LỜI CẢM ƠN

Trước hết em xin bày tỏ tình cảm và lòng biết ơn đối với Th.S Nguyễn Thị Thanh Thoan – Bộ môn Công nghệ thông tin – Trường Đại học Dân Lập Hải Phòng, người đã dành cho em rất nhiều thời gian quý báu, trực tiếp hướng dẫn tận tình giúp đỡ, chỉ bảo em trong suốt quá trình làm đồ án tốt nghiệp.

Em xin chân thành cảm ơn tất cả các thầy cô giáo trong Bộ môn Công nghệ thông tin - Trường ĐHDL Hải Phòng, chân thành cảm ơn các thầy giáo, cô giáo tham gia giảng dạy và truyền đạt những kiến thức quý báu trong suốt thời gian em học tập tại trường, đã đọc và phản biện đồ án của em giúp em hiểu rõ hơn các vấn đề mình nghiên cứu, để em có thể hoàn thành đồ án này.

Em xin cảm ơn GS.TS.NGƯT Trần Hữu Nghị Hiệu trưởng Trường Đại học Dân lập Hải Phòng, Ban giám hiệu nhà trường, Bộ môn tin học, các Phòng ban nhà trường đã tạo điều kiện tốt nhất trong suốt thời gian học tập và làm tốt nghiệp.

Tuy có nhiều cố gắng trong quá trình học tập, trong thời gian thực tập cũng như trong quá trình làm đồ án nhưng không thể tránh khỏi những thiếu sót, em rất mong được sự góp ý quý báu của tất cả các thầy giáo, cô giáo cũng như tất cả các bạn để kết quả của em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Hải Phòng, ngày 06 tháng 07 năm 2010

Sinh viên

Lê Văn Minh

MỞ ĐẦU

Do nhu cầu của những hệ thống phần mềm tăng lên các nhà nghiên cứu cũng như những nhà thực hành giỏi luôn cố gắng tìm kiếm những phương pháp luận và kỹ thuật phát triển để làm tự động hoá việc sản xuất phần mềm và làm dễ dàng việc bảo trì chúng. Những kỹ thuật này mới đây đã bao hàm các mẫu thiết kế và khung làm việc thiết kế. Đặc biệt, chúng ta nhận ra sự cần thiết có một phương pháp luận phát triển để phát triển những hệ thống phức tạp qui mô lớn, và đồng thời học tập được những kinh nghiệm từ những nhà thiết kế các hệ thống khác trong việc giải quyết những vấn đề thiết kế diễn ra.

Hiện nay có rất nhiều phương pháp và kỹ thuật được áp dụng để nghiên cứu và phát triển các phần mềm. Nhưng để làm tự động hóa việc sản xuất phần mềm cũng như dễ dàng cho việc bảo trì những phần mềm đó, thì ý tưởng sử dụng lý thuyết để phát triển phần mềm bằng phương pháp hướng đối tượng đang mở ra một hướng đi cho việc phát triển phần mềm có quy mô lớn, và có thể sử dụng lại phần mềm đó một cách hiệu quả.

Trong đó khi nghiên cứu lý thuyết hướng đối tượng song song với việc tạo ra các phần mềm, thì các lập trình viên cũng như những người quản lý đã nhận ra được một số vấn đề có thể được giải quyết bằng một giải pháp chung nào đó. Vì vậy giải pháp tương tự nên được vận dụng. Những giải pháp tương tự được tổng quát hóa, hình thức hóa được gọi là mẫu thiết kế. Việc tạo những mẫu thiết kế là một vấn để của sự trừu tượng hóa đặc điểm giống nhau của những vấn để và giải pháp vì vậy hướng chung của giải pháp gốc có thể được áp dụng tới vấn đề mới

Đồ án sẽ giới thiệu tổng quan về phương pháp phân tích thiết kế hướng đối tượng và ngôn ngữ mô hình hóa thống nhất. Đồng thời đi sâu tìm hiểu về các mẫu thiết kế cũng như nắm bắt được mục đích sử dụng, vai trò và tác dụng của mẫu thiết kế đối với việc phát triển phần mềm. Áp dụng phương pháp phân tích thiết kế hướng mẫu kết hợp thử nghiệm ngôn ngữ lập trình C# để cài đặt chương trình demo cho việc ứng dụng các mẫu thiết kế

Đồ án bao gồm bốn chương:

- Chương 1: Ý tưởng phát triển phần mềm bằng phương pháp hướng đối tượng

Trình bày tổng quan về phương pháp phân tích thiết kế hướng đối tượng. Khái quát ý tưởng phát triển phần mềm bằng phương pháp hướng đối tượng.

- Chương 2: Phương pháp phân tích thiết kế hướng mẫu:

Trình bày các khái niệm của mẫu thiết kế, mục đích vai trò sử dụng của phương pháp phân tích thiết kế hướng mẫu, cũng như tác dụng của mẫu thiết kế. Qua đó rút ra được những vấn đề cũng như các gải pháp của phương pháp này. Đồng thời đi sâu tìm hiểu các đặc trưng, sự cấu thành của các mẫu thiết kế đối với việc phát triển phần mềm

- Chương 3: Tiến trình của phương pháp phân tích thiết kế hướng mẫu

Trong phần này sẽ trình bày chi tiết về mục đích, tiến trình, sản phẩm của từng giai đoạn. Đồng thời sẽ tìm hiểu thêm về sự sinh mã nguồn và những lợi ích, hạn chế của phương pháp phân tích thiết kế hướng mẫu

- Chương 4: Một số mẫu thiết kế

Trong phần này các mẫu thiết kế thường dùng sẽ được tìm hiểu thông qua các vấn đề được đặt ra của từng mẫu, các giải pháp, lĩnh vực áp dụng và chương trình ứng dụng minh họa cho các mẫu

Cuối cùng là phần kết luận và hướng phát triển của đề tài.

MỤC LỤC

CHƯƠNG 1: Ý TƯỞNG PHÁT TRIỂN PHẦN MỀM BẰNG PHƯƠNG PHÁP HƯỚNG ĐỐI TƯỢNG

1.1 Ý tưởng phát triển phần mềm bằng phương pháp hướng đối tượng

Ý tưởng cơ bản của tiếp cận hướng đối tượng là phát triển một hệ thống bao gồm các đối tượng độc lập tương đối với nhau và tương tác với nhau thông qua các thông báo. Mỗi đối tượng bao hàm trong nó cả dữ liệu và các xử lý tiến hành trên các dữ liệu này được gọi là bao gói thông tin. Nhờ các thông báo để thực hiện các chức năng lớn hơn các đối tượng liên kết với nhau

1.2 Các giai đoạn của chu trình phát triển phần mềm hướng đối tượng

a) Phân tích hướng đối tượng (Object Oriented Analysis - OOA)

Là giai đoạn phát triển một mô hình chính xác và súc tích của vấn đề, có thành phần là các đối tượng và khái niệm đời thực, dễ hiểu đối với người sử dụng.

b) Thiết kế hướng đối tượng (Object Oriented Design - OOD)

Là giai đoạn tổ chức chương trình thành các tập hợp đối tượng cộng tác với nhau, mỗi đối tượng trong đó là một lớp. Các lớp là thành viên tạo thành một cây cấu trúc với mối quan hệ thừa kế hay tương tác bằng thông báo.

c) Lập trình hướng đối tượng (Object Oriented Programming - OOP)

Giai đoạn xây dựng phần mềm có thể được thực hiện sử dụng kỹ thuật lập trình hướng đối tượng. Đó là phương thức thực hiện việc chuyển các thiết kế hướng đối tượng thành chương trình bằng việc sử dụng một ngôn ngữ lập trình có hỗ trợ các tính năng hướng đối tượng. Kết quả chung cuộc của giai đoạn này là một loạt các mã máy có thể chạy được, nó chỉ được đưa vào sử dụng sau khi đã trải qua nhiều vòng quay của nhiều bước thử nghiệm khác nhau.

1.3. Những vấn đề đặt ra trong phân tích thiết kế hướng đối tượng

Đặc điểm của phân tích và thiết kế hướng đối tượng là nhìn nhận hệ thống như một tập các đối tượng tương tác với nhau để tạo ra một hành động cho một kết quả ở mức cao hơn. Để thực hiện được điều này người ta phải sử dụng hệ thống mô hình các đối tượng với các đặc trưng cơ bản sau:

- Tính trừu tượng hoá cao

- Tính bao gói thông tin

- Tính mô đun hoá

- Tính kế thừa

Ngày nay, UML là một công cụ được thiết kế có tất cả những tính chất và điều kiện giúp ta xây dựng được các mô hình đối tượng có được 4 đặc trưng trên

Quá trình phát triển gồm nhiều bước lặp mà một bước lặp bao gồm: xác định yêu cầu của hệ thống, phân tích, thiết kế, triển khai và kiểm thử. Trong đó thì hoạt động phân tích và thiết kế đặt ra 3 vấn đề lớn:

- Làm cách nào để xác định được các lớp đối tượng từ hệ thống thực. Các đối tượng ở đây là sự trừu tượng hoá các bộ phận tham gia vào thực hiện các chức năng. Ngoại trừ một số đối tượng gắn với thực thể dữ liệu, việc nhận ra các lớp đối tượng khác tham gia thực hiện ca sử dụng không phải là đơn giản. Hoạt động này phụ thuộc rất lớn vào sự cảm nhận, khả năng phân tích và kinh nghiệm của các nhà phân tích.

- Đa số trường hợp có nhiều lớp đối tượng tương tác với nhau để thực hiện một hành vi lớn hơn. Vậy phải phân công trách nhiệm giữa các đối tượng như thế nào là hợp lý, không để một lớp phải thực hiện quá nhiều, lớp khác thực hiện quá ít làm ảnh hưởng đến chất lượng và thời gian thực hiện hành vi chung

- Việc phân nhỏ các lớp đến đâu là vừa phải. Về nguyên tắc thì việc phân càng nhỏ các lớp cho phép ta dễ nắm bắt và hiểu công việc để thiết kế và dễ quản lý các lớp đối tượng. Nhưng khi tăng số lượng các đối tượng thì số lượng mối quan hệ giữa chúng cũng tăng lên một cách đáng kể và dẫn đến việc làm tăng thời gian tương tác.

CHƯƠNG 2: PHƯƠNG PHÁP PHÂN TÍCH THIẾT KẾ HƯỚNG MẪU

2.1. Mẫu thiết kế

Những người phát triển giàu kinh nghiệm nhận thấy rằng khi họ cố gắng giải quyết một vấn đề mới. Tình huống thường có là một số cái có chung một giải pháp như họ đã tạo ra hoặc được nhìn thấy. Những vấn đề có thể không giống y hệt và giải pháp y hệt sẽ giải quyết một vấn đề mới nhưng những vấn đề là tương tự, vì vậy giải pháp tương tự nên được vận dụng. Giải pháp tương tự được tổng quát hóa, hình thức hóa được gọi là mẫu thiết kế. Việc tạo những mẫu thiết kế là một vấn để của sự trừu tượng đặc điểm giống nhau của những vấn để và giải pháp vì vậy hướng(khía cạnh) chung của giải pháp gốc có thể được áp dụng tới vấn đề mới.

Hai vấn đề cơ bản liên quan được kết hợp với những mẫu thiết kế, đầu tiên phải làm với ứng dụng của mẫu thiết kế. Vấn đề của việc nhận ra bản chất của vấn đề và kiểm thử mẫu thư viện cho những cái tốt nhất gọi là pattern hatching. Và như John Vlissides tác giả của cuốn sách đã chỉ ra"Chúng ta sẽ không tạo một vài thứ mới nhưng phát triển từ những nguyên lí cơ bản tồn tại từ trước.". Những nguyên lí cơ bản tồn tại từ trước là những mẫu thiết kế đạt được của chúng tôi và chúng tôi có thể sử dụng để xây dựng giải pháp làm việc trong những tình huống mới.

Những kết quả khác, tất nhiên là sự nhận dạng và nắm bắt được mẫu thiết kế mới để thêm vào thư viện. Và quá trình này gọi là quá trình khai phá mẫu(pattern mining). Nó liên quan tới sự trừu tượng hóa của vấn đề tới những thuộc tính bản chất của nó, việc tạo một giải pháp chung và sau đó hiểu kết quả của giải pháp trong ngữ cảnh vấn đề trong đó mẫu được áp dụng.

Những mẫu thiết kế không chỉ là phần mềm dùng lại mà hơn thế nữa là dùng lại khái niệm. Hầu hết những mẫu, ví dụ như những mẫu chỉ ra trong cuốn sách này là những mẫu thiết kế. Thiết kế luôn luôn là một sự tối ưu của mô hình phân tích, và những mẫu thiết kế luôn là khái niệm cho mô hình phân tích tối ưu như thế nào trong phương pháp cụ thể với những kết quả cụ thể.

Sự tối ưu là đối tác luôn thay đổi. Sự tối ưu luôn đòi hỏi cải tiến một vài khía cạnh của hệ thống phải chịu những chi phí khác. Ví dụ, một vài mẫu sẽ tối ưu khả năng sử dụng lại thì mất đi sự hoàn thiện trong trường hợp xấu. Những mẫu khác sẽ tối ưu an toàn ở chi phí thuê hệ thống(giá trên mỗi phần được định vị). Bất cứ khi nào bạn tối ưu một tập các khía cạnh, thì bạn tối ưu lại những cái khác.

2.1.1. Định nghĩa và các khái niệm về mẫu

Nói chung, một mẫu mô tả môt vấn đề thường hay xảy ra trong thiết kế và cài đặt phần mềm, và sau đó mô tả giải pháp để vấn đề đó theo một cách như vậy có thể được dùng lại. Các mẫu được đưa ra để chứng minh cho các bài thực hành thiết kế tốt. Các mẫu có thể được phân loại theo các giai đoạn phát triển, đó là các Mẫu Phân tích[Fowler 1997], các mẫu Kiến trúc[Buschmann et al. 1996], các Mẫu thiết kế[Gamma et al, 1995] và các idiom (Các cách diễn đạt) [Coplien 1992]:

- *Các mẫu phân tích*: Việc phân tích bao hàm việc tìm kiếm phía sau bề mặt của các yêu cầu để hiểu vấn đề. Martin Fowler 1997 đã định nghĩa các mẫu phân tích như là " *…các nhóm ý tưởng thể hiện việc xây dựng phổ biến trong mô hình kinh doanh"*. Fowler đã chứng minh một vài Mẫu phân tích bên ngoài các kinh nghiệm làm dự án dành cho một vài lĩnh vực kinh doanh. Các mẫu Kiểu, Giám sát và Đo đếm là các Mẫu trong số các Mẫu phân tích đã được Fowler chứng minh.

- *Các mẫu kiến trúc:* một Mẫu kiến trúc giải thích một giản đồ tổ chức có kiến trúc cơ bản cốt lõi cho các hệ thống phần mềm. Nó cung cấp một bộ các hệ thống con được định nghĩa trước hoặc các thành phần, chỉ định các trách nhiệm của chúng và bao gộp các luật cùng với guideline cho việc tổ chức các quan hệ giữa chúng[Buschmann et al. 1996]. Các mẫu Broker, Blackboard, và Filters-Pipes là các mẫu thuộc loại này. Các mẫu có kiến trúc là một biện pháp chứng minh kiến trúc dành cho các hệ thống phức tạp và không đồng nhất, vì vậy việc giúp quản lý tự động là rất phức tạp.

- *Các mẫu thiết kế*: một mẫu thiết kế cung cấp một biểu đồ cho việc cải tiến các hệ thống con hoặc các thành phần của một hệ thống phần mềm hoặc mối quan hệ giữa chúng. Nó mô tả một cách phổ biến kiến trúc tuần hoàn của các thành phần giải quyết vấn đề thiết kế tổng quát trong một ngữ cảnh cụ thể[Gamma et al. 1995]. Các mẫu chiến lược, trạng thái, và Proxy là những ví dụ thuộc loại này.

- *Thành ngữ (Idioms)*: thành ngữ là một Mẫu mức thấp đặc trưng cho một ngôn ngữ lập trình. Một thành ngữ mô tả cách thức để triển khai các khía cạnh phổ biến của các thành phần hoặc các mối quan hệ giữa chúng bằng cách dùng các đặc trưng của ngôn ngữ lập trình đã cho như C++, Java, hoặc Smalltalk. Các ví dụ về thành ngữ là cách thức để cài đặt các Singletons trong C++[Buschmann et al. 1996] và con trỏ được đếm Counted Pointer[Coplien 1992].

Các Mẫu là các kinh nghiệm thiết kế đã được kiểm chứng tốt. người ta thường nói rằng các Mẫu được khám phá ra hoặc được chứng minh chứ không nói là nó được phát minh ra bởi vì chúng phải phát triển lên từ nhiều hơn một dự án thực sự đang tồn tại. Các Mẫu đưa ra các cách rất xúc tích và hiệu quả để chuyển các ý tưởng và các kinh nghiệm Phần mềm vào trong các vấn đề thực tế.

2.1.2. Mục đích của phân tích và thiết kế hướng mẫu

Do nhu cầu của những hệ thống phần mềm tăng lên các nhà nghiên cứu cũng như những nhà thực hành giỏi tìm kiếm những phương pháp luận và kỹ thuật phát triển để làm tự động hoá việc sản xuất phần mềm và làm dễ dàng việc bảo trì chúng. Những kỹ thuật này mới đây đã bao hàm các mẫu thiết kế và khung làm việc thiết kế. Đặc biệt, chúng ta nhận ra sự cần thiết có một phương pháp luận phát triển để phát triển những hệ thống phức tạp qui mô lớn, và đồng thời, học tập được những kinh nghiệm từ những nhà thiết kế các hệ thống khác trong việc giải quyết những vấn đề thiết kế diễn ra.

Có tài liệu về mẫu thiết kế, như nó hiện có, đã miêu tả chi tiết về một mẫu, cách sử dụng nó, cấu trúc, hành vi của người tham gia, sức mạnh và kết quả, và các hướng dẫn triển khai. Những gì chúng ta nhận thấy sự khiếm khuyết là cấu thành các mẫu như thế nào để phát triển những ứng dụng. Một hệ thống đầy đủ không thể, và sẽ không bao giờ được xây dựng từ một mẫu đơn lẻ. Nó là sự tích hợp và cấu thành từ những mẫu để làm nên một hệ thống tổng thể.

Chúng ta có thể cấu thành những mẫu lại cùng nhau ở mức độ lớp hoặc mức độ đối tượng. Những mô hình lớp phơi ra sự triển khai và các khía cạnh bảo trì của một mẫu, trong khi những mô hình đối tượng phơi ra thời gian chạy, hành vi, và những mặt về vai trò. Một số những nhà nghiên cứu và nhà thực hành [Reenskaug 1996; Riehle 1997] lưu ý để những mẫu sử dụng mô hình hóa vai trò và trách nhiệm (Sự phức hợp

về hành vi). Ít quan tâm hơn dành cho vấn đề sáng tạo những mẫu như một sự cấu thành từ lớp (sự phức hợp về cấu trúc). Cuốn sách này là một sự tiến bộ về kỹ thuật phạm vi cấu thành cấu trúc. Chúng ta tin tưởng rằng những mẫu đó không chỉ về những hành vi hoặc vai trò; mặt cấu trúc của chúng là số một, mặt mà chúng ta sẽ triển khai khi sử dụng các ngôn ngữ lập trình hướng đối tượng truyền thống. Mặc dù, dễ dàng hiểu được mối quan hệ giữa các đối tượng khi sử dụng mô hình hành vi, ta cũng dễ dàng triển khai hành vi đó nếu chúng ta có một mô hình cấu trúc nắm bắt được những lớp và mỗi quan hệ giữa chúng. Điều đó cũng giống như những nhà phát triển và nhà thiết kế sẽ tiếp tục sử dụng ngôn ngữ lập trình hướng đối tượng như Java hay C++, và điều đó cũng giống hơn những ai sử dụng cấu trúc lập trình không chuẩn khác. Bởi vậy, một cách tiếp cận cấu thành từ mẫu sử dụng các mô hình cấu trúc mà có được một ánh xạ một - một đến cấu trúc chương trình như những mô hình các lớp là cần thiết.

