAN

- ✓ Triển khai hệ thống bằng cách ánh xạ các thành phần thực thi được vào các nút trong mô hình triển khai. Công việc này chủ yếu dựa vào các lớp động được tìm thấy trong quá trình thiết kế.
- ✓ *Xây dựng các lớp thiết kế và những hệ thống con đã tìm được trong quá trình thiết kế*. Đặc biệt, các lớp thiết kế được xây dựng thành những thành phần dạng tập tin mã nguồn.
- ✓ Kiểm thử đơn vị các thành phần, rồi tích hợp chúng bằng các biên dịch và liên kết chúng lại với nhau thành một hoặc nhiều thành phần thực thi được, sau đó kiểm thử tích hợp và kiểm thử hệ thống.

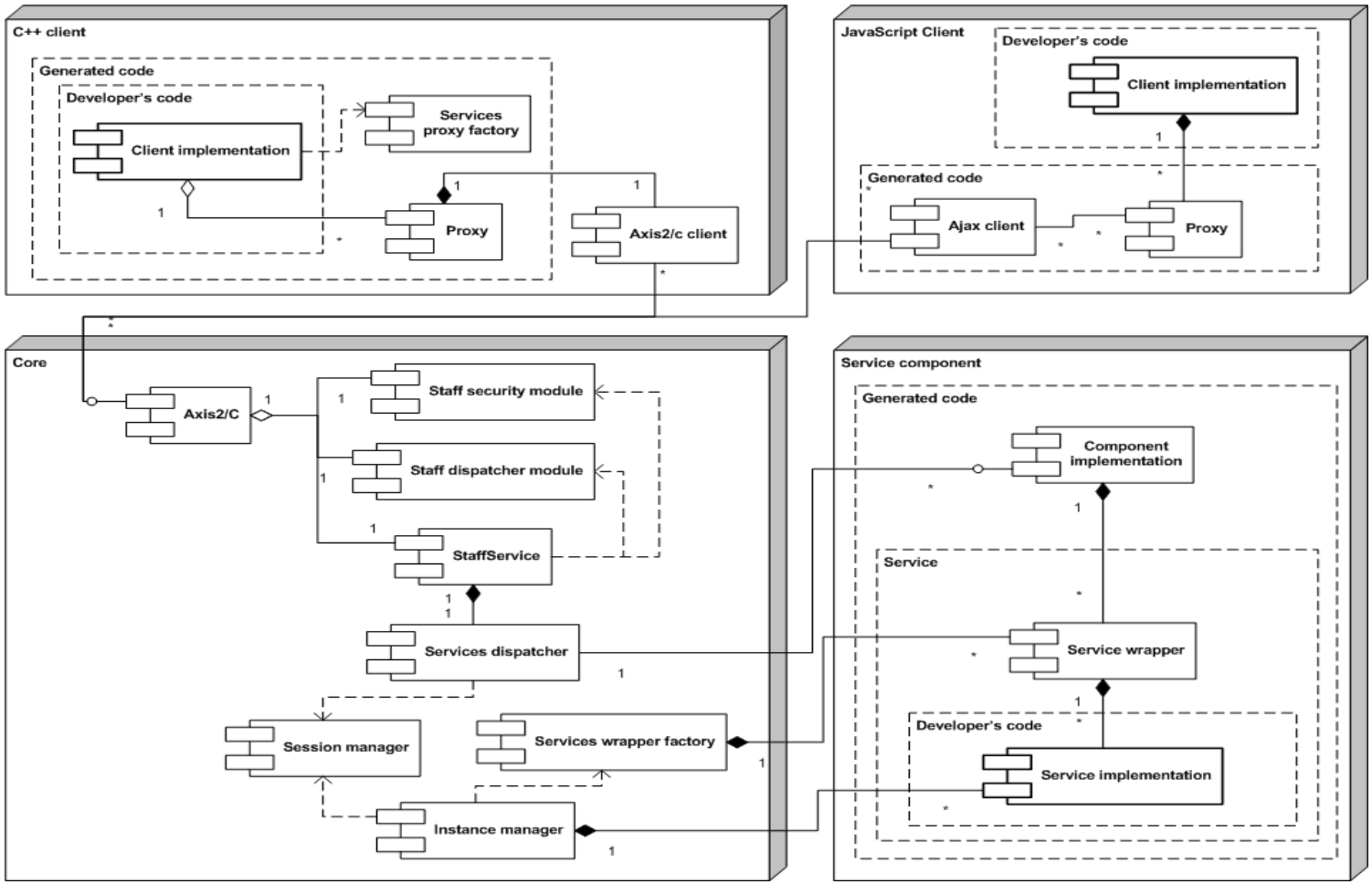
6.1. TỔNG QUAN

Việc xây dựng chương trình cần theo các tiêu chí sau:

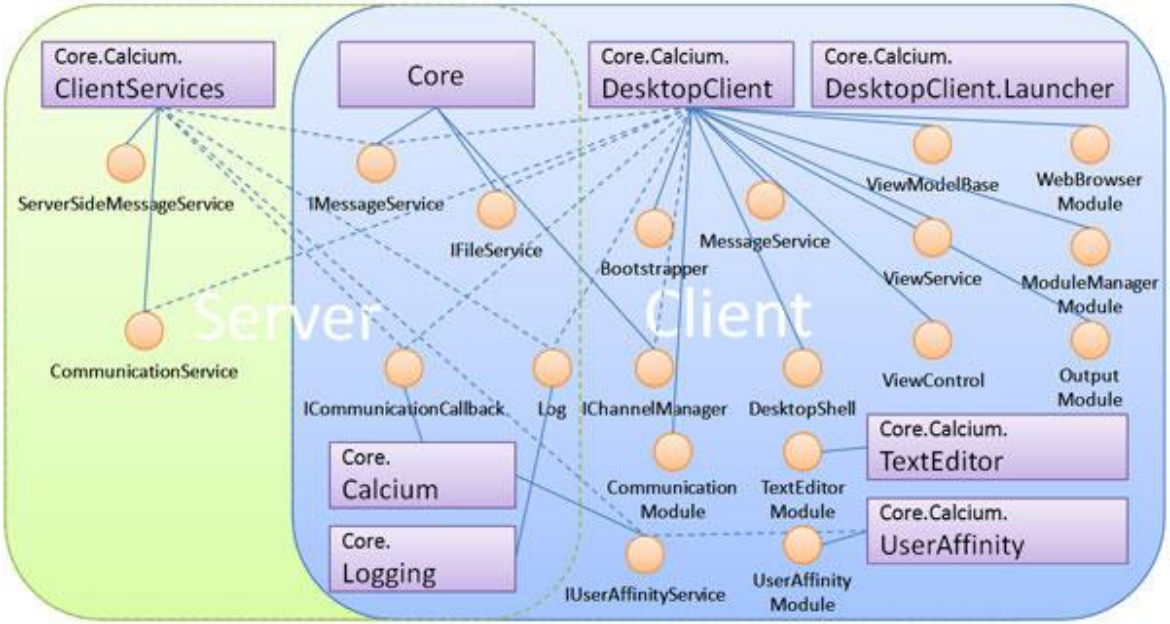
- ✓ Cấu trúc chương trình, CTDL cùng những định nghĩa được chọn lựa và thiết lập trong suốt quá trình thiết kế thủ tục cần được tổ chức tốt để dễ dàng nhận biết trong quá trình xây dựng chương trình,
- ✓ Mức trừu tượng của thiết kế về lớp đối tượng, mô-đun, thuật toán, cấu trúc dữ liệu, kiểu dữ liệu cũng phải linh động trong quá trình thực hiện,
- ✓ Giao diện giữa các thành phần của hệ thống phần mềm được mô tả rõ ràng trong thực hiện,
- ✓ Quá trình thực hiện cũng được kiểm tra độ tin cậy của đối tượng và thao tác với trình biên dịch (trước khi kiểm tra chương trình thực sự).
- ✓ Đảm bảo những đặc trưng ở trên phụ thuộc vào việc chọn lựa ngôn ngữ thực hiện và kiểu lập trình