Mục đích của phân tích và thiết kế hướng mẫu (POAD) là để

- *Đẩy mạnh sự phát triển dựa trên mẫu*: Chúng ta đang tìm kiếm những cách thức để có nhiều nhà thiết kế hơn sử dụng mẫu. Chúng ta muốn thu hút những nhà thiết kế mới vào nghề và giúp họ sử dụng những mẫu bằng cách cung cấp những cách tiếp cận và phương pháp đã được làm đơn giản hoá để họ có thể theo đó sử dụng những mẫu trong trong quá trình thiết kế của mình. Để đẩy mạnh sự phát triển của các mẫu cơ sở, chúng ta cần xác định cách tiếp cận thành phần cái mà rất dễ sử dụng.

- *Phát triển các cách tiếp cận có hệ thống để gắn các mẫu lại*: Có một nhu cầu đang tăng lên để phát triển những cách tiếp cận cấu thành có có hệ thống làm tiện lợi cho việc gắn các mẫu. Những mô hình làm tiện lợi việc tích hợp các mẫu thiết kế cần được phát triển để trợ giúp cho các cách tiếp cận này.

- *Phát triển các khung làm việc thiết kế*: Chúng ta có thể làm dễ dàng việc phát triển những khung làm việc thiết kế (*framework design)* bằng cách sử dụng những mẫu như những khối thiết kế xây dựng.

- *Cải thiện chất lượng thiết kế*: Mẫu thiết kế là những thiết kế có chất lượng cao. Khi sử dụng lại mẫu trong một thiết kế được dự kiến trước để cải tiến chất lượng thiết kế của những ứng dụng phần mềm được xây dựng bằng cách sử dụng các mẫu như là những khối xây dựng cơ sở.

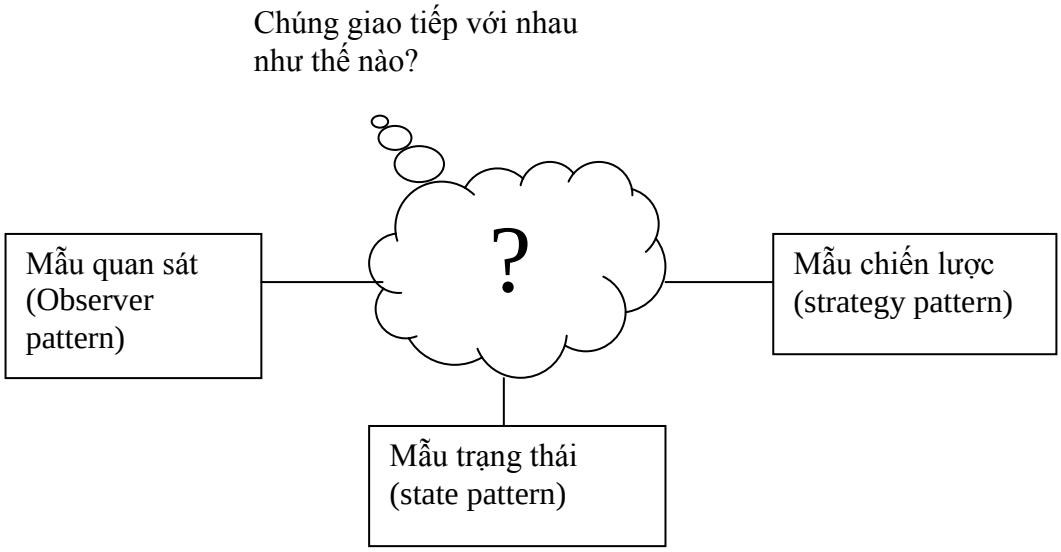
2.1.3. Vai trò của mẫu trong phát triển phần mềm

Sự phức tạp của các hệ thống phần mềm ngày tăng lên, chúng ta tìm kiếm những cách tiếp cận để làm cho thuận tiện sự phát triển ứng dụng phần mềm. Mẫu thiết kế (Design Patterns) [Gamma et al. 1995; Buschmann et al. 1996] và khung làm việc thiết kế (design frameworks) [Johnson & Foote 1998; Fayad & Schmidt 1997; Pree 1996; Fayad & Schmidt 1999] là một trong số những cách tiếp cận đầy triển vọng này. Mẫu thiết kế cho phép sử dụng lại tài sản phần mềm một cách rất sớm trong vòng đời phát triển phần mềm. Để gặt hái những lợi ích của việc phát triển những giải pháp thiết kế đã được chứng minh, chúng ta cần xác định kỹ thuật thiết kế cấu thành để xây dựng những ứng dụng sử dụng mẫu. Những mô hình thiết kế tổng thể cần phát triển để hỗ trợ cho kỹ thuật này.

Khi sử dụng lại phần mềm trong những ứng dụng thực tế là một nhiệm vụ khó khăn, nhưng nó là cần thiết để giảm bớt công sức phát triển và bảo đảm chất lượng phần mềm cao hơn. Mẫu thiết kế giúp cho việc sử dụng lại trong các pha thiết kế, bởi vì nó cung cấp một bảng từ vựng chung cho thiết kế. Nó còn cung cấp những phương thức trên để hiểu những thiết kế và nó là những khối xây dựng đã được chứng minh (đã được kiểm chứng) từ những ứng dụng phức tạp hơn đã được xây dựng. Sự tập hợp những bản mẫu thiết kế có thể dùng được một cách rộng rãi đã thúc đẩy nhiều hơn những ý tưởng làm thế nào để sử dụng những giải pháp đúng đắn trong phát triển ứng dụng.

Khi thiết kế những ứng dụng bằng việc triển khai một cách có hệ thống mẫu thiết kế không phải là một quá trình tầm thường. Cách tiếp cận thiết kế sử dụng kỹ thuật cấu thành từ mẫu đã được đề nghị, nhưng từ mục tiêu để ra đi đến một qui trình có tính hệ thống còn là vấn đề không đơn giản.

2.1.4. Những vấn đề của POAD



Hình 2.1. Vấn đề của POAD

Trong việc đẩy mạnh sự phát dựa triển mẫu và tạo ra những cách tiếp cận mới để cấu thành những mẫu, chúng ta đã đương đầu với nhiều thách thức:

- Cái gì định chất lượng (tính chất) của một mẫu như là một thành phần thiết kế? Để sử dụng những mẫu như là những khối xây dựng, chúng ta cần tìm ra những đặc tính định chất lượng một mẫu như là một thành phần thiết kế. Chúng ta có thể xác định những giao diện mẫu cho mục đích tích hợp với những mẫu khác như thế nào?

- Chúng ta có thể cấu thành những ứng dụng một cách đơn độc từ những mẫu thiết kế không? Nhiều ứng dụng sử dụng một hoặc nhiều mẫu trong thiết kế của họ. Sự thách thức là có hay không những ứng dụng có thể được xây dựng bằng cách gắn kết những mẫu thiết kế với nhau? Các mẫu đó có thể giao tiếp với nhau như thế nào? Những giao diện mẫu là cái gì? Những sai lầm gì về giao diện có thể nảy sinh? Với các tài liệu hiện nay và những mẫu thiết kế này đã đủ để thiết kế các ứng dụng khi sử dụng những mẫu thiết kế? Kiểu mẫu nào có thể được sử dụng?

- Chúng ta có thể phát triển một cách có hệ thống như thế nào những ứng dụng khi sử dụng những mẫu thiết kế? Đã có một quy trình thiết kế được xác định tốt để phát triển những ứng dụng sử dụng những mẫu như là những khối xây dựng của nó chưa?

2.1.5. Giải pháp POAD

Chúng ta phát triển POAD hướng đến những vấn đề ở trên. Trong POAD, những mẫu được xâu chuỗi ở thiết kế mức cao, và những phần của chúng được kết hợp lại trong những giai đoạn thiết kế tiếp theo để thu được thiết kế sâu sắc và chặt chẽ. Cách tiếp cận POAD cung cấp những giải pháp sau:

1. *Mô hình hóa các mẫu như là các thành phần thiết kế*. Từ các mẫu sẽ là những khối xây dựng của các thiết kế hướng mẫu, đầu tiên chúng sẽ xử lý đặc điểm của một thành phần, chủ yếu tạo khả năng cấu thành ở mức độ thiết kế. Điều này đòi hỏi phải xác định những giao diện mẫu và các thuộc tính mẫu chủ yếu để có thể làm cho trở thành một thành phần thiết kế. POAD định nghĩa một kiểu cụ thể của các mẫu gọi là các mẫu thiết kế cấu trúc. POAD cũng xác định các mức đó được lần vết đến mức thiết kế thấp hơn như thế nào theo thuật ngữ của các lớp. Việc phát triển dựa trên mẫu, chúng ta cần xác định một cách tiếp cận cấu thành dễ dàng sử dụng được.

2. *Một phương pháp luận thiết kế*. Chúng ta thảo luận một phương pháp luận để xây dựng những thiết kế hướng mẫu. POAD là một sự cố gắng rõ ràng để nghiên cứu những mẫu như là những khối xây dựng cốt lõi của những thiết kế hướng đối tượng. Trong POAD, chúng ta học những kinh nghiệm của phương pháp phân tích và thiết kế hướng đối tượng và xác định một phương pháp hướng mẫu mới được xây dựng trên cú pháp và ngữ nghĩa của ngôn ngữ mô hình hoá thống nhất (UML). Chúng ta gọi những thiết kế được phát triển khi sử dụng phương pháp luận này là : *những thiết kế hướng mẫu.*

3. *Những ứng dụng thế giới thực*. Để nghiên cứu khả năng ứng dụng của kỹ thuật POAD, chúng ta áp dụng phương pháp luận thiết kế cho bốn ứng dụng và xây dựng một thiết kế hướng mẫu cho mỗi cái.

Mục đích chính của POAD là để cung cấp một giải pháp nhằm cải tiến thực hành của việc phát triển những mẫu thiết kế một cách hệ thống trong phát triển ứng dụng. Chúng ta tin tưởng rằng vấn đề sẽ được xử lý ở một giai đoạn sớm trong vòng đời phát triển, chủ yếu ở mức phân tích và thiết kế. Cách tiếp cận chúng ta tiến hành là để phát triển phương pháp luận POAD cho xây dựng thiết kế hướng đối tượng khi sử dụng những mẫu thiết kế như là những khối xây dựng.

Trong pham vi kỹ nghệ phần mềm, một phương pháp luận thiết kế có ba thứ nguyên chính đó là: *Công nghệ, tiến trình, và tổ chức*.

- *Mặt công nghệ*. Ở mặt này xác định những nền tảng của phương pháp luận thiết kế, bao gồm những khái niệm, những ký pháp, và những mô hình trực quan và hình thức. Chúng ta xác định những mô hình trực quan, cái đó được dùng để gắn những mẫu một cách có cấu trúc để phát triển những thiết kế hướng mẫu. Cho mục đích này, chúng ta giới thiệu khái niệm của những giao diện mẫu và thảo luận mỗi quan hệ với các thành phần phần mềm và kiến trúc. Chúng ta thảo luận những trợ giúp về ngữ nghĩa và cú pháp của UML cho phương pháp luận POAD.

- *Mặt tiến trình*. Mặt này xác định những nhiệm vụ và những bước cơ bản để phát triển những mẫu hướng mẫu. Dựa trên những mô hình trực quan, các bước phân tích và thiết kế đã được xác định. Đầu ra và những xuất phẩm ở mỗi bước cũng được xác định. Công cụ giúp cho POAD cũng được thảo luận trong cuốn sách này.

- *Mặt tổ chức*. Mặt này xác định Doanh nghiệp được tổ chức như thế nào để được phương pháp và một cách hiệu quả.

2.2. Sự cấu thành của các mẫu thiết kế đối với kỹ nghệ phần mềm

2.2.1. Lịch sử của mẫu thiết kế

Trong công nghệ phần mềm, một mẫu thiết kế là một giải pháp tổng thể cho các vấn đề chung trong thiết kế phần mềm. Một mẫu thiết kế không phải là một thiết kế hoàn thiện để có thể được chuyển đổi trực tiếp thành mã; nó chỉ là một mô tả hay là sườn (template) mô tả cách giải quyết một vấn đề mà có thể được dùng trong nhiều tình huống khác nhau. Các mẫu thiết kế hướng đối tượng thường cho thấy mối quan hệ và sự tương tác giữa các lớp hay các đối tượng, mà không cần chỉ rõ các lớp hay đối tượng của từng ứng dụng cụ thể. Các giải thuật không được xem là các mẫu thiết kế, vì chúng giải quyết các vấn đề về tính toán hơn là các vấn đề về thiết kế.

Việc viết tài liệu cho một mẫu thiết kế nên chứa đựng đủ thông tin về vấn đề mà mẫu đề cập, ngữ cảnh trong đó nó được sử dụng và giải pháp được đề nghị. Tuy vậy, các tác giả mẫu thường sử dụng các dạng trình bày mẫu riêng của mình để trình bày các mẫu thiết kế, và các dạng trình bày mẫu thường tương tự với các phần cần thiết đã nêu. Các tác giả thường bao gồm thêm một số đoạn để cung cấp nhiều thông

tin hơn và tổ chức các phần cần thiết trong những đoạn khác nhau có thể có với những tên khác nhau.

Các mẫu xuất phát từ một khái niệm kiến trúc đưa ra bởi Christopher Alexander. Vào năm 1987, Kent Beck và Ward Cunningham bắt đầu thử nghiệm ý tưởng áp dụng các mẫu vào lập trình và đưa ra các kết quả của chúng tại hội thảo OOPSLA(Object-Oriented Programming, Systems, Languages, Applications) vào năm đó. Vào các năm tiếp theo, Beck, Cunningham và những người khác vẫn tiếp tục với công việc này.

Các mẫu thiết kế đã trở nên phổ biến trong khoa học máy tính sau khi cuốn sách Design Patterns: Elements of Reusable Object-Oriented Software được phát hành vào năm 1994 (Gamma et al). Vào cùng năm đó, cuộc hội thảo đầu tiên về Các ngôn ngữ mẫu cho các chương trình đã được tổ chức và vào năm sau, kho dự trữ các mẫu Portland (Portland Pattern Repository) đã được thiết lập để lưu trữ văn bản về các mẫu thiết kế.

2.2.2. Vòng đời của mẫu thiết kế trong việc phát triển phần mềm

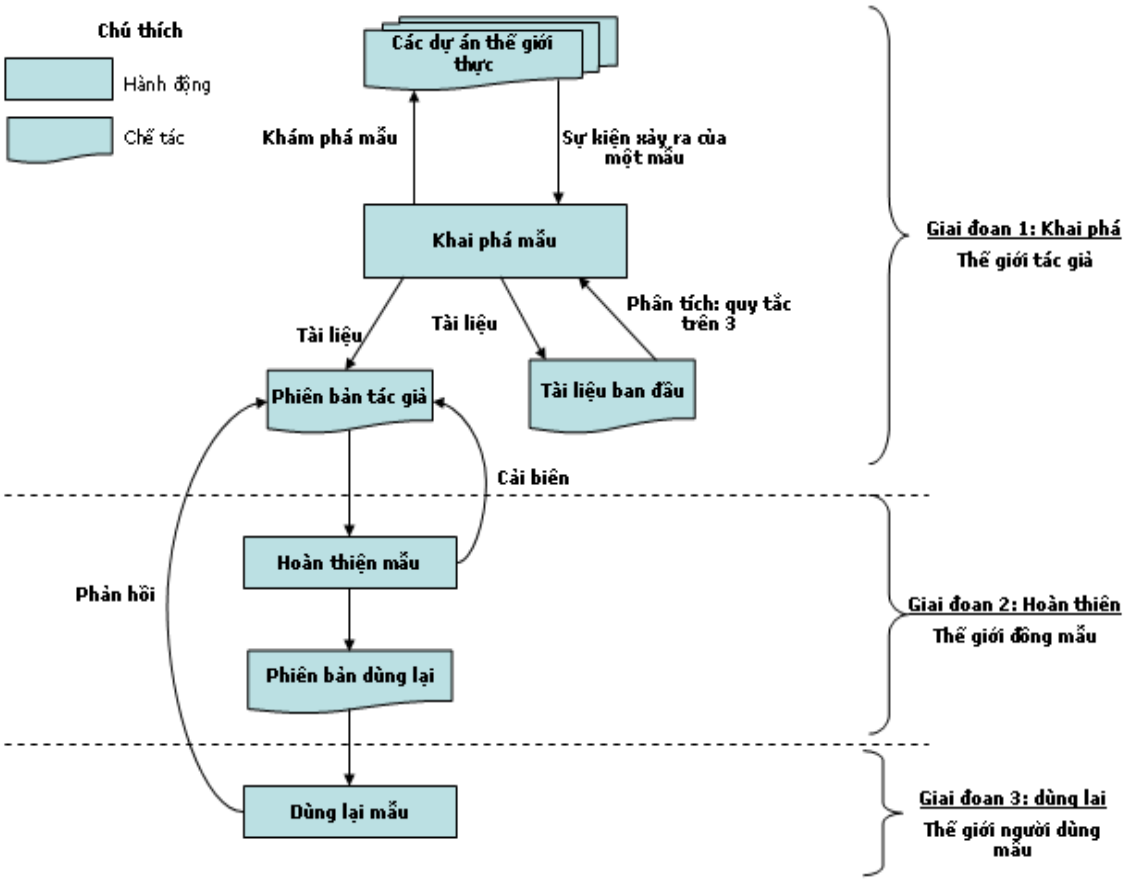
Giai đoạn 1: **Khai phá.**

Trong việc tạo ra bất kì mẫu thiết kế nào, giai đoạn đầu tiên đều liên quan tới việc chứng minh khởi đầu của một Mẫu. Hành động chính trong giai đoạn này là việc khai mở các Mẫu.

Giai đoạn 2: **Hoàn thiện.**

Giai đoạn thứ 2 được các nhà nghiên cứu và các nhà thực hành có kinh nghiệm đề cập đến việc đánh giá và cải tiến các Mẫu. Trong giai đoạn này tác giả của Mẫu đưa Mẫu ra để xem xét như một trong các cuộc hội thảo PloP hàng năm. Việc đệ trình này sau đó được phân công cho một nhà phê bình. Mẫu hoặc được nhân vào cuộc thảo luận hoặc bị từ chối. Các mẫu được chấp nhận sẽ được thẩm định lại trong cuộc họp với một nhóm các tác giả có kinh nghiệm, họ tạo ra các yêu cầu cho các cải tiến đối với Mẫu và chia sẻ kinh nghiệm của họ trong việc giải quyết các vấn đề cùng nhau. Đối tượng của giai đoạn này là giúp đỡ tác giả của Mẫu cải thiện phiên bản Mẫu của họ và trong một vài trường hợp là từ bỏ Mẫu vì sự thiếu hợp lý và thiếu tiềm năng của việc dùng lại. Đầu ra của giai đoạn này là Mẫu được chứng minh tốt cho việc dùng đối với

những người mới làm quen, các nhà thiết kế ứng dụng, và các nhà phát triển. Sau đó phiên bản được xem lại sẽ được công bố trong Biên bản lưu của cuộc họp.