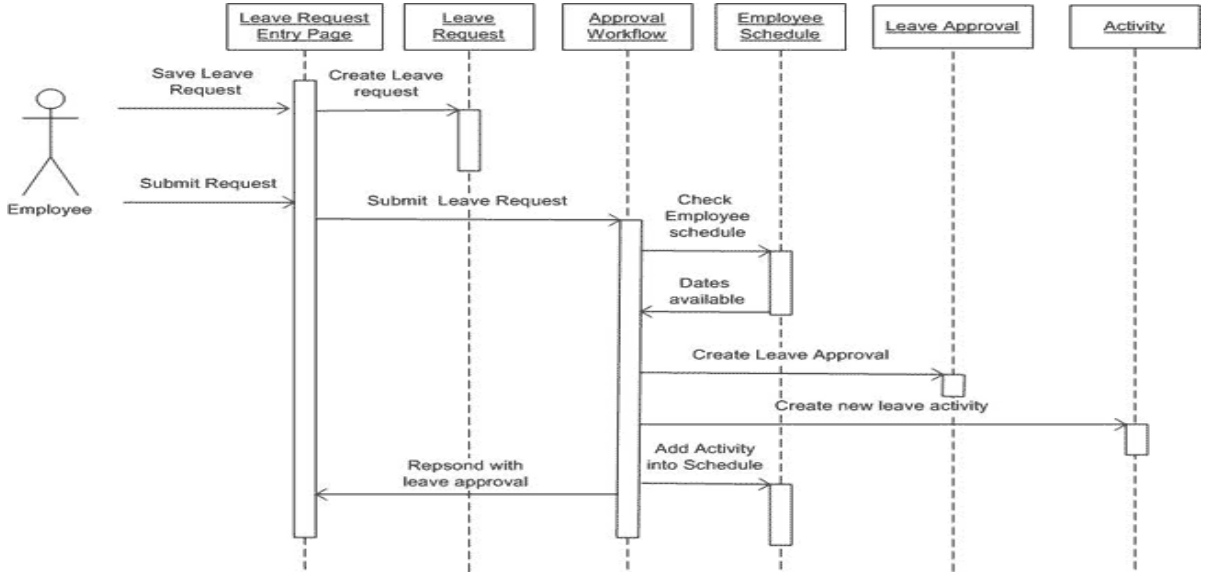
6.1. TỔNG QUAN

Ví dụ minh họa về cấu trúc thành phần của phần mềm cần hiện thực



6.1. TỔNG QUAN


Core.Calcium.
ClientServices
Core
Core.Calcium.
DesktopClient
Core.Calcium.
DesktopClient.Launcher
ServerSideMessageService
IMessageService
IFileService
Bootstrapper
MessageService
ViewModelBase
WebBrowser
Module
ViewService
ModuleManager
Module
Server
Client
CommunicationService
ICommunicationCallback
Log
IChannelManager
DesktopShell
ViewControl
Output
Module
Core.Calcium.
TextEditor
Core.
Calcium
Communication
Module
TextEditor
Module
Core.Calcium.
UserAffinity
Core.
Logging
IUserAffinityService
UserAffinity
Module


Leave Request
Entry Page
Leave
Request
Approval
Workflow
Employee
Schedule
Leave Approval
Activity
Save Leave
Request
Create Leave
request
Employee
Submit Request
Submit Leave Request
Check
Employee
schedule
Dates
available
Create Leave Approval
Create new leave activity
Add Activity
into Schedule
Repsond with
leave approval

6.2. MÔI TRƯỜNG LẬP TRÌNH

- ✓ Việc chọn lựa “đúng” ngôn ngữ lập trình cho quá trình hiện thực hóa là vấn đề quan trọng trong quy trình lập trình.
- ✓ Mức lý tưởng, việc thiết kế nên được thực hiện độc lập và tránh bất kỳ liên quan nào đến dạng ngôn ngữ sẽ được dùng khi hiện thực sau đó, nhằm tạo ra bản thiết kế có thể thực hiện được trên ngôn ngữ lập trình bất kỳ.
- ✓ Việc chọn được ngôn ngữ lập trình phù hợp cho quá trình xây dựng phần mềm cần được căn cứ trên các tiêu chí sau:

6.2.1. TIÊU CHÍ VỀ CHẤT LƯỢNG CỦA NNLT

Tiêu chí chất lượng này được thể hiện qua những yếu tố:

Tính mô-đun hóa	Luồng điều khiển	Khả năng tích hợp
Giá trị của tài liệu	Tính hiệu quả Hỗ trợ hộp thoại	Tính khả chuyển
Cấu trúc dữ liệu	Yếu tố ngôn ngữ chuyên biệt	

6.2.2. TIÊU CHÍ VỀ KHẢ NĂNG MÔ-ĐUN HÓA CỦA NNLT

- ✓ Tiêu chí này nhấn mạnh về khả năng mô-đun hóa, tức là mức độ hỗ trợ việc phân chia chương trình thành những thành phần độc lập nhỏ hơn.
- ✓ Việc phác thảo một chương trình lớn thành nhiều mô-đun là điều kiện tiên quyết để thực thi trong dự án phần mềm.
- ✓ Không có khả năng mô-đun hóa thì việc phân chia công việc trong giai đoạn thực hiện trở nên không khả thi.
- ✓ Những chương trình đơn nhất trở nên không thể quản lý vì chúng khó có thể bảo trì và tài liệu kỹ thuật và chúng thực hiện với thời gian biên dịch dài.
- ✓ Nếu một ngôn ngữ lập trình hỗ trợ phát thảo một chương trình thành những phần nhỏ, chúng phải đảm bảo những thành phần phải hoạt động với nhau.
- ✓ Nếu một thủ tục được thực thi ở mô-đun khác, cũng được kiểm tra để biết thủ tục đó có thực sự tồn tại và nó có được sử dụng chính xác hay không (nghĩa là số tham số và kiểu dữ liệu là

6.2.2. TIÊU CHÍ VỀ KHẢ NĂNG MÔ-ĐUN HÓA CỦA NNLT

- ✓ Tiêu chí này nhấn mạnh về khả năng có thể đọc và bảo trì của chương trình thông qua tài liệu kỹ thuật.
- ✓ Điều này càng quan trọng đối với những chương trình lớn hay phần mềm mà khách hàng vẫn tiếp tục phát triển.
- ✓ Tài liệu kỹ thuật của một chương trình mang lại giá trị cao hơn kết quả đạt được hiện hành của chương trình đó, vì tài liệu kỹ thuật sẽ được tham khảo nhiều lần và được khai thác triệt để nhằm phục vụ quá trình bảo trì và nâng cấp cho chương trình đó.
- ✓ Hơn nữa, do nhiều ngôn ngữ mở rộng với quá nhiều chức năng chuyên biệt, nếu thiếu tài liệu kỹ thuật, ta sẽ khó hiểu được tất cả chi tiết bên trong của chương trình nên có thể dẫn đến việc hiểu hay giải thích sai ý tưởng của chúng.

6.2.2. TIÊU CHÍ VỀ KHẢ NĂNG MÔ-ĐUN HÓA CỦA NNLT

- ✓ Khi hiện thực hóa chương trình, các dữ liệu phức tạp cần được xử lý hoàn chỉnh.
- ✓ Khả năng (của ngôn ngữ lập trình) sẵn sàng hỗ trợ hiện thực cấu trúc dữ liệu sẽ đóng vai trò quan trọng trong quá trình hiện thực chương trình.
- ✓ Ví dụ: họ ngôn ngữ C cho phép khai báo con trỏ đối với cấu trúc dữ liệu, điều này cho phép cấu trúc dữ liệu phức tạp, phạm vi và cấu trúc của chúng có thể thay đổi ở thời điểm thực thi, tuy nhiên không nghiêm ngặt trong truy xuất (khi so sánh với Java).

6.2.2. TIÊU CHÍ VỀ KHẢ NĂNG MÔ-ĐUN HÓA CỦA NNLT

- ✓ Trong dự án lớn với nhiều nhóm dự án, dữ liệu trừu tượng là một vấn đề trọng tâm và mang ý nghĩa cụ thể.
- ✓ Ngôn ngữ lập trình hướng đối tượng có những đặc trưng mở rộng về loại kiểu dữ liệu trừu tượng, nhằm cho phép hiện thực hoá những hệ thống phần mềm phức tạp.
- ✓ Đối với những giải pháp mở rộng và uyển chuyển, ngôn ngữ lập trình hướng đối tượng cung cấp tuỳ chọn đặc biệt tốt hơn ngôn ngữ lập trình thông thường khác.