Hình 2.2. Vòng đời của một Mẫu

Giai đoạn 3: Việc dùng lại.

Giai đoạn thứ 3 liên quan đến cùng với việc dùng lại Mẫu trong các ứng dụng. Các người dùng Mẫu tìm kiếm các Mẫu trong số các Biên bản hội nghị PloPD đã được công bố hoặc các sách PloPD (Pattems Language Of Program Design). Sau đó họ thể hiện Mẫu trong các dự án thực tế. Người dùng cung cấp thông tin phản hồi lại cho tác giả của Mẫu những trở ngại mà họ phải đối mặt trong khi cài đặt và dùng chúng, và đưa ra các yêu cầu cải tiến các Mẫu đó.

Như được chỉ trong hình, tiến trình trên là lập lại và vì vậy một Mẫu sẽ được cải tiến tiếp tục. Chúng ta mong đợi rằng các Mẫu đã được chứng minh tính đúng đắn trong việc dùng lại sẽ đạt được chất lượng cao, bởi vì nó đã trải qua một số giai đoạn cải tiến. Chất lượng cao là một thuộc tính then chốt của thành phần thiết kế và vì vậy

các Mẫu tuân theo vòng đời phát triển này sẽ đủ điều kiện để là các thành phần thiết kế.

2.2.3. Sự cấu thành từ các mẫu thiết kế

Sự phát triển ứng dụng khi sử dụng mẫu thiết kế như các thành phần yêu cầu xem xét kĩ các kĩ thuật cấu thành. Trong chương này, chúng ta sẽ điểm qua một vài kĩ thuật cấu thành các mẫu thiết kế. Có thể phân loại như sau:

- Các kĩ thuật cấu thành hành vi
- Các kĩ thuật cấu thành cấu trúc

Các kĩ thuật hành vi dựa trên đặc tả tương tác đối tượng để chỉ rõ các thể hiện cụ thể của các mẫu có thể được cấu thành như thế nào. Còn kĩ thuật cấu trúc lại dựa trên các đặc tả kiến trúc tĩnh bao gồm các mẫu cụ thể khi sử dụng sơ đồ lớp. Mặc dù, chúng ta chấp nhận và thực hiện cách tiếp cận cấu trúc trong chương tiếp theo nhưng kĩ thuật lai của hai kĩ thuật có thể bao gồm một cách tiếp cận cho việc đặc tả các mẫu thiết kế được cấu thành như thế nào.

Phân tích các mẫu hành vi là một chủ đề rất quan trong, nhất là khi mà các hành vi đó kết hợp với nhau phát triển một thiết kế ứng dụng. Sự cấu thành hành vi có thể cho kết quả không tin cậy mà trong đó lỗi thiết kế được đưa vào liên quan đến sự tương tác không tính trước giữa các thành phần thiết kế riêng biệt. Những tương tác này phải được phân tích thông qua các đặc trưng hành vi của thành phần đó. Do đó, rất nhiều nhà thiết kế thích kĩ thuật cấu thành hành vi hơn. Dù sử dụng kĩ thuật cấu thành nào, thì đều cần phân tích hành vi của các thành phần thiết kế một cách cẩn thận, nhằm tránh các tương tác sai. Chú ý rằng mẫu thiết kế kiến trúc chủ yếu tạo ra các mẫu trong đó các phương thức cụ thể sẽ điều khiển sự tương tác giữa các đối tượng, và do đó sẽ đưa ra các thiết kế điều khiển các tương tác đối tượng. Giao diện này cung cấp đặc tả tương tác giữa các đối tượng trong mỗi loại mẫu khác nhau và dễ dàng phân tích hành vi.

Việc tăng số lượng các mẫu và các ngôn ngữ mẫu đánh dấu sự phát triển của kĩ thuật phân loại mẫu và mối quan hệ giữa các mẫu. Sự phân loại mẫu được chúng tôi giới thiệu ở đầu cuốn sách này bao gồm mẫu tạo ra, cấu trúc và hành vi. Mỗi mẫu khi được phát triển cũng chỉ ra mối quan hệ với 23 mẫu trong cuốn : Elements of Object-Oriented Software [Gamma et al. 1995]. Từ đó, mỗi mẫu đều miêu tả mối liên hệ với

các mẫu khác, các tài liệu mẫu cũng đã chỉ ra lược đồ phân loại mối quan hệ. Năm 1998, James Noble đã phát triển lược đồ phân loại mối quạn hệ giữa các mẫu. Ba mối quan hệ sơ cấp và một vài mối quan hệ thứ cấp giữa các mẫu đã được chỉ ra. Mối quan hệ sơ cấp dựa trên những yếu tố sau:

- Sử dụng mối liên hệ - một pattern "sử dụng" pattern khác
- Làm mịn mối liên hệ - một pattern "làm mịn" pattern khác
- Đối lập mối liên hệ - một pattern "xung đột" pattern khác

Thông thường, các mẫu phức tạp sẽ "sử dụng" các mẫu đơn giản hơn, ví dụ như 1996, mẫu MVC (Model - View - Constroller) sử dụng cách cài đặt của mẫu quan sát (Observer pattern), mẫu chiến lược (Strategy pattern), mẫu cấu thành (Composite pattern). Mối quan hệ "Làm mịn" sử dụng để xác định một mẫu mới như là sự làm mịn các mẫu đã có. Mối quan hệ "đối lập" chỉ ra ví dụ của các mẫu không thể sử dụng cùng nhau - nhất là, khi chúng đều đưa ra các giải pháp loại trừ lẫn nhau về cùng vấn đề, và do đó không thể sử dụng chúng thay thế lẫn nhau được. Noble cũng đã thảo luận 1 số mối quan hệ thứ cấp và cho rằng chúng được xác định bởi các mối quan hệ sơ cấp. Lược đồ phân loại này giúp ích cho tiến trình chọn mẫu từ cơ sỏ dữ liệu mẫu trong suốt pha phân tích của POAD

Trong phần này, chúng ta không thảo luận mối liên quan giữa các mẫu nhưng các mẫu có thể cấu thành cùng nhau để phát triển các mẫu cấu thành, ứng dụng hướng đối tượng và các khung làm việc hướng đối tượng. Điều này liên quan đến vần đề cấu thành các thành phần phần mềm. Hiểu được mối quan hệ giữa các mẫu riêng biệt là một cách rèn luyện tốt nhưng không giải quyết được vấn đề liên quan của sự cấu thành mẫu.

2.2.4.Tính cấu trúc của sự cấu thành các mẫu thiết kế

Các hướng tiếp cận Cấu trúc sự cấu thành (Structural composition) xây dựng một thiết kế bằng cách gắn kết các mẫu cấu trúc, cái mà được dựng lên như một sơ đồ lớp. Cấu trúc sự cấu thành làm nổi bật hơn trong thực tế của thiết kế hơn sự trừu tượng, sử dụng những kiểu khác nhau của mô hình như mô hình vai trò. Kỹ thuật cấu thành hành vi, như vai trò [Reenskaug 1996, Riehle 1997, Kristensen & Osterbye 1996] để lại những sự lựa chọn riêng biệt để người thiết kế với sự hiểu biết ít ỏi có thể tiếp tục pha thiết kế lớp. Các kỹ thuật chỉ ra cả cấu trúc và khung hành vi có thể trở

nên phức tạp và khó sử dụng. Về việc đó, cách tiếp cận POAD ủng hộ một hướng tiếp cận Cấu trúc sự cấu thành (Structural Composition) với các sơ đồ lớp mẫu. Cấu trúc mẫu thiết kế trong mỗi giao diện mẫu có thể trở nên rõ ràng cụ thể, làm cho bản thân chúng tới gần hướng tiếp cận Cấu trúc sự cấu thành (Structural Composition).

Trong những phần sau, chúng ta thảo luận về một vài kỹ thuật Cấu trúc sự cấu thành (Structural Composition) và đối chiếu những kỹ thuật đó với hệ phương pháp luận POAD đã đưa ra của chúng ta. Chúng ta đã phác thảo một kế hoạch cho mẫu thiết kế định hướng đưa ra bởi Ram, Anantha, and Guruprasad in "A Pattern-Oriented Technique for Software Design" (1997). Trong sự thỏa thuận tới hướng tiếp cận trên xuống của chúng tôi, thì hướng nghiên cứu này mô tả một tiến trình xử lý dưới lên để thiết kế phần mềm sử dụng các mẫu thiết kế. Hướng tiếp cận này chỉ ra làm thế nào để liên kết các mẫu được lựa chọn; tuy nhiên, nó không chỉ ra rõ ràng mẫu nào sẽ được sửa soạn. Tuy thế, nó đưa ra một ví dụ của việc cố gắng thử trước trong tài liệu để phát triển một tiến trình xử lý có phương pháp cho việc phát triển phần mền hướng thiết kế.

Chúng ta tóm gọn lại khái niệm của thiết kế phần mền sử dụng thành phần thiết kế, mà nó có thể sửa lại cho hợp với những thay đổi yêu cầu nhanh chóng như đã đưa ra bởi Keller and Schauer trong "Design Components: Towards Software Composition at the Design Level" (1998). Công việc này được đưa ra để làm nổi bật tầm quan trọng của khái niệm thành phần thiết kế phát sinh, nơi các thành phần được chỉ rõ trong mức cao của sự trừu tượng. Khái niệm này có liên quan rất nhiều tới quan điểm về việc sử dụng mẫu thiết kế như là các thành phần thiết kế. Khái niệm này cũng không giống với khái niệm của sự phát triển phần mềm dựa trên các thành phần, cái mà chủ điểm chính của nó là sự phát triển hoặc là mã hóa thành phần mức cao. Như việc mô tả, ví dụ những sơ đồ thành phần trong những bộ phận có thể mở ra để các thành phần thiết kế miền rõ ràng từ chúng. Chúng ta có thể thu được thành phần mã hóa mức cao. Hướng tiếp cận này đảm bảo rằng thiết kế chất lượng chuyển sang mã hóa chất lượng

Chúng ta cũng đề cập đến một thảo luận khác trong tài liệu, sự phát triển của khung làm việc dựa trên thành phần sử dụng các mẫu. Trong công việc này, một thiết kế kiến trúc có trật tự được phát triển trong một thành phần có thể thực thi rất nhiều mẫu và nhiều khung làm việc và một mẫu thiết kế hay một khung làm việc có thể được thực thi thông qua rất nhiều thành phần. Công việc này đưa ra một hướng tiếp cận cấu

trúc để gắn lại các mẫu thiết kế sử dụng các lớp giao diện UML với giao diện mô hình của mẫu. Các sơ đồ thành phần UML sau đó được sử dụng để chỉ ra khung làm việc như một thành phần của các thành phần vật lý mà không có ánh xạ một – một với các mẫu. Chúng ta tóm gọn lại thảo luận sự khác nhau giữa công việc này với sự tiếp cận POAD cho sự phát triển hướng thiết kế.

Hướng tiếp cận Catalysis đưa ra bởi D'Souza and Wills và thảo luận trong "Catalysis: Component and Framework-based Development" (1998). Hướng tiếp cận này chỉ rõ một hướng tiếp cận dựa trên thành phần tới sự phát triển phần mềm cái mà nặng về việc dựa trên giao diện tại cả mức thực hiện và thiết kế. Hướng tiếp cận Catalysis thường được sử dụng để xây dựng đối tượng và hệ thống dựa trên thành phần sử dụng UML và mở rộng của UML.

Cuối cùng, chúng ta tóm tắt lại sự chỉ dẫn khái niệm mới được phát triển của các mẫu cấu thành và thiết kế hướng chủ để [Clarke & Walker 2001; Clarke et al. 2000]. Những khái niệm này làm xuất hiện một kỹ thuật phát triển dựa trên khái niệm chương trình hướng giao diện.

2.3. Các đặc trưng của phân tích thiết kế hướng mẫu- POAD

Quan sát các mô hình và các phác thảo toàn diện về tiến trình được mô tả trong các phần trước, chúng ta có thể suy ra một vài đặc tính của phương pháp POAD. Trong phần này chúng ta sẽ tổng hợp một vài đặc tính đó.

2.3.1. Điều khiển bởi mẫu (Pattern-Driven)

Phần lớn các phân tích hướng đối tượng và các phương pháp luận thiết kế sử dụng các lớp và các đối tượng như là các khối xây dựng thiết kế. Nhiều ngôn ngữ mô hình hóa hướng đối tượng như UML, cung cấp các cơ chế nhóm gộp như các gói và hệ con để cấu trúc một *khung nhìn hạt lớn* của thiết kế. Các cơ chế nhóm gộp được sử dụng để quản lý sự phức tạp của mô hình bằng cách xem thiết kế như là số các gói cộng tác, tức là được phân rã và làm mịn.

Tiếp cận POAD được điều khiển bằng mẫu có nghĩa là, mọi thứ bắt đầu từ việc thể hiện các mẫu trong thiết kế. Trong POAD, ta xây dựng các ứng dụng từ các khối thiết kế lớn, tức là từ các mẫu thiết kế cấu trúc. Trong ngữ cảnh này, các mẫu được sử dụng các cơ chế nhóm gộp và đồng thời như các giải pháp thiết kế có thể dùng lại được. Với cơ chế nhóm gộp, một mẫu sẽ bao gói một tập các lớp để giải quyết một vấn

đề thiết kế cụ thể. Như một giải pháp thiết kế có thể dùng lại, mỗi mẫu cung cấp một giải pháp trừu tượng cho các vấn đề thiết kế phổ biến. POAD dựa trên các mẫu giống như các khối xây dựng có thể sử dụng lại của các thiết kế ứng dụng.

2.3.2. Phát triển dựa trên thành phần

Kĩ nghệ phần mềm dựa trên thành phần *(componnt-based engineering)* đang nổi lên như mô hình có lợi cho phát triển phần mềm, mở ra nhiều hứa hẹn cho việc sử dụng lại phần mềm: *Bản chất của việc xác định một thành phần, khi nào một thành phần có thể được sử dụng trong phát triển phần mềm và ở các pha phát triển nào*. Một thành phần có thể được trừu tượng hóa một chức năng, dữ liệu, gói hay cấu trúc hệ thống. Nhìn chung, nhiều người ám chỉ một thành phần như một lớp, một đoạn mã, một đoạn thiết kế, một module có khả năng thực thi, một thư viện liên kết tại thời gian chạy (thư viện động, DLL) hay là một thư viện tĩnh. Các thành phần có thể được phân loại dựa trên các đặc tính của chúng:

- *Các thành phần đặc tả*: Đặc tả các chức năng hay hành vi mong đợi của một thành phần giúp cho người phát triển tự do áp dụng thành phần trong các ngôn ngữ lập trình khác nhau. Một ví dụ sử dụng các đặc tả như các thành phần được minh họa bởi Iglesis và Justo trong "Building System Requirements with Specification Components"(1998).

- *Khả năng thực thi*: Các thành phần có thể thực thi được có thể là các thư viện tĩnh, DLLs hoặc là các mảnh có khả năng thực thi của một ứng dụng. Nhiều tài liệu tham chiếu đến các thành phần loại này.Thông thường, mã nguồn của các thành phần này không có sẵn, nhưng các hướng dẫn để tích hợp chúng trong quá trình phát triển được đưa ra trong các tài liệu phân phối cùng thành phần tương ứng.

- *Các thành phần thiết kế*. Một thành phần có thể là một nguyên tắc hay cấu trúc thiết kế. Các mẫu thiết kế cấu trúc như thảo luận trước ở chương IV được sử dụng như các khối xây dựng thiết kế trong việc cấu trúc ứng dụng hướng đối tượng. Thông thường, các thành phần loại này được sử dụng như là các hộp trắng tại mức thiết kế.

Trong POAD, các ứng dụng được xây dựng bằng cách sử dụng các thành phần thiết kế như là các khối xây dựng của chúng. Nó tạo ra thiết kế mà bản chất là dựa trên thành phần, nhưng chúng ta phải nhớ rằng, các thành phần đã sử dụng ở đây là các thành phần thiết kế hộp trắng. Nó không làm giảm bớt sự quan trọng của việc sử dụng

các giao diện để dính kết các thành phần này. Trong POAD, các mẫu thiết kế cấu trúc được sử dụng như là các thành phần thiết kế, và chúng có các giao diện giống như đã thảo luận trong chương IV.

2.3.3. Sự phát triển kiến trúc

Kiến trúc phần mềm xem xét cấu trúc của một ứng dụng hơn là chức năng của nó [Shaw & Garlan 1996]. Khi chọn một cấu trúc tốt làm ảnh hưởng đến các thuộc tính chức năng và phi chức năng của ứng dụng, như sự dễ dàng tích hợp, hiệu suất và vững chắc trước các thay đổi [Dyson & Anderson, 1997]. Khi phát triển một kiến trúc ứng dụng, chúng ta không quan tâm đến các chi tiết về sự thực thi, mà quan tâm chủ yếu đến các chức năng được định vị cho các thành phần và xác định sự tương tác của chúng.

Trong POAD chúng ta xây dựng các ứng dụng từ tập hợp các mẫu thiết kế cấu trúc, đó là các thành phần thiết kế. Chúng ta chọn một mẫu để giải quyết môt vấn đề thiết kế cụ thể và xác định mối quan hệ giữa các mẫu và cách thức chúng tương tác với nhau. Do đó, các mô hình kết quả từ POAD có thể được sử dụng như là khung nhìn kiến trúc của ứng dụng. Phân rã ở mức cao của các ứng dụng được xem xét là một cách tiếp cận dựa trên kiến trúc. Chúng ta đã thảo luận về các mặt kiến trúc của khung nhìn mức mẫu.

2.3.4. Phát triển điều khiển bởi thư viện (Library-Driven)

POAD sử dụng các danh mục có sẵn các mẫu thiết kế. Các tiếp cận này chủ yếu dựa vào sự tồn tại của các thư viện có thể sử dụng lại của các mẫu thiết kế mà nó có thể xem và yêu cầu. Do đó, nhiều vấn đề về thư viện sử dụng lại truyền thống gắn với việc trích rút tài sản lưu trữ trong các thư viện mẫu. Bản chất của một mẫu thúc đẩy cách tiếp cận khác để bảo trì và duyệt các thư viện khác với thư viện tài sản mã nguồn. Một thư viện của các mẫu là cần thiết cho cách tiếp cận POAD. Tuy nhiên, các vấn đề liên quan đến việc xây dựng vào bảo trì các thư viện mẫu là các chủ đề nghiên cứu sau này.

2.3.5. Sử dụng lại thiết kế

Tiếp cận POAD khuyến khích dùng lại ở mức thiết kế bằng việc sử dụng lại các mẫu thiết kế cấu trúc để xây dựng các thiết kế ứng dụng. So sánh với sử dụng lại mã, chúng ta tin rằng, sử dụng lại đối với mức thiết kế có một số ưu điểm sau:

- Bản thiết kế phần mềm là sản phẩm của các hoạt mệt mỏi và phức tạp của việc phát triển phần mềm, đặc biệt là phân tích yêu cầu và thiết kế ứng dụng. Những quyết định trong thiết kế thì thường khó hơn và căng thẳng hơn là các quyết định triển khai ở mức thấp. Sử dụng các quyết định thiết kế thành công sẽ cải tiến chất lượng phần mềm và làm giảm các rủi ro của sự phát triển.