6.2.5. VÍ DỤ

Ví dụ: Xét phần mềm quản lý thư viện:

- ✓ Ngôn ngữ lập trình chọn: VB/ C++ / C# hay Java …
- ✓ Hê QT CSDL: MS Access / SQL Server/Oracle …
- ✓ Hệ thống lớp đối tượng: tạo lập lớp đối tượng (THU_VIEN, DOC_GIA, SACH) theo mô tả của phần thiết kế trong môi trường cụ thể nào đó
- ✓ Hệ thống giao diện: Tạo các giao diện (màn hình chính, màn hình lập thẻ, …) theo mô tả của phần thiết kế trong môi trường cụ thể nào đó,
- ✓ Hệ thống lưu trữ: tạo lập cấu trúc cơ sở dữ liệu (các bảng THU_VIEN, DOC_GIA, SACH, MUON_SACH) theo mô tả của phần thiết kế trong môi trường cụ thể nào đó

6.3. PHONG CÁCH LẬP TRÌNH

- ✓ Sau khi hiện thực và kiểm tra chương trình, hệ thống phần mềm hiếm khi được sử dụng trong một thời gian dài mà không có sửa đổi hay điều chỉnh.
- ✓ Điều này thực sự đúng vì khi một yêu cầu của phần mềm được cập nhật hoặc mở rộng (sau khi hoàn chỉnh sản phẩm hay trong suốt quá trình thực hiện thao tác), ta thường khó phát hiện ra lỗi hay những thiếu sót phát sinh.
- ✓ Vì vậy, giai đoạn hiện thực phần mềm chắc chắn phải được sửa đổi và mở rộng, đòi hỏi lặp lại việc đọc hiểu chương trình nguồn nhiều lần.
- ✓ Trong trường hợp lý tưởng, chức năng của một thành phần chương trình được tìm hiểu chỉ dựa trên chương trình nguồn mà không được cung cấp thông tin nào từ tài liệu thiết kế.

6.3. PHONG CÁCH LẬP TRÌNH

- ✓ Tuy nhiên, chương trình nguồn chỉ là một dạng tài liệu phản ánh hiện trạng của thực thi.
- ✓ Khả năng "đọc được" của một chương trình (tức là ta có thể đọc hiểu được mã nguồn của chương trình đó) phụ thuộc vào ngôn ngữ lập trình được dùng và phong cách lập trình của người thực hiện.
- ✓ Việc viết một chương trình để có thể đọc được là một tiến trình sáng tạo.
- ✓ Phong cách lập trình của người thực hiện sẽ ánh hưởng đến khả năng đọc được của chương trình hơn là ngôn ngữ lập trình được sử dụng.
- ✓ Các yếu tố quan trọng nhất của phong cách lập trình tốt là:

6.3.1. TÍNH CẤU TRÚC

Tính chất này thể hiện bằng việc:

- ✓ Phân rã một hệ thống phần mềm dựa trên độ phức tạp của nó thông qua mức trừu tượng của thành phần, tạo cấu trúc chương trình lớn,
- ✓ Hoặc chọn lựa những thành phần chương trình phù hợp trong việc định ra những thuật toán của thủ tục con, tạo cấu trúc chương trình nhỏ.

6.3.2. ƯU ĐIỂM CỦA DIỄN ĐẠT

- ✓ Quy trình hiện thực một hệ thống phần mềm bao gồm việc đặt tên đối tượng và mô tả các công việc thực thi của đối tượng này.
- ✓ Việc chọn lựa tên sẽ đặc biệt quan trọng khi viết thuật toán.