- Thiết kế là một tiến trình lặp. Thiết kế có thể được hướng dẫn bởi các phân tích ở mức cao (từ trên xuống) cũng như tiếp cận phát triển ở mức thấp (dưới lên). Tiếp cận từ trên xuống đảm bảo rằng, thiết kế theo đúng các yêu cầu của người sử dụng, trong khi đó tiếp cận từ dưới lên lại đảm bảo thiết kế có tính thực thi.

- Sự thể hiện của sử dụng lại thường giống nhau nhiều ở các thiết kế hơn là khía cạnh thực thi hay mã nguồn. Nó thường khó để đạt được đặc tả thành phần hoặc các yêu cầu phía người sử dụng đối với tính thực hiện được của các đặc tả thành phần. Sự thỏa mãn đối với các thiết kế có thể khả thi hơn nhờ có khả năng sử dụng lại các hộp trắng của thiết kế thông qua sự làm thích nghi.

- Các mẫu thiết kế có tính cấu trúc là các giải pháp thiết kế có đặc điểm chung. Những người thiết kế ứng dụng thường sử dụng giải pháp thiết kễ phổ biến và tập trung vào các ứng dụng cụ thể.

- Các ngôn ngữ lập trình không cung cấp sự trừu tượng hóa ở mức cao cần thiết phù hợp với các nhà thiết kế hệ thống. Các mẫu thiết kế cấu trúc cung cấp sự trừu tượng hóa ở mức cao này bằng cách cô lập các phần cung cấp có ý nghĩa và có chức năng của mẫu.

- Từ khi POAD bắt đầu với các thành phần thiết kế được sử dụng lại, nó khuyến khích sử dụng lại ở mức thiết kế.

2.3.6. Phát triển phân cấp

Sự phân rã và tích hợp một cách phân cấp thường được ám chỉ các mỗi quan hệ bao gồm hay một bộ phận của...Các mô hình mà chúng ta thảo luận trong chương V về POAD cho phép phân rã phân cấp. Khi xem xét mối quan hệ giữa toàn bộ và các phần của nó, chúng ta có thể xác định ba loại cấu trúc phân cấp:

- *Sự bao gói*: Tổ hợp toàn bộ che giấu các thành phần bên trong với thế giới bên ngoài.

- *Sự nhúng:* Tổ hợp toàn bộ được cấu thành từ các thành phần mà thế giới bên ngoài có thể nhìn thấy.
- *Sự phân rã ảo*: Tổ hợp toàn bộ là những khái niệm, không tồn tại như là một chế tác.

Khi sử dụng các mẫu thiết kế cấu trúc như các thành phần thiết kế là một tiếp cận ảo để phân rã. Đó là vì chính bản thân các mẫu có thể không hữu hình so sánh với các lớp mà sẽ được thực thi trong cấu trúc mã, xây dựng và các thể hiện như các đối tượng. Do đó mẫu có thể được xem như là một nhóm ảo. Tại mức biểu đồ mẫu, các mẫu bao gói các chi tiết bên trong chúng và đưa ra các giao diện.

2.3.7. Phát triển lặp

Tiến trình phát triển là một quá trình lặp, giống như phân tích trước đây, khi sử dụng các biểu đồ tuần tự, thí dụ: các quan hệ giữa các lớp có thể thay đổi. Những thay đổi này phải được phản hồi lại trong biểu đồ mức mẫu. Công cụ hỗ trợ cho tiến trình POAD nên duy trì sự nhất quán giữa các mô hình thiết kế. Các bước thiết kế chi tiết mà đi theo các kĩ thuật hướng đối tượng truyền thống bằng cách xác định các chi tiết và các khía cạnh hàng vi của các lớp và các chế tác thiết kế khác.

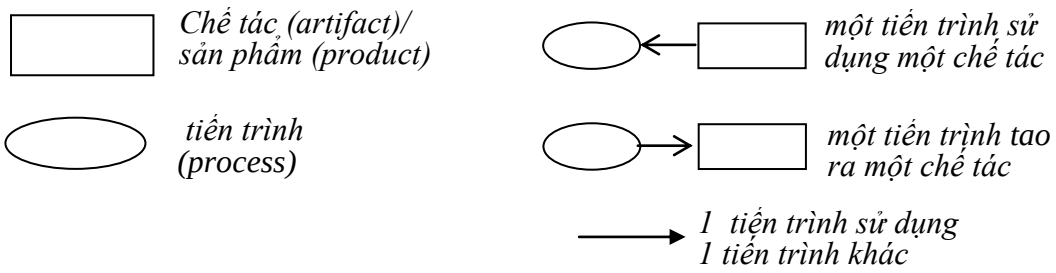
CHƯƠNG 3: TIẾN TRÌNH CỦA PHÂN TÍCH THIẾT KẾ HƯỚNG MẪU

Các khía cạnh của tiến trình POAD giải thích các pha và các bước để phát triển một thiết kế ứng dụng có sử dụng các mẫu. Đầu ra của tiến trình điều khiển là một thiết kế mẫu hướng đối tượng hoặc là một khung làm việc hướng mẫu. Yêu cầu chính của đầu vào tiến trình là thư viện các mẫu thiết kế cấu trúc được xây dựng. Chúng ta sử dụng thuật ngữ pha để ám chỉ các giai đoạn đã được biết trong phát triển phần mềm: *phân tích, thiết kế, thiết kế chi tiết, thực thi, kiểm thử*… Chúng ta sử dụng thuật ngữ bước để chỉ các hoạt động hay bước tiến trình mà người phân tích hay thiết kế tiến hành bên trong các pha cụ thể. Chúng ta sẽ giải thích mỗi bước của pha phát triển qua *mục đích, tiến trình* và *sản phẩm*; Phần mục đích giải thích vì sao nhà thiết kế tiến hành bước đó, phần tiến trình mô tả sự hoạt động mà người thiết kế thực hiện trong bước đó, phần sản phẩm mô tả đầu ra mong muốn của bước đó.

Thông thường tiến trình POAD gồm có 3 pha: Trong *Pha phần tích* một tập các mẫu được chọn từ một thư viện miền cụ thể. Ở pha thiết kế ở mức cao, các mẫu được gắn lại với nhau bằng cách sử dụng các mô hình cấu thành mẫu để tạo ra biểu đồ lớp ban đầu, và pha làm mịn thiết kế biểu đồ lớp ban đầu được xử lý để tạo ra biểu đồ lớp đậm đặc hơn và sâu hơn cho ứng dụng .

Mô hình thác nước mà chúng ta sẽ sử dụng để giải thích cho tiến trình POAD giúp ta có một khung nhìn tổng quát về các giai đoạn và các bước bên trong mỗi pha. Sự tăng trưởng và sự phát triển lặp được khuyến khích và trên thực tế được minh họa bằng cách sử dụng một vài vòng lặp trong các tiến trình. Sự lặp lại từ một cho đến nhiều bước được khuyến khích và có thể thực hiện được bằng cách giữ các mô hình thiết kế đã được tạo ra trong mỗi bước và cung cấp sự lần vết theo các mô hình này bằng cách sử dụng một môi trường phát triển hay một công cụ hỗ trợ .

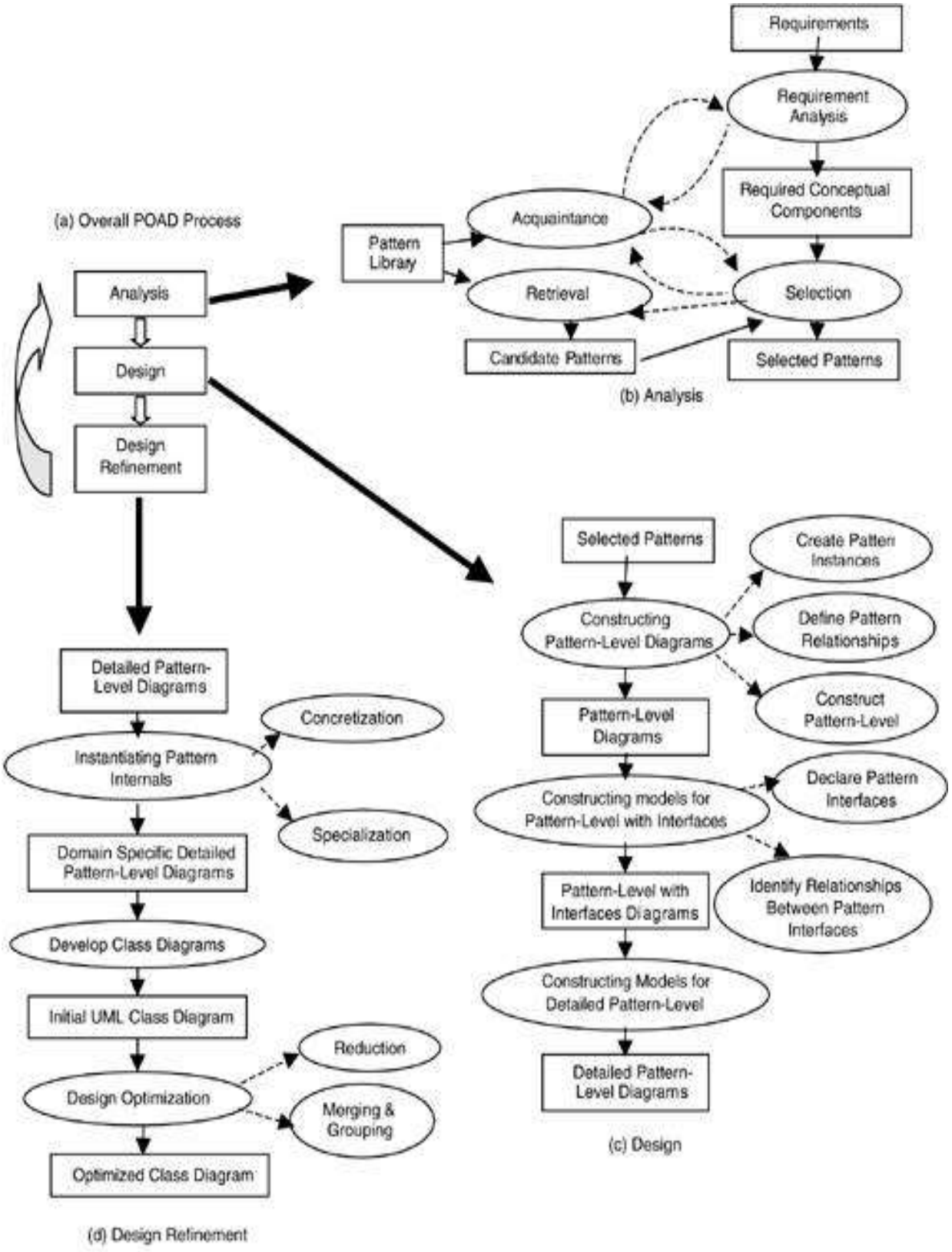
Hình 3.1 minh họa kí pháp mà chúng ta sử dụng để mô tả vòng đời của POAD. Chúng ta sử dụng hình vuông để mô tả một tác nhân và hình elíp để mô tả một tiến trình trong một bước nào đó. Hình 3.2 minh họa toàn bộ các pha phát triển của POAD. Hình 3.2a cho thấy sự phân tích, thiết kế và các pha làm mịn thiết kế, mỗi pha này được giải thích chi tiết hơn trong hình 3.2a,b,c,d



Hình 3.1. Các kí hiệu được sử dụng để miêu tả POAD

Trong hình 3.2(b), pha phân tích gồm có bước phân tích yêu cầu tiếp theo sau là một bước lựa chọn. Trong phân tích yêu cầu, thành phần khái niệm hay logical được xác định, còn trong bước lựa chọn thì tập các mẫu cần cho các thành phần logical sẽ được lựa chọn. Trong pha thiết kế, các biểu đồ mẫu được xây dựng như trong hình 3.2(c). Pha làm mịn thiết kế tạo ra biểu đồ lớp tối ưu như hình 3.2(d).

Trong các phần nhỏ sau, chúng ta sẽ mô tả ngắn gọn về các bước trong ba pha nói trên. Các phần tiếp theo sẽ mô tả chi tiết về mỗi pha một cách đơn giản và có minh họa bằng các ví dụ. Phần áp dụng từng tiến trình được minh họa trong ví dụ cụ thể trong chương sau .



***Hình 3.2**. Qui trình POAD: (a) Bao gồm tất cả các pha, (b) Phân tích, (c) thiết kế, và (d) Sự lọc thiết kế [2]*

3.1. Pha phân tích

Tổng quan

Tương tự như bất kỳ một phương pháp luận phát triển phần mềm nào, POAD bắt đầu bằng việc phân tích các yêu cầu của ứng dụng. Thông thường, có một mối quan hệ phụ thuộc chặt chẽ giữa kỹ thuật được sử dụng trong tiến trình phân tích và loại hình phương pháp luận phát triển. Điều đó là tất nhiên, vì tiến trình phân tích tạo ra các chế tác phần mềm mà sẽ được sử dụng để thiết kế và kiến trúc ứng dụng trong các giai đoạn phát triển tiếp theo. Vì thế, tiến trình phân tích hướng đến tạo ra các chế tác là thích hợp nhất cho pha thiết kế và các giai đoạn khác của tiến trình phát triển. Chẳng hạn, trong các phương pháp luận hướng đối tượng truyền thống, một tập các đối tượng phân tích thường là một trong những đầu ra của tiến trình phân tích. Còn trong phương pháp luận POAD thì đầu ra của tiến trình phân tích là tập các mẫu được lựa chọn sẽ được sử dụng trong thiết kế ứng dụng.

Trong phần này, chúng ta tập trung giới thiệu giai đoạn phân tích của phương pháp luận POAD. Trong pha này, trước hết phân tích các yêu cầu của ứng dụng để xác định những vấn đề cần phải giải quyết, và qua đó nhận dạng các mẫu thiết kế được sử dụng. Đầu vào của pha này là các yêu cầu thu thập được từ người dùng hệ thống hoặc từ các chuyên gia trong lĩnh vực này. Ngoài ra, đầu vào cho pha phân tích là cơ sở dữ liệu các mẫu thiết kế mà có thể là cơ sở dữ liệu các mẫu thiết kế có mục tiêu chung, hoặc các mẫu của một lĩnh vực cụ thể mà chúng ta sẽ thảo luận sau trong chương này.

Giai đoạn phân tích bao gồm các hoạt động chính sau:

- Phân tích các yêu cầu để xác định các vấn đề cần giải quyết và phân chia ứng dụng thành một tập các thành phần logic.
- Làm quen bước đầu với cơ sở dữ liệu để biết được các mẫu đang tồn tại
- Tìm và lấy ra các mẫu từ cơ sở dữ liệu miền cụ thể để chọn một tập các mẫu ứng viên theo một cách tự động.
- Lựa chọn các mẫu từ tập mẫu ứng viên để sử dụng trong tiến trình thiết kế.

Mục đích của giai đoạn này là xác định tập các mẫu thiết kế sẽ sử dụng trong thiết kế ứng dụng. Bắt đầu từ các yêu cầu về chức năng của ứng dụng và một cơ sở dữ liệu mẫu thiết kế, các sản phẩm của giai đoạn này bao gồm:

- Tập các mẫu được các nhà phân tích lựa chọn để dùng trong phát triển ứng dụng.
- Sự hợp lý của việc lựa chọn các mẫu này.
- Những vấn đề của ứng dụng cụ thể được xác định thông qua phân tích các yêu cầu của ứng dụng.
- Tài liệu trình bày vì sao mà các mẫu được chọn tiên đoán là hướng đến các vấn đề đặt ra.

Phân tích trong giai đoạn này không cần quan tâm đến các chi tiết bên trong mẫu. Chẳng hạn, khi phân tích không cần đi sâu vào các chi tiết cụ thể, sơ đồ lớp bên trong, chuỗi tương tác giữa các thành phần, sự thay đổi của các trạng thái cấu trúc của mẫu. Ở giai đoạn này không cần thiết phải hiểu các mẫu giải quyết vấn đề như thế nào, thay vào đó cần quan tâm đến việc những mẫu nào được lựa chọn, tại sao lại lựa chọn những mẫu đó và so sánh tính ưu việt của các mẫu này với các mẫu ứng viên khác. Hình 3.3 minh hoạ toàn cảnh cách tiếp cận phân tích.

Đi theo cùng một mục đích, tiến trình và sản phẩm được sử dụng, chúng ta sẽ mô tả từng hoạt động trong giai đoạn này. Ngoài ra chúng ta bổ sung một số lời khyên và các hướng dẫn cần thiết cho việc phân tích. Các chú thích trong hình 3.3. có ý nghĩa tương tự như những gì đã dùng. Chúng ta sử dụng nhiều đường biên nét mảnh cho các phần tử của biểu đồ để biểu diễn các chế tác đầu vào và đầu ra. Trong các phần sau, chúng ta sẽ giải thích rõ từng hoạt động trong bốn hoạt động cơ bản: phân tích yêu cầu (*requirements),* làm quen bước đầu với cơ sở dữ liệu mẫu (*acquaitance*), tìm và lấy ra các mẫu ứng viên (*retrieval*) và lựa chọn các mẫu (*selection*) cùng các chế tác được tạo ra/tổng hợp của mỗi hoạt động. Sản phẩm của pha này là một tập các mẫu được các nhà phân tích ứng dụng lựa chọn.

a. Mục đích

Mục đích của pha này là để phân tích các yêu cầu của ứng dụng và quyết định các mẫu thiết kế thích hợp nào sẽ được sử dụng trong việc thiết kế hệ thống.

b Tiến trình

Sử dụng các yêu cầu ứng dụng là đầu vào, ở các bước khác nhau nhà phân tích theo thứ tự sẽ quyết định tập hợp các mẫu thiết kế lấy từ thư viện của các mẫu ứng dụng cụ thể. Trong bước này, biểu đồ ca sử dụng của UML và biểu đồ tuần tự có thể được sử dụng để xác định các mẫu cần thiết hỗ trợ các tương tác này. Các mẫu này sẽ

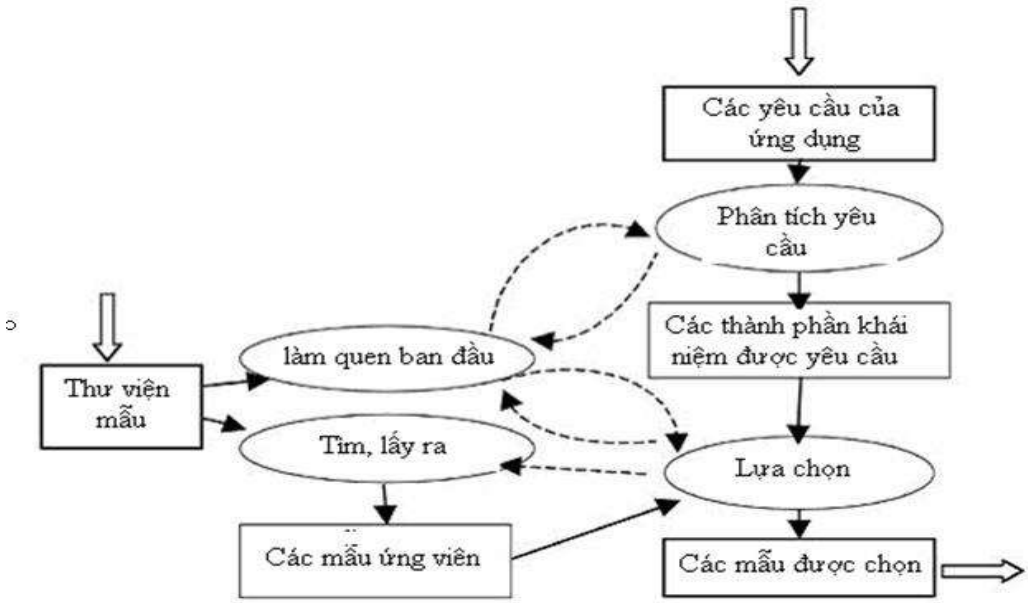
được sử dụng trong việc xây dựng thiết kế hệ thống. Ở mức này, nhà phân tích không quan tâm tới các chi tiết bên trong của các mẫu. Ví dụ, nó không cần biết được mẫu này giải quyết các vấn đề được đưa ra trong các yêu cầu của ứng dụng như thế nào. Thay vào đó, nhà phân tích có liên quan tới sự xác định xem một mẫu có thể được sử dụng hay không và tại sao nếu nó được sử dụng thì sẽ tốt hơn là sử dụng các mẫu khác trong thư viện hay là các giải pháp riêng. Nhà phân tích đầu tiên phải xác định các gói khái niệm hay là các thành phần mà họ lấy làm đại diện cho các thuộc tính chức năng của hệ thống. Các trách nhiệm chức năng này xác định vai trò toàn diện của thành phần khái niệm và được sử dụng để hướng dẫn tiến trình lựa chọn mẫu bằng cách định danh các trách nhiệm cần thiết được yêu cầu từ một mẫu cụ thể để triển khai thành phần khái niệm này.