6.3.2. ƯU ĐIỂM CỦA DIỄN ĐẠT

Một số đề nghị

- ✓ Với một hệ thống, ta gán tên chỉ dựa trên một ngôn ngữ (ví dụ đừng dùng lẫn lộn tiếng Anh và tiếng Việt).
- ✓ Nếu dùng chữ viết tắt, ta nên sử dụng tên đặt này để giúp người đọc chương trình có thể hiểu mà không cần bất cứ sự giải thích nào. Việc sử dụng những từ viết tắt chỉ bao gồm ngữ cảnh.
- ✓ Dùng chữ hoa và chữ thường để phân biệt các loại định nghĩa khác nhau (như chữ hoa đầu tiên cho kiểu dữ liệu, lớp, mô-đun, chữ thường đầu tiên cho biến), đặt tên dài hơn để dễ hiểu (như CheckInputValue).
- ✓ Dùng danh từ cho giá trị, động từ cho hoạt động, và thuộc tính cho điều kiện để làm rõ ý nghĩa nhận diện (như width, ReadKey, valid).
- ✓ Thiết lập những qui luật để sử dụng một cách thích hợp.

6.3.2. ƯU ĐIỂM CỦA DIỄN ĐẠT

- ✓ Ghi chú rõ ràng và đầy đủ các diễn giải sử dụng, nhằm đóng góp cho khả năng đọc được của chương trình.
- ✓ Hiệu chỉnh việc ghi chú chương trình không dễ dàng và đòi hỏi kinh nghiệm, sáng tạo và khả năng diễn đạt thông điệp gọn gàng và chính xác.

6.3.2. ƯU ĐIỂM CỦA DIỄN ĐẠT

Một số luật cho cách tạo ghi chú

- ✓ Mỗi thành phần hệ thống (mô-đun hay lớp) nên bắt đầu với ghi chú chi tiết để cung cấp cho người đọc những thông tin liên quan đến thành phần của hệ thống như:
- ✓ Thành phần này làm gì?
- ✓ Thành phần này được sử dụng ra sao trong những ngữ cảnh gì?
- ✓ Những phương thức đặc biệt được sử dụng.
- ✓ Ai là tác giả của thành phần này?
- ✓ Thành phần này được viết khi nào?
- ✓ Những sửa đổi cập nhật nó được thực hiện.

6.3.2. ƯU ĐIỂM CỦA DIỄN ĐẠT

- ✓ Mỗi thủ tục và phương thức nên được cung cấp các ghi chú mô tả công việc. Điều này cần đặc biệt phần mềm cho phần đặc tả giao diện.
- ✓ Ghi chú để giải thích ý nghĩa của biến và hằng.
- ✓ Gán nhãn và ghi chú cho các thành phần có những tác nhiệm riêng.
- ✓ Những khối lệnh khó hiểu (hoặc thành phần hay thủ tục rắc rối) nên được mô tả ghi chú sao cho người đọc dễ dàng hiểu chúng.

6.3.2. ƯU ĐIỂM CỦA DIỄN ĐẠT

- ✓ Phản ánh trong ghi chú những cập nhật về các thay đổi của chương trình liên quan cả phần khai báo và những khối lệnh thành phần.
- ✓ Việc tuân thủ những luật trên cần được cân nhắc vì không có luật áp dụng đồng nhất cho tất cả các hệ thống phần mềm và mỗi phạm vi phần mềm. Việc tạo ghi chú cho hệ thống phần mềm là một nghệ thuật cũng giống như phần thiết kế cài đặt hệ thống phần mềm.
- ✓ Ngoài việc chọn tên và ghi chú, khả năng đọc của phần mềm cũng phụ thuộc vào cách thức trình bày bên ngoài.

6.3.3. CÁCH THỨC TRÌNH BÀY BÊN NGOÀI

- ✓ Một số luật cho hình thức trình bày chương trình:
  - Mỗi thành phần của chương trình, những khai báo (kiểu dữ liệu, hằng biến …) nên được tách biệt mỗi phần của khối lệnh.
  - Phần khai báo nên có một cấu trúc đồng nhất khi có thể như thứ tự sau: hằng, kiểu dữ liệu, lớp, mô-đun, phương thức và thủ tục.
  - Mô tả giao diện (danh sách tham số cho phương thức và thủ tục) nên tách tham số nhập liệu, kết xuất và nhập/xuất.
  - Phần ghi chú và chương trình nguồn nên tách biệt.
- ✓ Cấu trúc của chương trình nên được nhấn mạnh ở phần canh chỉnh lề (sử dụng phím tab cho từ mỗi đầu khối lệnh đến khối lệnh theo sau)

6.4. ĐÁNH GIÁ CHẤT LƯỢNG CÔNG VIỆC

6.4.1 Hiện thực tăng cường

6.4.2 Đánh giá lại thiết kế và chương trình

6.4.1 HIỆN THỰC TĂNG CƯỜNG

- ✓ Ý tưởng cơ bản của việc hiện thực tăng cường gần với việc kết hợp giai đoạn thiết kế và hiện thực hơn là tách biệt hai giai đoạn này, như mô hình quy trình phát triển tuần tự cổ điển đề ra.
- ✓ Phương pháp này cho thấy những quyết định trong thiết kế và cài đặt có tác động lẫn nhau, nếu ta tách biệt thiết kế rời rạc thì sẽ không đạt được mục tiêu tăng cao chất lượng công việc.
- ✓ Việc hiện thực tăng cường nghĩa là sau mỗi bước thiết kế có liên quan đến kiến trúc, thì kiến trúc phần mềm hiện hành được thẩm định dựa trên những trường hợp thực tế.

6.4.1 HIỆN THỰC TĂNG CƯỜNG

- ✓ Theo đó, sự tác động qua lại giữa các thành phần trong hệ thống cụ thể – trong thiết kế và trong hình thức đặc tả giao diện – cần được thẩm định.
- ✓ Để có thể làm được điều này, những thành phần hệ thống (với hành vi xuất/nhập) được mô phỏng hay thực tế hóa như khuôn mẫu.
- ✓ Nếu có những nghi ngờ liên quan đến tính khả thi của thành phần thì tiến trình thiết kế được ngắt và những thành phần được thực hiện.
- ✓ Chỉ khi hiện thực và nhúng chúng vào trong kiến trúc hệ thống (đã được kiểm tra trước đó) thì tiến trình thiết kế mới được tiếp tục, hoặc kiến trúc được chấp nhận tương ứng với kiến thức thu được trong khi hiện thực thành phần.

6.4.1 HIỆN THỰC TĂNG CƯỜNG

- ✓ Hiệu quả của phương pháp này phụ thuộc vào việc mở rộng khả năng tích hợp những thành phần hệ thống (được hoàn chỉnh theo chuẩn khác nhau ở cấp độ khác nhau) đối với toàn bộ hệ thống để hiện thực gần với thực tế.
- ✓ Vài thành phần hệ thống (như giao diện người dùng và mô hình dữ liệu) được thể hiện dưới dạng khuôn mẫu, còn các thành phần khác (thư viện có sẵn, hay tồn tại như các hiện thực hoàn chỉnh) được thể hiện dưới dạng mã nguồn thực thi, còn các thành phần hệ thống khác có sẵn được thể hiện như đặc tả giao diện.
- ✓ Đối với sự hợp lệ của thiết kế hệ thống hiện hành, khi giao diện người dùng được triển khai thì tương ứng khuôn mẫu cũng cần được kích hoạt.

6.4.2 ĐÁNH GIÁ LẠI THIẾT KẾ VÀ CHƯƠNG TRÌNH

- ✓ Việc xem lại (*review*) thiết kế và chương trình sẽ giúp ta hoàn chỉnh chất lượng hiệu quả của công việc hơn là chỉ điều chỉnh những thay đổi đơn lẻ trong quá trình phát triển phần mềm.
- ✓ Trong các phần mềm lớn, vấn đề quan trọng là nhu cầu xem xét lại những yêu cầu, đặc tả, thiết kế, và cả chương trình của ta, để từ đó giúp ta điều chỉnh thiếu sót, luận lý, cấu trúc, tính sáng tỏ.
- ✓ Khi chương trình không rõ hay mơ hồ xáo trộn, việc thêm ghi chú hay viết lại nó một cách đơn giản hơn sẽ làm cho chương trình dễ đọc và dễ hiểu.

6.4.2 ĐÁNH GIÁ LẠI THIẾT KẾ VÀ CHƯƠNG TRÌNH

- ✓ Mục đích của việc xem lại là để đảm bảo chương trình tạo ra đạt chất lượng cao nhất.
- ✓ Một số dạng xem lại là thao tác kiểm duyệt, duyệt qua, xem xét mục riêng từ thiết kế đến từng dòng lệnh.
- ✓ Việc xem lại có thể được dùng trên yêu cầu, thiết kế, hồ sơ tài liệu hay bất kỳ yếu tố của sản phẩm.