Là một phần của pha phân tích, các nhà phân tích tìm kiếm mẫu phân tích từ danh sách các mẫu ứng viên. Để tìm kiếm một cách có hiệu quả, thư viện miền cụ thể của các mẫu sẽ tốt hơn là thư viện cho mục đích chung. Tương tự việc quản lý thư viện các thành phần phần mềm trước đây, tiến trình này bao gồm việc *làm quen bước đầu* và *trích rút* thông tin.

Sự định nghĩa không rõ ràng các thành phần khái niệm sẽ làm cho việc *làm quen bước đầu* không có hiệu quả. Nó sẽ gây ra sự khó khăn khi xây dựng mẫu thiết kế để thực thi chức năng đã được xác định trong thành phần khái niệm. Do đó, bước này thường dựa vào những tìm hiểu các thư viện mẫu có sẵn được lựa chọn. Trên thực tế, một vài mẫu trong thư viện có thể thúc đẩy nhà phân tích xác định các thành phần khái niệm được thiết kế một cách dễ dàng và được thực thi bằng cách sử dụng các mẫu đó. Ví dụ, một ngôn ngữ mẫu của các máy trạng thái và biểu đồ trạng thái đã thúc đẩy nhà phân tích phát triển mô hình (hay kiến trúc) dựa trên các thành phần cộng tác có hành vi được mô hình hóa thích hợp trong dạng đặc tả trạng thái. Mô hình hóa hướng đối tượng thời gian thực của Selic, Gullekson và Ward[1994] có các mô hình kiến trúc dựa trên những kĩ thuật như vậy. Sự làm quen với nội dung của thư viện là một hoạt động mà nhà phân tích sẽ cân nhắc đến trong pha phân tích.

Tiến trình trích chọn xác định việc lựa chọn mẫu như thế nào từ danh mục. Sự trích rút các tài sản phần mềm từ thư viện là một chủ đề nghiên cứu lớn hàng chục năm nay. Nhiều kĩ thuật đã được đưa ra, và nhiều vấn đề khác nhau đã được xác định

bao gồm các tiêu chuẩn phù hợp, việc trích rút chính xác hoặc xấp xỉ và các tiêu chuẩn đánh giá là đúng đắn, gọi lại, tỉ lệ được đưa ra, độ phức tạp, và sự tự động hóa [Mili et al, 1998]. Các mẫu thiết kế cũng không phải là ngoại lệ. Tuy nhiên, trích rút các mẫu từ cơ sở dữ liệu có thể sẽ dễ dàng hơn bởi vì cộng đồng mẫu đã thực hiện công việc một cách hoàn hảo gắn với định nghĩa của các mẫu khi sử dụng một hoặc một số tiêu bản, làm cho nó dễ dàng trích rút tự động một cách dễ dàng hơn bằng cách tiếp cận vấn đề hay phần nội dung mẫu với vấn đề đã được giải quyết.



Hình 3.3. Pha phân tích [2]

c. Sản phẩm

Sản phẩm của pha này là một tập các mẫu thiết kế đã được các nhà phân tích ứng dụng lựa chọn. Chúng ta chú ý là, ở pha này không có bất cứ chi tiết cụ thể nào về mẫu được đưa ra, có nghĩa là hoặc không có mô hình lớp hoặc mô hình hành vi nào có trong bước phân tích này.

3.2. Pha thiết kế

Tổng quan

Trong phần này chúng ta giới thiệu về giai đoạn thiết kế trong phương pháp POAD. Trong giai đoạn này chúng ta sử dụng các mẫu trong tập mẫu đã được lựa chọn qua giai đoạn phân tích làm cơ sở xây dựng bản thiết kế đầu tiên. Các tài liệu của quá trình phân tích cũng được sử dụng để liên kết các mẫu với nhau nhằm phát triển

thiết kế của ứng dụng.Các tài liệu này chứa nhiều thông tin về cách thức giải quyết vấn đề của các mẫu và cách phân tích ứng dụng thành nhiều thành phần.

Giai đoạn thiết kế gồm ba hoạt động chính:

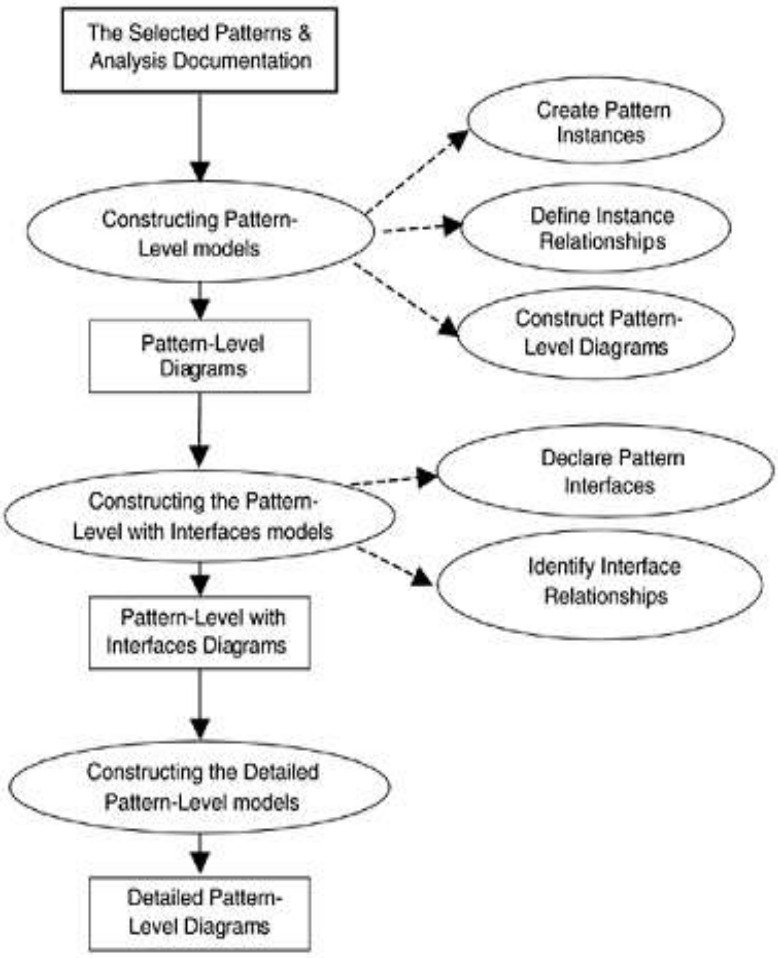
- Xây dựng sơ đồ mức mẫu thể hiện cấu tạo chung của các mẫu.
- Xây dựng sơ đồ chi tiết thể hiện mối quan hệ giữa các mẫu.
- Xây dựng sơ đồ chi tiết ứng dụng với cấu tạo bên trong của các mẫu.

Giai đoạn này tạo ra các mô hình thiết kế.Các loại mô hình thiết kế và các yếu tố sẽ được sử dụng trong chương này với mục đích phát triển mô hình thiết kế hướng mẫu.

Mục đích của giai đoạn thiết kế là sử dụng các mẫu thiết kế làm đơn vị cơ sở để tạo ra các bản thiết kế của ứng dụng. Bắt đầu từ tập các mẫu đã được lựa chọn trong giai đoạn phân tích. Chi tiết của giai đoạn thiết kế gồm :

- Thiết kế tổng quát ứng dụng từ một hoặc nhiều sơ đồ mức mẫu.
- Thiết kế giao diện dựa trên mối liên hệ giao diện giữa các mẫu.
- Thiết kế chi tiết ứng dụng dựa trên cấu tạo bên trong của các mẫu.

Như đã đề cập tất cả các mô hình thiết kế đều được lưu trữ. Khả năng truy suất cho phép người thiết kế có thể xem lại các mô hình thiết kế từ mức trừu tượng nhất trở xuống. Khả năng truy suất là một chức năng được hỗ trợ trong môi trường phát triển POAD.



Hình 3.4. Tiến trình pha thiết kế [2]

a. Mục đích

Mục đích của pha thiết kế này là phát triển thiết kế ứng dụng bằng cách cấu thành các mẫu đã được chọn từ pha phân tích. Một phần của pha thiết kế là các hoạt động trong pha này sẽ tạo ra các mô hình thiết kế ở các mức trừu tượng khác nhau. Các mô hình này có thể lần vết một cách dễ dàng từ các mức trừu tượng cao đến các mức trừu tượng thấp hơn.

b. Tiến trình

Đầu tiên, người thiết kế tạo ra các thể hiện của các mẫu đã được chọn trong pha phân tích và xác định quan hệ giữa các thể hiện này. Đây là ứng dụng trực tiếp của cách tiếp cận ghép chuỗi mẫu mà kết quả nằm trong khung nhìn lôgic mẫu mức cao. Sự hoạt động này bao gồm một vài các hoạt động con khác nhau như là tạo ra các mẫu bằng cách đưa mẫu thiết kế vào trong các chế tác thiết kế hữu hình, xác định các mối quan hệ giữa các mẫu, và cấu trúc các biểu đồ mức mẫu cho toàn bộ hệ thống và cho

từng hệ thống con riêng rẽ. Chia nhỏ hệ thống thành các hệ con sẽ thúc đẩy hoạt động cho các hệ lớn phức tạp.

Từ biểu đồ mức mẫu đã tạo ra, người thiết kế xử lý để nhận được biểu đồ mức mẫu cùng với các giao diện. Người thiết kế sẽ phân tích các quan hệ giữa các thể hiện mẫu và sau đó là lần vết các quan hệ này đến mối quan hệ thiết kế ở mức thấp hơn giữa các giao diện mẫu. Sau đó chúng ta sẽ xác định các giao diện mẫu và phân rã các quan hệ phục thuộc của mẫu thành các mối quan hệ mức thấp hơn giữa các giao diện mẫu. Kết quả là các biểu đồ mức mẫu với các giao diện và các biểu đồ mức mẫu được làm mịn.

Bắt đầu từ biểu đồ mức mẫu với các giao diện, người thiết kế xác định các chi tiết trong mẫu theo thuật ngữ của biểu đồ lớp, nhằm mục đích tạo ra một biểu đồ lớp ban đầu cho hệ thống, kết quả là một biểu đồ mức mẫu đã chi tiết được tạo ra.

c. Sản phẩm

Sản phẩm của pha này là các biểu đồ mức mẫu được chi tiết là mô hình thiết kế của ứng dụng như là một tập hợp các mẫu được thể hiện.

3.3. Pha làm mịn thiết kế

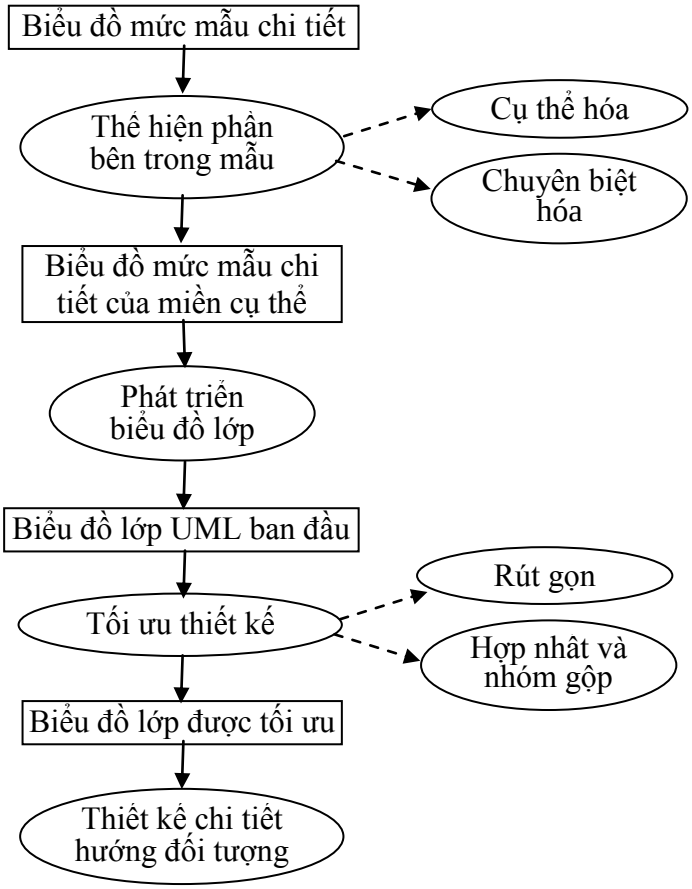
Tổng quan

Trong phần này, chúng ta thảo luận về pha làm mịn thiết kế của phương pháp luận POAD. Trong pha làm mịn thiết kế, chúng ta tạo ra biểu đồ lớp của hệ ứng dụng khi sử dụng biểu đồ lớp của các khối mẫu được xây dựng. Đầu vào của pha này là biểu đồ mức mẫu chi tiết đã xây dựng từ pha thiết kế. Những biểu đồ thiết kế này được sử dụng để mô hình hóa ứng dụng như một tập các mẫu, các mối liên kết và các quan hệ giữa chúng. Trong tiến trình thiết kế, nhà thiết kế đã nắm bắt được những chi tiết bên trong của mẫu thiết kế được chọn. Chi tiết này sẽ hữu ích trong mọi hoạt động của pha làm mịn thiết kế.

Pha làm mịn thiết kế có ba hoạt động chính:

- *Thể hiện phần bên trong mẫu*, mà ở đó chúng ta thêm bản chất của ứng dụng cụ thể vào thiết kế bên trong của các mẫu.
- *Phát triển các biểu đồ lớp*, mà ở đó chúng ta xây dựng biểu đồ lớp ban đầu của ứng dụng.

– *Tối ưu thiết kế*, mà ở đó chúng ta tối ưu biểu đồ lớp của ứng dụng bằng cách chồng lấp lên nhau các thể hiện của mẫu.



Hình 3.5. Tiến trình làm mịn thiết kế POAD [2]

Mục đích của pha này là tạo biểu đồ lớp của ứng dụng khi sử dụng biểu đồ lớp của mẫu và về miền cụ thể của ứng dụng. Bắt đầu từ tập biểu đồ mức mẫu chi tiết được phát triển trong pha thiết kế trước. Sản phẩm đưa ra từ pha làm mịn thiết kế bao gồm:

– Thể hiện ứng dụng cụ thể của mỗi mẫu thiết kế được sử dụng. Đó là mô tả mô hình thiết kế gốc của mẫu khi sử dụng ứng dụng cụ thể và đặt tên mẫu thích hợp.

– Các mô hình biểu đồ lớp ban đầu của thiết kế ứng dụng biểu diễn mô hình biểu đồ thiết kế của ứng dụng như một bộ lắp ghép lỏng lẻo từ các mẫu.

– Các mô hình biểu đồ lớp tối ưu cho ứng dụng mô tả mô hình biểu đồ lớp của ứng dụng một cách dày đặc và sâu sắc hơn. Chúng sẽ đước sử dụng mà được sử dụng trong pha thiết kế chi tiết và triển khai.

– Mọi mô hình thiết kế tạo ra trong pha này đều được lưu trữ. Cơ chế lần vết cho phép nhà thiết kế duyệt qua toàn bộ mô hình thiết kế và lần vết các thành phần tham gia mẫu từ mức này sang mức khác

Trong những phần tiếp theo, chúng ta sẽ thảo luận chi tiết một trong ba hoạt động chính của tiến trình thiết kế: *Thể hiện thiết kế bên trong mẫu, xây dựng các mô hình biểu đồ lớp ban đầu cho ứng dụng, và tối ưu biểu đồ lớp* để tạo ra thiết kế dày đặc và sâu sắc hơn.

a. Mục đích

Mục đích của pha này là phát triển biểu đồ lớp sâu sắc, đậm đặc cho ứng dụng để các nhà phát triển để triển khai.

b. Tiến trình

Người phân tích bắt đầu với kết quả của pha thiết kế: biểu đồ mức mẫu được chi tiết hóa. Họ sẽ thể hiện một cách cụ thể cho từng mẫu trong ngữ cảnh của ứng dụng. Nó gồm sự lựa chọn tên cho các thành phần tham gia mẫu sao cho có đầy đủ ý nghĩa trong ngữ cảnh của ứng dụng, xác định các tên ứng dụng cụ thể cho các thao tác trong các lớp mẫu. Các hoạt động này có thể được khái niệm hóa, được phân loại cụ thể hóa, đặc tả, xác định phạm vi và tầm nhìn của thiết kế. Các hành động này được thảo luận trong quyển sách "Design Components : Towards Software composition at the Design Level" của Keller và Schauer[1998]. Kết quả là thiết kế mức chi tiết mẫu cho ứng dụng cụ thể được tạo ra.

Khi sử dụng các biểu đồ mức chi tiết mẫu, người thiết kế sẽ phát triển biểu đồ lớp ban đầu của thiết kế ứng dụng. Để phát triển một biểu đồ lớp từ các biểu đồ mức mẫu chi tiết, người thiết kế phải lần vết tất cả các mối quan hệ giữa các mẫu tại các miền xác định của mức mẫu chi tiết đến mối các quan hệ lớp, như là các liên kết, các sự phụ thuộc, tổng quát hóa …Thiết kế kết quả được nhìn nhận như là biểu đồ các lớp UML ban đầu, có thể là đậm đặc và sâu sắc.

Biểu đồ lớp thu được từ sự kết hợp của các mẫu với nhau trong thiết mức cao, không đậm đặc và sâu sắc. Nó có nhiều lớp trừu tượng lặp lại theo cùng một mục đích nhờ sử dụng cùng một mẫu trong nhiều thể hiện. Nó có nhiều lớp có trách nhiệm tầm thường. Người thiết kế sử dụng việc rút gọn, hợp nhất, và nhóm gộp để làm tối ưu các biểu đồ thiết kế. Sự rút gọn là có thể để lấy đi các lớp trừu tượng được lặp lại, là kết

quả từ việc sử dụng cùng một mẫu trong một vài thể hiện trong cùng một thiết kế. Thí dụ như sử dụng hai thể hiện ứng dụng cụ thể như mẫu FeedbackObserver và mẫu SensorObserver của mẫu Observer.