6.4.2 ĐÁNH GIÁ LẠI THIẾT KẾ VÀ CHƯƠNG TRÌNH

- ✓ Trong thực tế, nhiều dự án phần mềm đã được triển khai đến tận giai đoạn kiểm thử nhưng thiếu công tác xem lại, điều này không thực sự hiệu quả.
- ✓ Xem lại thiết kế và chương trình là các cách thức hiệu quả để tìm và sửa chữa thiếu sót.
- ✓ Bằng việc xem lại, ta có thể tìm ra trong chương trình những thiếu sót trực tiếp, các số lỗi mà ta cần cập nhật hoàn chỉnh và chính xác hơn.

6.4.2 ĐÁNH GIÁ LẠI THIẾT KẾ VÀ CHƯƠNG TRÌNH

- ✓ Công việc xem lại cho phép ta quay trở lại bất kỳ việc gì đã làm trước đó trong chương trình.
- ✓ Nhóm phát triển nên cùng nhau đọc lại mọi thiết kế và chương trình, rồi nghiên cứu để hiểu chúng tường tận nhằm sửa những sai sót về luận lý, cấu trúc hay tính rõ ràng,
- ✓ sau đó có thể viết lại chương trình hay thêm ghi chú và viết lại hoàn chỉnh cho những phần nội dung không rõ ràng để làm cho chúng dễ đọc và dễ hiểu hơn.

6.5. VÍ DỤ MINH HOẠ

Ví dụ: Xét phần mềm hỗ trợ giải bài tập phương trình đại số với yêu cầu là Soạn đề bài, Soạn đáp án, Giải bài tập, Chấm điểm. Các giai đoạn thực hiện trong quy trình bao gồm:

Giai đoạn 1: Xác định yêu cầu

- ✓ Yêu cầu 1: Soạn đề bài với mô tả và quy tắc
- ✓ Yêu cầu 2: Soạn đáp án với mô tả, quy tắc.
- ✓ Yêu cầu 3: Giải bài tập với mô tả, quy tắc và biểu mẫu.
- ✓ Yêu cầu 4: Chấm điểm với mô tả, quy tắc.

Giai đoạn 2: Sơ đồ luồng công việc cho 4 chức năng

Giai đoạn 3: Phân tích yêu cầu chức năng

Giai đoạn 4: Thiết kế phần mềm

6.5. VÍ DỤ MINH HOẠ

Giai đoạn 5: Thực hiện phần mềm

- ✓ Hệ thống Lớp đối tượng: Tạo các lớp đối tượng SACH_BAI_TAP, BAI_TAP theo mô tả thiết kế trong môi trường cụ thể (VB, C#, Java)
- ✓ Hệ thống giao diện: tạo (vẽ) giao diện (màn hình chính, màn hình soạn đề bài, màn hình soạn đáp án, màn hình giải bài tập, màn hình chấm điểm) theo mô tả của thiết kế trong môi trường cụ thể (VB, C#, Java),
- ✓ Hệ thống lưu trữ: tạo cấu trúc cơ sở dữ liệu (các bảng SACH_BAI_TAP, BAI_TAP, BAI_GIAI, BUOC_GIAI) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (MS Access / SQL Server, Oracle …)

Giai đoạn 6: Kiểm chứng phần mềm (xem ở bài sau)

TÓM TẮT

Các kiến thức về:

- ✓ Môi trường lập trình và các tiêu chí của ngôn ngữ lập trình (Chất lượng, Khả năng mô-đun hóa, Giá trị tài liệu kỹ thuật, Cấu trúc dữ liệu)
- ✓ Phong cách lập trình và các vấn đề liên quan (Tính cấu trúc, Ưu điểm của diễn đạt, Cách thức trình bày bên ngoài)
- ✓ Đánh giá chất lượng công việc thông qua việc Hiện thực tăng cường và Đánh giá lại thiết kế và chương trình.

BÀI TẬP

1. Phụ lục A trang 170
2. Phụ lục B trang 179