Bắt đầu với biểu đồ các lớp được làm mịn và cách nhìn thiết kế hướng mẫu, người thiết kế có thể tiếp tục phân tích hệ thống theo các mô hình thiết kế hướng đối tượng truyền thống như: sự cộng tác, tương tác, và các mô hình biểu đồ trạng thái.

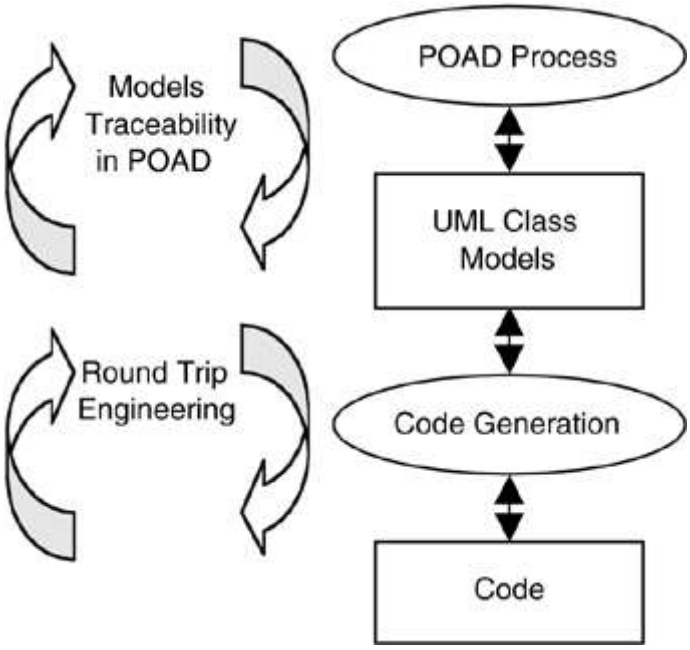
c. Sản phẩm

Sản phẩm của pha này là biểu đồ lớp được tối ưu cho ứng dụng.

3.4. Sự sinh mã nguồn của POAD

Tiến trình POAD phần lớn có liên quan đến tiến trình phân tích và thiết kế. Nó không quan tâm đến các phong cách lập trình cụ thể hay ngôn ngữ lập trình hướng đối tượng sử dụng trong các pha thực thi. POAD tập trung vào việc tạo các mô hình thiết kế từ các thiết kế được chi tiết hóa và sự thực thi chúng có thể được sinh ra. Ta có thể chú ý, đầu ra của tiến trình là các biểu đồ lớp đơn giản, nó chứa các mô hình hướng đối tượng truyền thống bên trong phạm vi các lớp, các sự liên kết, kế thừa…Do đó, các mô hình đã tạo ra từ POAD có thể được triển khai bằng cách sử dụng nhiều ngôn ngữ lập trình hướng đối tượng.

Bây giờ ta hãy chú ý đến câu hỏi: những gì sẽ xảy ra nếu nhà phát triển thay đổi mọi thứ ở mức mã. Đây là vấn đề phổ biến trong bất cứ môi trường mô hình hóa nào. Làm thế nào để khắc phục khoảng trống giữa lập trình với các mô hình phân tích và thiết kế. Rất nhiều công cụ mô hình hóa UML hỗ trợ các nguyên lý kĩ nghệ quay vòng, điều này giúp mã nguồn luôn được giữ đồng bộ với các mô hình đã sử dụng để tạo ra chúng. Thí dụ, tất cả các biểu đồ lớp có thể được giữ đồng bộ với mã nguồn bị thay đổi bởi người phát triển, tất cả mã nguồn có để được tạo lại trong sự xem xét các phần mã mà được điều khiển bởi các công cụ mô hình hóa với các phần dành cho người phát triển thực hiện.



Hình 3.6. Đồng bộ hóa giữa mã nguồn và các mô hình [2]

Với các cơ chế của kĩ nghệ này, cùng với kĩ nghệ dịch ngược được đặt vào cùng một chỗ, chúng ta có thể nói rằng, mã nguồn đã được giữ đồng bộ với các mô hình thiết kế ở mức thấp như là các biểu đồ lớp. Giờ đây câu hỏi của chúng ta lại là: Tác động đến các mô hình được tạo ra bởi POAD ở mức phân tích và thiết kế như thế nào? Các chương tiếp theo sẽ mô tả kĩ tiến trình và các ví dụ nguyên cứu cụ thể, ta sẽ biết POAD có các cơ chế lần vết mà khả năng theo dấu từ các mô hình thiết kế ở mức thấp (các biểu đồ lớp) đến các mô hình ở mức cao hơn (các biểu đồ mức mẫu,các biểu đồ giao diện mẫu…) và ngược lại. Các cơ chế lần vết có thể được coi là kĩ thuật quay vòng ở mức phân tích và thiết kế, nó được cung cấp một cách tự động bởi các công cụ hỗ trợ phương pháp luận POAD. Với các kĩ thuật quay vòng trong POAD đã nối giữa các mô hình phân tích và thiết kế mẫu với các mô hình biểu đồ lớp. Các kĩ thuật quay vòng giữa các biểu đồ lớp với phần mã nguồn cũng được hỗ trợ trong các công cụ mô hình hóa UML hiện nay. Chúng ta có một sự đồng bộ hóa hoàn toàn giữa phần mã với các mô hình thiết kế ở mức cao hơn, minh họa trong hình vẽ trên

3.5. Những lợi ích và hạn chế

Có lẽ ta đang tự hỏi, có phải POAD thực sự là hữu ích? Ta thực sự cần xây dựng các ứng dụng như những cấu thành của mẫu? Ta chỉ cần sử dụng các kĩ thuật hướng đối tượng truyền thống? Hay ta cần tập trung vào các lợi ích của việc xây dựng các ứng dụng sử dụng POAD và thảo luận một vài hạn chế của nó.

3.5.1. Lợi ích

- Những thiết kế hướng đối tượng là kết quả của việc áp dụng tiến trình phát triển POAD, giảm bớt các hành động chế tác lại. Đó là vì một phần của tiến trình tái chế đã được thực hiện trong chính các mẫu. ”Các mẫu thiết kế nắm bắt được nhiều cấu trúc là kết quả của sự tái chế”. Rất nhiều phương pháp luận của phát triển hướng đối tượng truyền thống ta sẽ được làm việc lại trong phân tích và cải tiến nó sâu sắc: POAD, bởi vì sử dụng các thiết kế đã được cải tiến từ rất nhiều sự ứng dụng thành công.

- *Việc làm tài liệu tốt hơn của thiết kế ứng dụng/khung làm việc.* Các biểu đồ mức mẫu cung cấp khung nhìn trừu tượng ở mức cao của thiết kế. Các khung nhìn trừu tượng này như những tài liệu thiết kế cho ứng dụng, do đó chúng ta có thể cải tiến khả năng hiểu được của thiết kế. Ở rất nhiều quy trình phát triển hướng đối tượng khác, các gói và các hệ thống con được sử dụng để nhóm các lớp lại với nhau. Những cái đưa vào cùng một gói thường là quyết định thiết kế quan trọng. Trong POAD, phần bên trong mẫu là đã được biết, chính là những mô hình lớp lớp được thiết kế tốt.

- Rủi ro trong việc sử dụng sai các khung làm việc thiết kế hướng đối tượng được giảm bớt nhờ tác dụng của sự cải tiến khả năng hiểu được của các thiết kế, các lớp. Kết quả của các tầng được đưa vào gần đây và tính lần vết đến các mức thiết kế thấp hơn cho chúng ta nhận ra rằng, thiết kế là dễ hiểu hơn.

- Sử dụng các mẫu như các khối xây dựng đem lại khung nhìn kiến trúc của ứng dụng, bởi vì chúng ta không cụ thể được các vấn đề thực thi tại khung nhìn trừu tượng mức cao của giao diện các mẫu. Tuy nhiên, các thành phần kiến trúc của chúng trên thực tế là các phần tử thiết kế bên trong giới hạn của sự cộng tác giữa các lớp. Do đó chúng ta có khả năng lần vết đến khung nhìn kiến trúc từ thiết kế mức thấp đến các khung nhìn thực thi.

- Những thiết kế hướng mẫu cho chất lượng cao hơn các thiết kế hướng đối tượng, bởi vì chúng sử dụng các mẫu thiết kế đã được làm tài liệu chất lượng cao, các giải pháp đã được kiểm trứng cho các vấn đề thiết kế.

- “Các mẫu triển khai” khi thực thi các mẫu thiết kế, những người lập trình tạo, mở rộng và chỉnh sửa các lớp ở khắp nơi của phần mềm. Nó có thể tạo ra sự duy trì chính và khả năng lần vết, vì các lập trình viên “đánh mất khả năng nhìn của các mẫu ban đầu”. Phương pháp POAD quan tâm đến vấn đề này bởi với các khung nhìn mức mẫu được thêm vào các lớp trên các pha thiết kế mức cao, các mẫu được xử lý như là các thực thể thiết kế.

- Phương pháp POAD dựa trên thành phần. Nó chuyển các nỗ lực phát triển từ việc tạo ra thiết kế đến việc lựa chọn, làm thích nghi và ghép nối các mảnh thiết kế. Mặt khác, nó không phải là sự cắm, lắp ghép và vận hành tiến trình như cách hiểu thông thường từ sự phát triển phần mềm dựa trên thành phần, bởi vì sự làm thích nghi và sự kết hợp các thành phần là các mảnh thiết kế. Người thiết kế vẫn phải làm việc trên chi tiết cụ thể bên trong và các đặc tả triển khai.

- POAD cung cấp một giải pháp cho vấn đề khả năng lần vết được thảo luận ở chương II. Giải pháp đơn giản sử dụng các mô hình thiết kế và nắm bắt khung nhìn ở mức cao của hệ thống. Sử dụng các mô hình để nắm bắt những chi tiết và cung cấp một liên kết giữa khung nhìn mức cao và các chi tiết mức thấp hơn. Trong POAD, những cấu trúc của các mẫu là lâu bền, có nghĩa là tất cả các lớp vẫn được thể hiện các mức thiết kế.

3.5.2 Các hạn chế

POAD có một vài điều trở ngại:

- Sự hiểu được các mẫu vốn là các hành động tinh thần đặc biệt. Để đưa ra quyết định đúng đắn trong việc chọn lựa mẫu trong thiết kế hướng mẫu, người thiết kế phải hiểu được sự trả giá, động cơ thúc đẩy, sức mạnh và các mẫu liên quan. Rất khó biết được cái giá phải trả cho các lợi ích mà người thiết kế đạt được khi sử dụng mẫu được lựa chọn trong phạm vi chất lượng của giải pháp đã được kiểm chứng.

- Để trợ giúp quy trình POAD, hiện thời không đủ tài liệu của các kho mẫu.Yếu tố cần thiết là có các kho mẫu để lưu trữ các thư viện mẫu điện tử trong cơ sở dữ liệu. Những thư viện này có thành phần truyền thống, thư viện các vấn đề :các kĩ thuật trích rút thành phần và cấu trúc thư viện.

- POAD giả định rằng, các giao diện mẫu được xác định trước. Khái niệm các giao diện mẫu là mới, nhiều người sẽ thấy lạ. Từ đó chúng được sử dụng để sử dụng các mẫu tại mức biểu đồ lớp. Trên thực tế, chúng ta đã tìm thấy các tài liệu chứng minh phần lớn mẫu hoàn toàn nói đến thứ gì đó về đối tượng khách hay nhân tố bên ngoài sử dụng mẫu. Các ví dụ về giao diện được mô tả trong phụ lục A. Chúng ta vẫn phải thừa nhận rằng, yêu cầu một sự nỗ lực để xác định các giao diện mẫu cho các tài liệu đã có và chỉ dẫn trên mẫu. Nguyên tắc đa giao diện cần được phối hợp tốt.

CHƯƠNG 4: MẪU THIẾT KẾ VÀ ỨNG DỤNG

4.1. Mẫu kiến trúc- Builder

a. Vấn đề

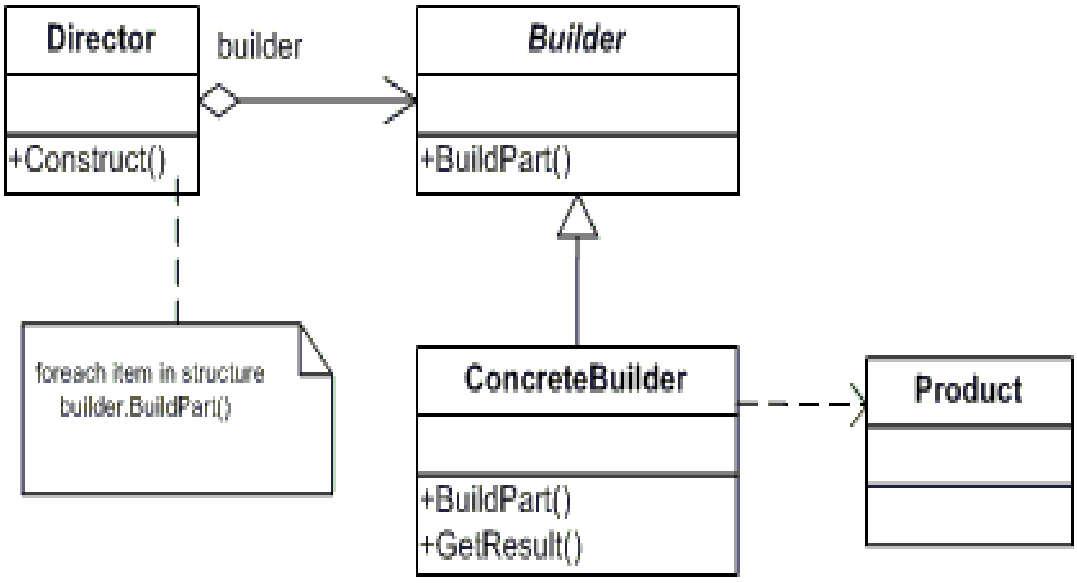
Module hóa các bước của việc xây dựng đối tượng phức tạp

Tăng tính độc lập việc xây dựng đối tượng của các lớp khác nhau

Các mẫu Builder phân cách xây dựng một đối tượng phức tạp từ đại diện của mình để quá trình xây dựng cùng một đại diện khác nhau có thể tạo ra

b. Giải pháp

Builder Pattern đơn giản hóa việc xây dựng các đối tượng phức tạp bằng cách chỉ quy định cụ thể loại và nội dung đối tượng yêu cầu. Quá trình xây dựng của đối tượng phức tạp do đó sẽ cho phép các đại diện khác nhau của đối tượng phức tạp được tạo ra bởi các nhà xây dựng khác nhau. Mỗi lớp trong số các đối tượng xây dựng cụ thể sẽ xây dựng một đại diện khác nhau của đối tượng phức tạp.



***Hình 4.1**. Giải pháp cho mẫu Builder [10]*

Các mẫu ***Builder*** bao gồm một, ***Builder ConcreteBuilder, Director*** và ***Product.***

Các đối tượng ***Director*** chịu trách nhiệm cho quá trình xây dựng của đối tượng phức tạp và tạo ra thực thể lắp ráp với giao diện ***Builder.*** Các đối tượng quy định các giao diện ***Builder*** để tạo các bộ phận của đối tượng phức tạp.

Product đại diện cho các đối tượng phức tạp được tạo ra bởi các đối tượng ***ConcreteBuilder.*** Sản phẩm này có nhiều phần được tạo riêng của các đối tượng ***ConcreteBuilder.***

Các đối tượng ***ConcreteBuilder*** tạo và lắp ráp các bộ phận tạo nên các ***Product*** thông qua giao diện ***Builder***.

c. Áp dụng

Builder Pattern được sử dụng khi:

Thuật toán tạo ra một đối tượng phức tạp và độc lập với các bộ phận mà trên các đối tượng được tạo ra

Hệ thống các nhu cầu để cho phép các đại diện khác nhau cho các đối tượng đang được xây dựng

Mẫu Builder giải quyết việc xây dựng một đối tượng phức tạp từ các thành phần nhỏ hơn. Việc xây dựng sẽ tiến hành qua từng bước với từng thành phần. Mẫu Builder làm tăng tính module hóa cho công việc xây dựng đối tượng của các lớp khác nhau và cũng như tính module hóa từng bước xây dựng một đối tượng.

4.2 Mẫu thay thế- Proxy

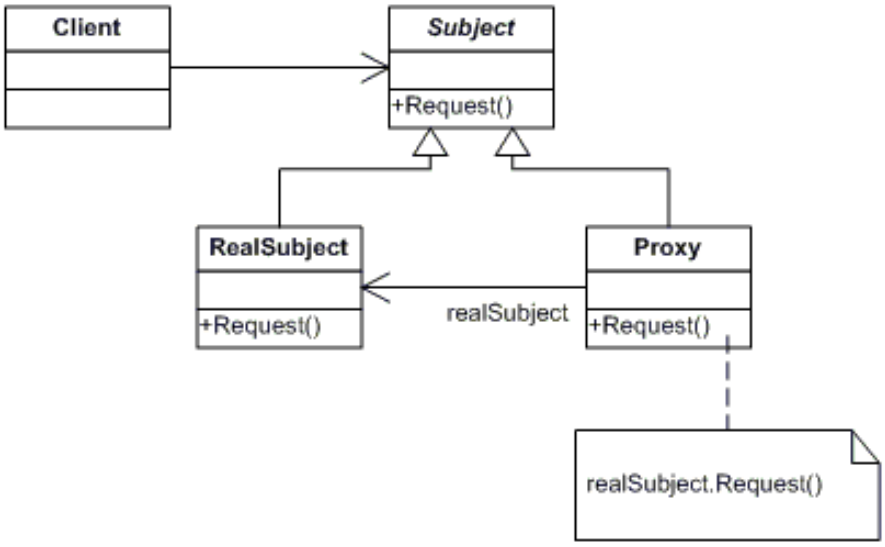
a. Vấn đề

Mẫu cấu trúc Proxy thường miêu tả một đối tượng đơn giản, được sử dụng cho việc thay thế vị trí của một đối tượng phức tạp khác mà có thể được chương trình gọi sau đó, như các đối tượng phức tạp trong một môi trường mạng hay các đối tượng được truy cập từ xa.

b. Giải pháp

Trong rất nhiều trường hợp, ta cần sử dụng các đối tượng phức tạp hoặc các đối tượng mà cần nhiều thời gian hay tài nguyên để khởi tạo hoặc tải về. Tuy nhiên các đối tượng này có thể không được sử dụng ngay lập tức hoặc việc khởi tạo hay tải về cần phải được hoàn thành trước khi sử dụng. Khi đó, ta có thể sử dụng mẫu cấu trúc Proxy,

cho phép tạo một đối tượng thay thế tạm thời vị trí của đối tượng thực đó. Đối tượng thay thế này có cùng giao diện với đối tượng thực và sẽ được thay thế bởi đối tượng thực khi đối tượng thực đã được khởi tạo hoặc tải về đầy đủ.



Hình 4.2. Giải pháp cho mẫu Proxy [10]

c. Áp dụng mẫu

Tải một đối tượng cần một lượng thời gian dài, ví dụ một hình ảnh có dung lượng lớn trên web.

Biểu diễn kết quả tính toán của một phép tính mà cần một lượng thời gian dài để hoàn thành, tuy nhiên các kết quả trung gian cần được hiển thị ngay lập tức trong khi quá trình tính toán vẫn được thực hiện.

Khởi tạo một đối tượng từ xa, ví dụ trên một máy khác trong một mạng network, và đặc biệt là trong một mạng mà việc truyền tải dữ liệu được thực hiện định kỳ.

Thẩm định việc truy cập các đối tượng có giới hạn quyền truy cập. Khi đó Proxy có thể cần thẩm định lại quyền truy cập của người dùng đến đối tượng đó.

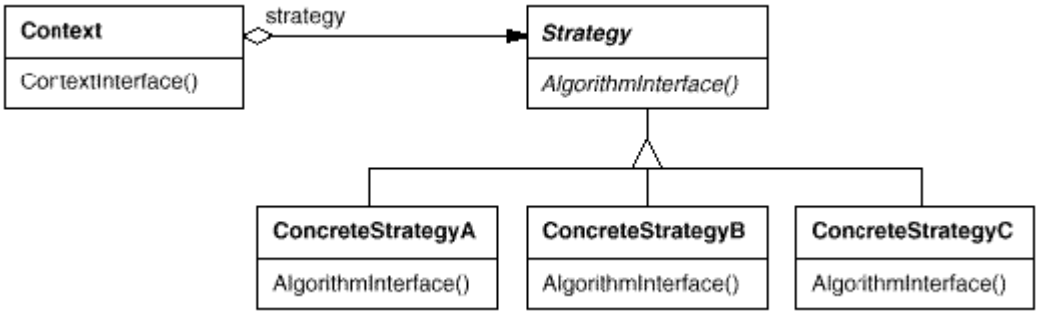
4.3. Mẫu chiến lược- Strategy

a. Vấn đề

Xác định cách giải quyết cho các đối tượng: Xác định một họ thuật toán và làm cho chúng có thể thay đổi được cho nhau, cho phép các thuật toán biến đổi độc lập với máy khách dùng nó.

b. Giải pháp

Cung cấp một cách để định hình một lớp với một cách xử sự trong nhiều cách xử xự của lớp đó. Cần thiết đưa ra các biến đổi khác nhau trong một giải thuật và chúng được thực hiện có thứ tự trong giải thuật.



Hình 4.3. Giải pháp cho Strategy[10]

Trong đó:

strategy ***:*** *Khai báo một lớp giao diện dùng chung cho tất cả các giải thuật.*

ConcreteStrategy: *Thực hiện giải thuật sử dụng giao diện chung do* ***strategy*** *cung cấp.*

Context ***:*** *được cấu hình với đối tượng ConcreteStrategy object, lưu trữ một tham chiếu đến đối tượng strategy, có thể định nghĩa một giao diện khác.*

c. Áp dụng

Mẫu strategy được sử dụng để lựa chọn các hình thức thể hiện khác nhau của 1 lớp nào đó như việc hỗ trợ ngôn ngữ hay các đơn vị tiền tệ cho 1 trang web

d. Các kết quả

Cung cấp một họ giải thuật có mối quan hệ với nhau

Cung cấp một lựa chọn cho các lớp con

Mẫu chiến lược cho phép lựa chọn một giải thuật trong họ giải thuật. Khi đó ***Context*** chuyển đổi giữa các chiến lược theo yêu cầu. Phương pháp này tránh được các câu lệnh điều kiện.

Khi có một số tham số khác nhau cần chuyển đổi giữa các thuật toán, mẫu chến lược cho phép phát triển thành phần giao diện context và mô hình Srategy.

4.4. Mẫu quan sát- Observer

a. Vấn đề

Các đối tượng nhận biết sự thay đổi từ một số đối tượng khác và có hành vi tức thời tương ứng với các thay đổi.

Đối tượng có dữ liệu thay đổi chỉ thực hiện thay đổi mà không quan tâm đến các đối tượng sử dụng dữ liệu

b. Giải pháp

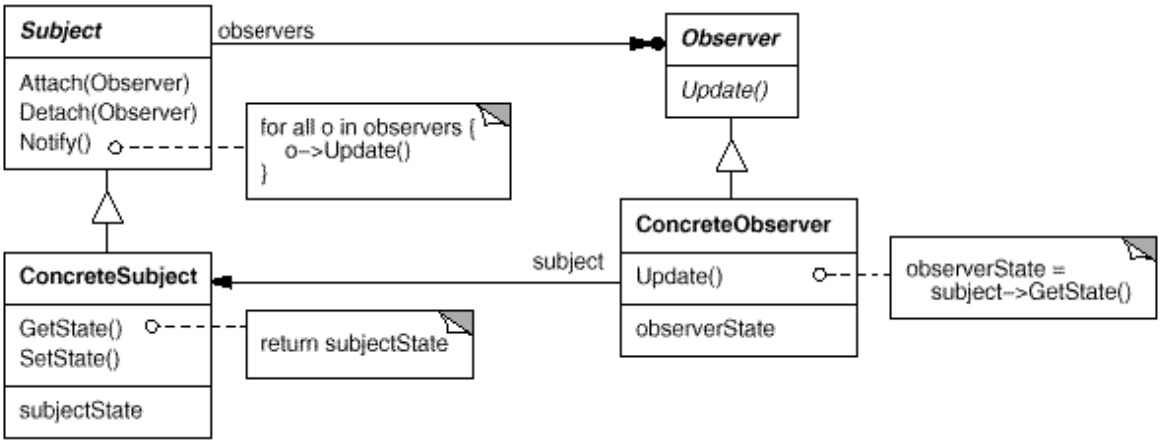
Phân đối tượng thành hai nhóm:

Subjects: là nhóm đối tượng được quan sát. Đối tượng thuộc nhóm này có dữ liệu được sử dụng bởi các đối tượng khác. Sự thay đổi dữ liệu của đối tượng nhóm này sẽ kéo theo các hành vi, trạng thái tương ứng của các đối tượng sử dụng dữ liệu.

Observers: nhóm đối tượng quan sát: gồm các đối tượng sử dụng dữ liệu của các đối tượng trên.

Nhóm quan sát luôn quan sát các đối tượng cụ thể mà nó đã đăng ký. Khi thấy có tín hiệu thay đổi trên đối tượng được quan sát, đối tượng quan sát tự nhận biết được thay đổi thông qua tín hiệu thay đổi của đối tượng được quan sát và có hành vi tương ứng.

Nhóm được quan sát không cần biết nó được quan sát bởi các đối tượng nào; khi có thay đổi, nó đưa tín hiệu thông báo đã thay đổi cho tất cả các đối tượng quan sát nó.



Hình 4.4. Giải pháp cho Observer[10]

c. Áp dụng mẫu

Áp dụng chung trong các ứng dụng có các đối tượng dữ liệu được sử dụng bởi nhiều đối tượng khác. Sự thay đổi trên các đối tượng dữ liệu yêu cầu hiệu ứng tức thì trên các đối tượng sử dụng dữ liệu.

Trong các ứng dụng soạn thảo, mẫu cũng được áp dụng để hiển thị trước các lựa chọn của người dùng như font chữ, kích thước.

Mẫu được sử dụng hiệu quả trong các ứng dụng xử lý ảnh, khi thay đổi dữ liệu hiệu chỉnh trên các form chọn màu, kích thước, cường độ sáng cần thấy ngay được kết quả trên ảnh xử lý.

Trong các ứng dụng quản lý cũng có thể sử dụng mẫu để hiển thị tức thời kết quả tính toán. Ví dụ việc chọn các mặt hàng, nhập số lượng thì tổng tiền phải được tính ngay trên hoá đơn…

d. Các kết quả

Tôn tại một cặp trừu tượng : thành phần quan sát và thành phần được quan sát. Tất cá các thành phần quan sát đều được lập danh sách trong đối tượng được quan sát, và chúng thích nghi với từng lớp giao diện quan sát trừu tượng đơn giản.

Cung cấp sự truyền kết nối rộng rãi: Sự thay đổi tự động truyền đi tới tất cả các đối tượng và được bổ sung vào thành phần quan sát.

Bất ngờ cập nhật: Lý do chủ yếu là do các thành phần được quan sát không có thông tin về các thành phần quan sát do đó chúng không thấy được giá trị thay đổi cuối cùng của thành phần quan sát. Một phép toán trên thành phần quan sát có thể là nguyên nhân dẫn đến sự cập nhật tới thành phần được quan sát và tất cả các đối tượng khác phụ thuộc vào chúng.

4.5. Mẫu cấu trúc Adapter

a. Vấn đề

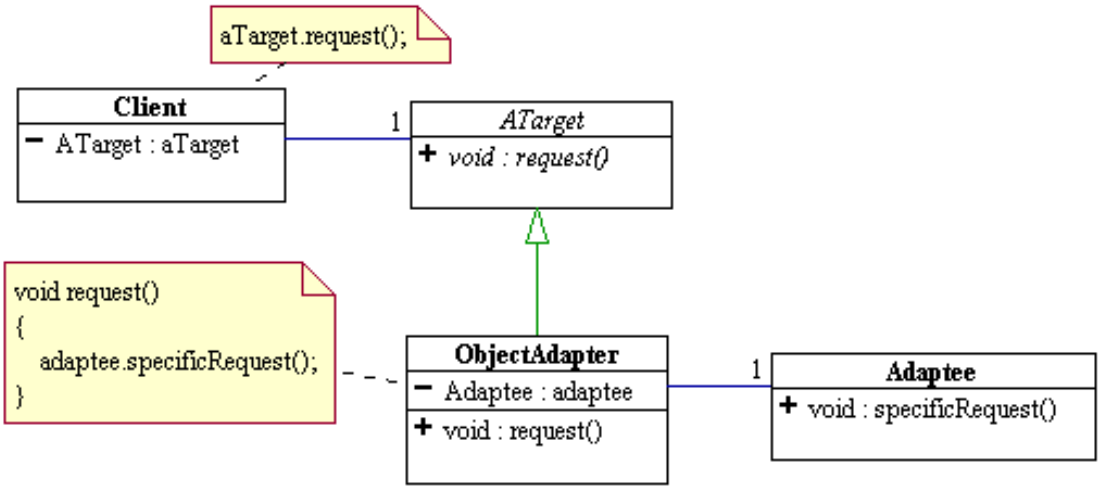
Do sự phát triển đa dạng của công nghệ đưa đến nhiều hệ thống thư viện lớp khác nhau phục vụ cho một mục đích, do đó những đoạn mã nguồn sử dụng trực tiếp các lớp của một thư viện sẽ không thể dùng lại một cách độc lập với thư viện đó. Và mẫu Adapter được dùng để tạo một giao diện lập trình không phụ thuộc một thư viện lớp cụ thể

b. Giải pháp

Có hai cách để cài đặt một mẫu cấu trúc Adapter.

Cách thứ nhất là sử dụng việc thừa kế. Trong cách này, trước hết ta viết một lớp đối tượng với một giao diện nào đó, và giao diện này có thể không phù hợp với mong muốn của chương trình tại một vị trí nào đó. Sau đó ta viết một lớp con thừa kế từ lớp cha này, và cài đặt các giao diện cho lớp con theo các yếu cầu hay mong muốn của chương trình tại vị trí cần sử dụng.

Cách thứ hai để cài đặt một mẫu cấu trúc Adapter là việc bao chứa một đối tượng nguồn có giao diện không phù hợp với yêu cầu của chương trình trong một đối tượng khác. Và trong đối tượng mới này, ta cài đặt các giao diện theo yêu cầu của chương trình, sau đó cài đặt các phương thức để truyền gọi việc thực thi đến đối tượng nguồn.



Hình 4.5. Giải pháp cho Adapter[10]

Với cách cài đặt thứ nhất, mẫu cấu trúc Adapter sử dụng việc thừa kế đối tượng để tạo nên cấu trúc của chương trình, nên thuộc vào loại mẫu cấu trúc lớp, và được gọi là mẫu cấu trúc Adapter lớp. Cách cài đặt thứ hai sử dụng việc quan hệ, tổ hợp các đối tượng nên thuộc vào loại mẫu cấu trúc đối tượng, và được gọi là mẫu cấu trúc Adapter đối tượng.

c. Áp dụng

Mẫu cấu trúc Adapter có thể được sử dụng để tạo các giao diện tương ứng của một lớp đối tượng sao cho phù hợp với yêu cầu của chương trình tại các vị trí khác nhau, làm cho chương trình dễ dàng hơn trong việc thao tác các đối tượng.

Cụ thể, mẫu cấu trúc Adapter được sử dụng để chuyển đổi các giao diện lập trình của một lớp đối tượng thành giao diện của một lớp đối tượng khác. Từ đó, các lớp đối tượng không liên quan với nhau vẫn có thể kết hợp để thực thi cùng nhau trong cùng một chương trình ứng dụng.

d. Kết quả

Có thể hiểu một cách đơn giản là: mẫu cấu trúc Adapter cho phép viết một lớp đối tượng với một giao diện nào đó, sau đó tạo khả năng giao tiếp với các lớp đối tượng khác với một giao diện khác.

Chuyển đổi giao diện của một số lớp thành một giao diện *mà* một số lớp khác hiểu được. Các AdapterPattern cho phép các lớp với giao diện không tương thích làm việc cùng nhau.

4.6. Ứng dụng

4.6.1. Builder- Gợi ý cách lựa chọn thiết bị máy tính

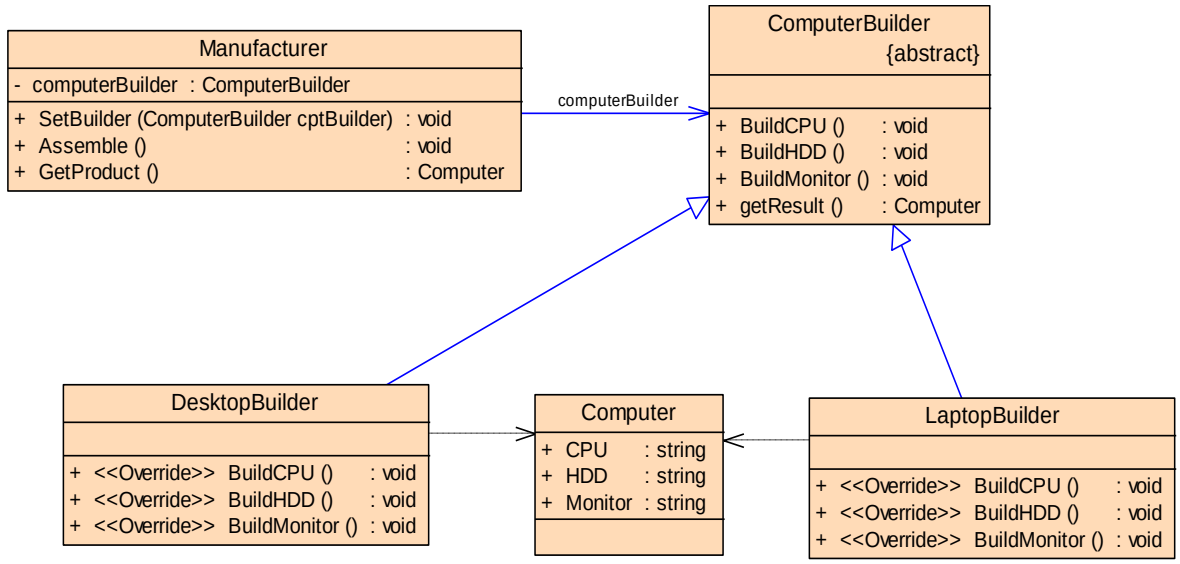
Bài toán đặt ra lúc này là gợi ý cách lựa chọn một số thiết bị phù hợp với cấu hình của hai loại máy tính Desktop và Laptop theo ý muốn. Tạo ra hai loại đối tượng độc lập của một đối tượng chung Computer, và hai đối tượng ấy kế thừa từ đối tượng chung, đồng thời cần có đối tượng thông qua lớp cha quy định loại và nội dung cho hai đối tượng con. Từ việc nghiên cứu và làm quen với các mẫu thiết kế em nhận thấy rằng Builder có thể giải quyết bài toán này một cách nhanh chóng. Dựa vào biểu đồ mức mẫu của Builder (hình 4.1) chúng ta sẽ lần lượt xây dựng biểu đồ chi tiết ứng dụng gọi là biểu đồ l cho bài toán.

Gợi ý lựa chọn các thiết bị phù hợp cho từng loại máy tính.

Lớp ComputerBuilder là lớp trừu tượng chứa phương thức getResult() trả về một đối tượng thuộc lớp Computer. Các phương thức BuildCPU(), BuildHDD(), BuildMoninor() sẽ xây dựng từng phần của Computer.

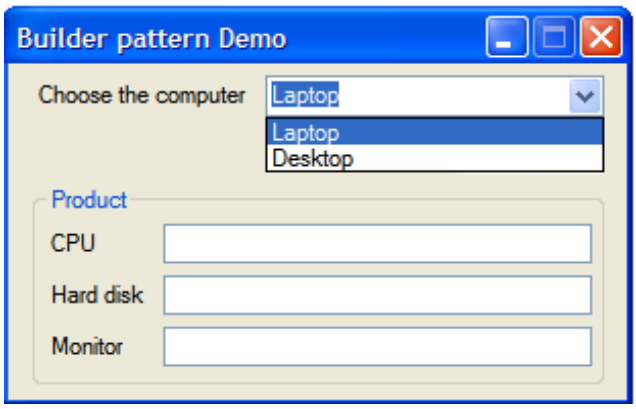
Hai lớp DesktopBuilder và LaptopBuilder thừa kế từ lớp ComputerBuilder sẽ cài đặt và override các phương thức BuildCPU(), BuildHDD(), BuildMoninor() của lớp cha.

Lớp Manufacturer là lớp client chứa reference computerBuilder kiểu ComputerBuilder và sẽ thực hiện xây dựng Computer bằng cách gọi hàm getResult() của computerBuilder.



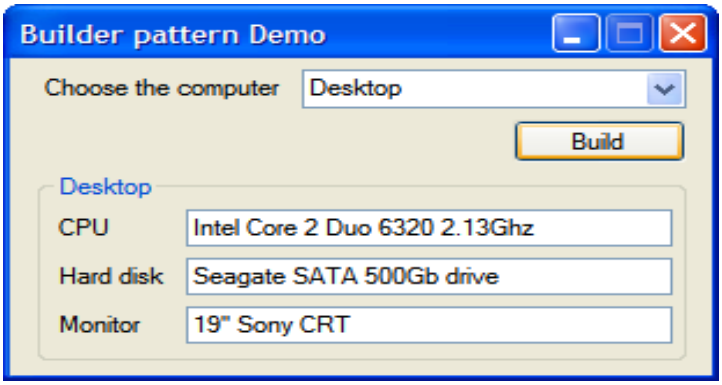
Hình 4.6. Biểu đồ lớp

- Giao diện chương trình: người dùng chọn loại máy tính cần lắp ráp, sau đó ấn nút Build để chương trình xây dựng máy tính từ các thành phần cho phù hợp.



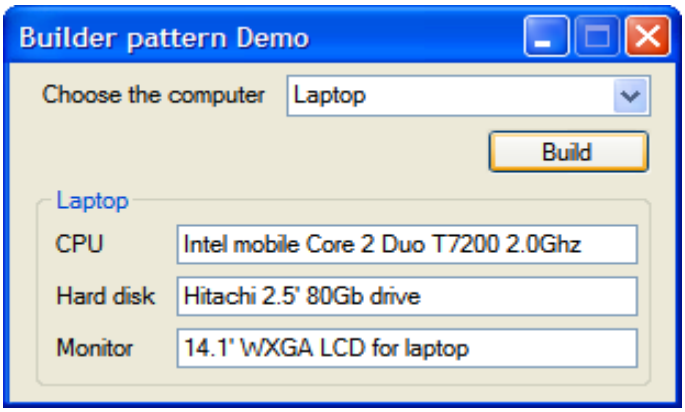
Hình 4.7. Giao diện chính

- CPU loại thường, ổ cứng máy để bàn, màn hình CRT Nếu chọn Desktop: Máy tính sẽ được lắp từ các thành phần phù hợp với máy để bàn:



Hình 4.8. Giao diện cho Desktop

- Nếu chọn Laptop: Máy tính sẽ được lắp từ các thành phần phù hợp với laptop: CPU mobile, ổ cứng 2.5 inches cho máy laptop, màn hình LCD

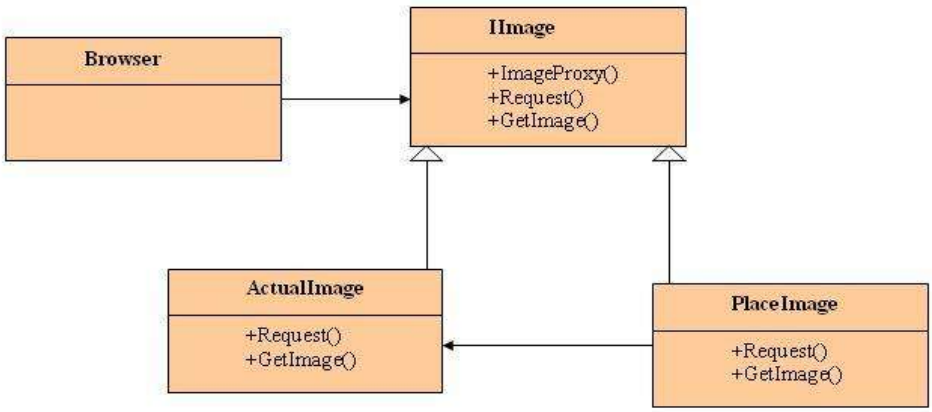


Hình 4.9. Giao diện cho Laptop

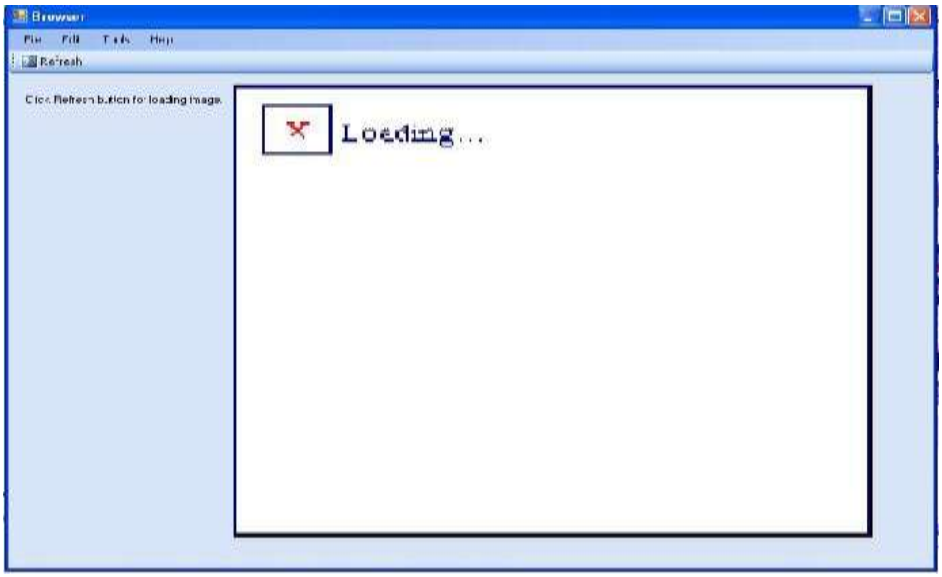
4.6.2. Proxy- Xây dựng chương trình tải ảnh

Xây dựng một trình duyệt đơn giản để tải file ảnh có dung lượng lớn. Hình ảnh phải được tải về đầy đủ trước khi hiển thị, trong quá trình tải ảnh thì một hình ảnh tạm thời được hiển thị. Khi tải xong nội dung hình ảnh được hiển thị trên Frame được xây dựng từ trước. Yêu cầu này hoàn toàn phù hợp với mẫu Proxy. Dựa vào biểu đồ mức mẫu(hình 4.2) ta xây dựng biểu đồ chi tiết ứng dụng như sau:

- Browser: yêu cầu từ trình duyệt
- ImageProxy: proxy để tải hình ảnh
- IImage: giao diện lớp đối tượng hình ảnh
- ActualImage: lớp đối tượng hình ảnh thực tế
- PlaceImage: lớp đối tượng hình ảnh thay thế



Hình 4.10. Biểu đồ lớp



Hình 4.11. Giao diện

4.6.3 Storage Explorer

4.6.3.1. Ý tưởng

Gama và cộng sự đã đề xuất 23 mẫu thiết kế cơ sở thường được gọi là mẫu thiết kế GOF , tuy nhiên hầu hết các ví dụ minh họa sử dụng riêng biệt cho mỗi mẫu. Và StorageExplorer là một ứng dụng nhỏ được thiết kế và sử dụng một số mẫu thiết kế cùng nhau. Trong khi mục đích chính của chương trình này là để cho người đọc hiểu thêm về các mẫu thiết kế. Các ứng dụng được sử dụng để đi sâu tìm hiểu các thành phần bên trong một tập tin được lưu trữ trên máy tính. Các mẫu thiết kế được sử dụng là: Chiến lược, Observer, Adapter…

Ý tưởng của ứng dụng này là khi chúng ta cần một ứng dụng tương tự như Windows Explorer có thể hiển thị cho chúng ta những thành phần bên trong kích

thước tập tin lưu trữ trong máy tính . Ví dụ, trong thư mục Program Files, thư mục sử dụng những gì mà kích thước của không gian, bao gồm cả những con số tỷ lệ phần trăm. Chúng ta cũng muốn biết làm thế nào kích thước tổng số những file như mp3 của chúng ta trong ổ đĩa so với các file jpg của chúng ta. Những thông tin muốn xem đó thể hiện như sau:

```
C:\Program Files:
        Microsoft           445,123 KB          43%
        Adobe               234,744 KB          25%
        Symantec             98,906 KB          10%
        ...
```

Và điều này

```
D:\
        *.mp3               602,456 KB          47%
        *.jpg               305,830 KB          30%
        *.doc               245,355 KB          20%
        ....
```

Chúng ta cần những thông tin này được trình bày trong một biểu đồ để giúp chúng ta hình dung những con số là tốt.

4.6.3.2. Các tính năng

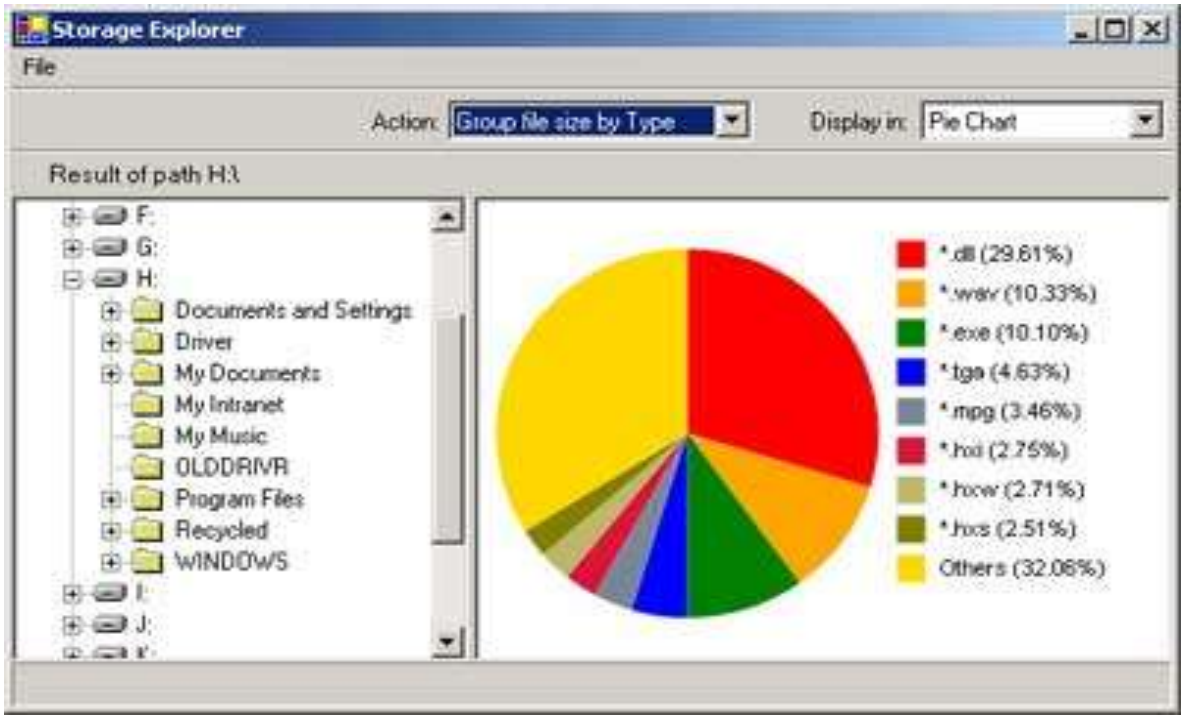
Các tính năng ứng dụng là:

Một TreeView chứa các thông số hay thư mục đại diện cho tập tin thư mục hệ thống. Các thư mục có thể được chọn để xem thông tin về cấu trúc tập tin kích thước bên trong đường dẫn đó.

Cho thấy thông tin về kích thước tập tin nhóm lại theo các thư mục

Cho thấy thông tin về kích thước tập tin được phân nhóm theo loại file

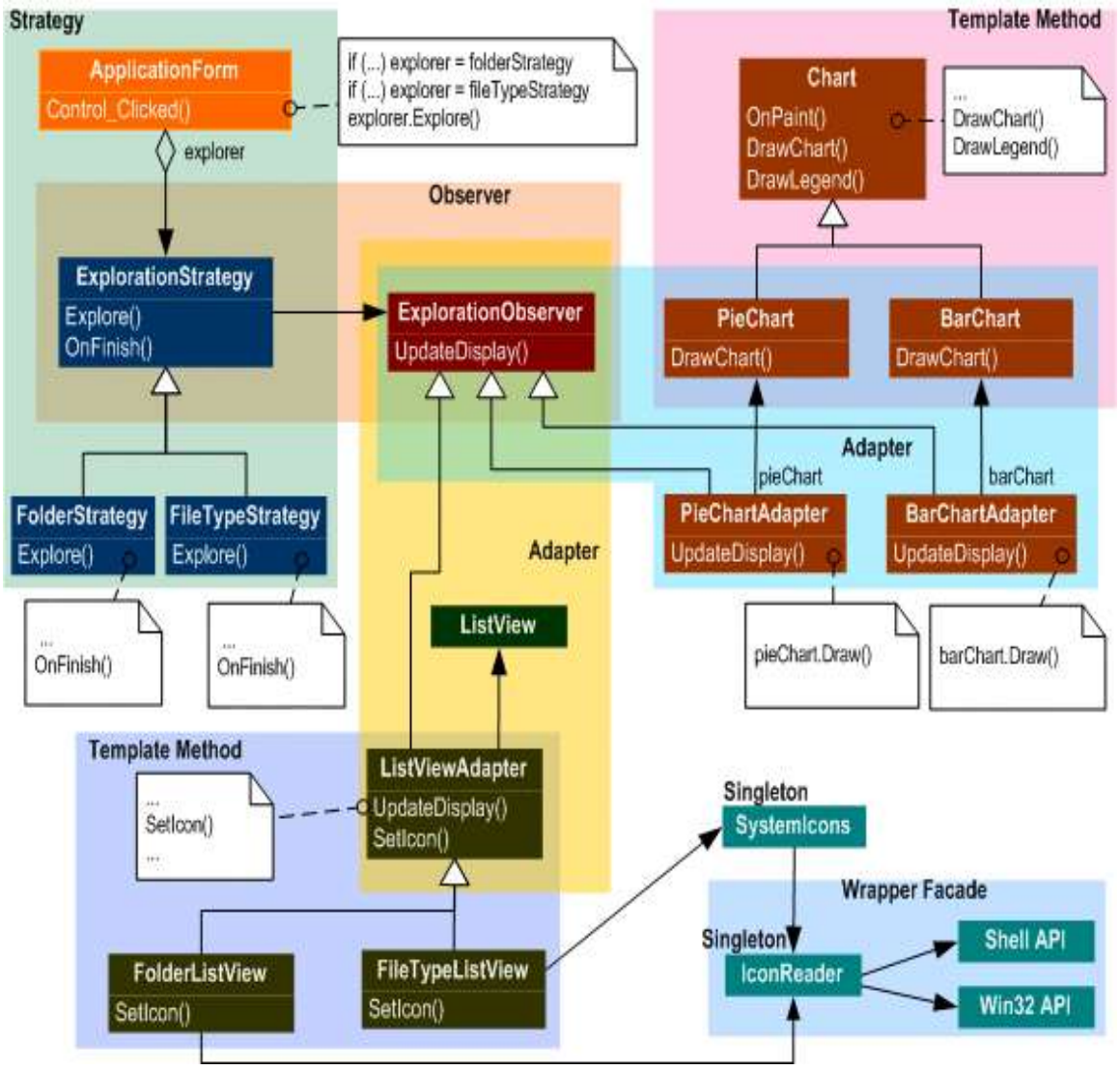
Thông tin được trình bày trong listview biểu đồ pie, hoặc biểu đồ bar trên bảng bên phải.



Hình 4.12. *Giao diện của Storage Explorer*

4.6.3.3 Các mẫu thiết kế bên trong ứng dụng

Như đã nêu ở trên StorageExplorer là một ứng dụng nhỏ được thiết kế và sử dụng một số mẫu thiết kế cùng nhau. Các mẫu này được vận dụng một cách hiệu quả để cho ra sản phẩm là một ứng dụng giúp chúng ta có thể có thể kiểm tra được các thành phần bên trong của một tập tài liệu hay thiết bị nhớ. Và các mẫu đó được thể hiện bằng sơ đồ như sau:

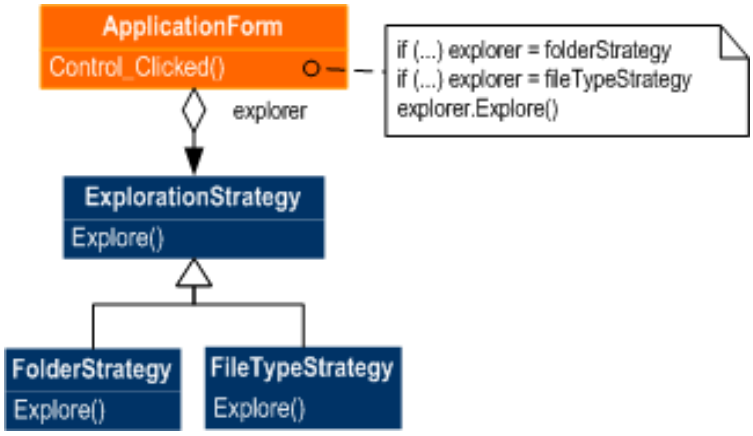


Hình 4.13. Sơ đồ kết hợp các mẫu [8]

a. Strategy

Mục đích: Xác định một họ của các thuật toán, đóng gói mỗi thuật toán đó, và làm cho chúng có thể thay thế cho nhau (GOF).

Các ứng dụng đi sâu tìm hiểu những hệ thống file bằng cách sử dụng hai thuật toán khác nhau, được thực hiện trong hai lớp chiến lược: FolderStrategy và FileTypeStrategy (xem hình dưới) lớp FolderStrategy thực hiện một thuật toán mà đi sâu tìm hiểu các file và kích thước file bên trong nhỏ hơn các thư mục khác nhau trong cùng một đường dẫn. Lớp FileTypeStrategy thực hiện một thuật toán mà đi sâu tìm hiểu các tập tin và kích thước bên trong tập tin cùng loại.



Hình 4.14. *Strategy* [8]

Bây giờ giả sử chúng ta muốn thêm một thuật toán mới. Nó có thể được thực hiện bằng cách tạo ra một phân lớp mới theo lớp ExporationStrategy và đưa đoạn code thuật toán mới theo phương thức Explore (). Trong thời gian chạy và tạo mới các trường hợp của lớp mới và gán cho nó đến đối tượng explorer. Khi chúng ta gọi explore thuật toán của phương thức Explore() được chạy. Đó là ý tưởng của các mô hình chiến lược: đóng gói và thực hiện những thuật toán hoán đổi cho nhau. Việc thực hiện như sau:

```
public abstract class ExplorationStrategy
{
  public virtual void Explore (...){}
}
public class NewAlgorithm : ExplorationStrategy
{
  public override void Explore (...)
  {
    // the new algorithm
  }
}
public class StorageExplorerForm : System.Windows.Forms.Form
{
  // Somewhere in the initialization section
  ExplorationStrategy explorer;
  ExplorationStrategy folderStrategy = new FolderStrategy ();
  ExplorationStrategy fileTypeStrategy = new FileTypeStrategy ();
```

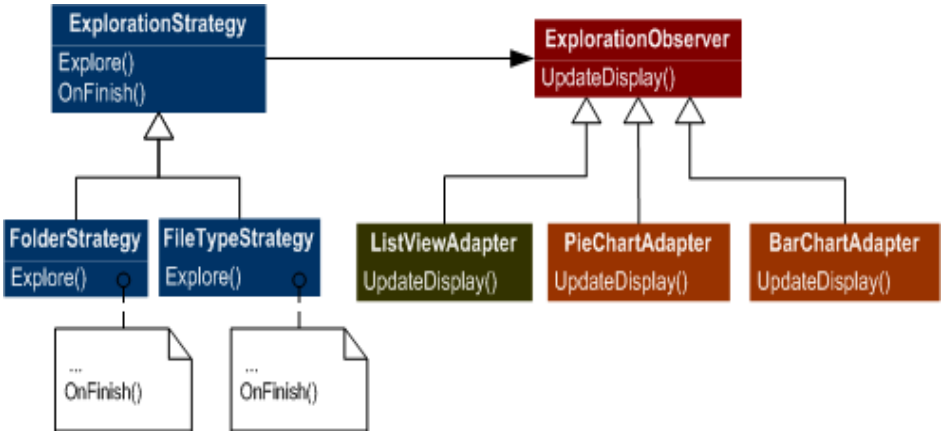
```
 ExplorationStrategy newStrategy = new NewAlgorithm ();


 // Somewhere else, an algorithm is selected
 switch (...)
 {
   case 0: explorer = folderStrategy; break;
   case 1: explorer = fileTypeStrategy; break;
   case 2: explorer = newStrategy; break;
 }
 // Execute the algorithm
 explorer.Explore (...);
}
```

b. Observer

Mục đích: Xác định phụ thuộc một-nhiều giữa các đối tượng để khi một đối tượng thay đổi trạng thái, tất cả các đối tượng phụ thuộc được thông báo và cập nhật tự động (GOF).

Trong ứng dụng mô hình này tạo ra một mối quan hệ giữa các đối tượng (đối tượng ExplorationStrategy) và các nhà quan sát (đối tượng ExplorationObserver), để khi một quá trình thăm dò kết thúc, thông tin này được thông báo cho các đối tượng quan sát, và sau đó là màn hình quan sát kết quả (xem hình dưới) . Các chủ đề cụ thể (FolderStrategy hoặc FileTypeStrategy) thông báo cho các đối tượng quan sát (ListViewAdapter, PieChartAdapter và BarChartAdapter) bằng cách gọi phương thức OnFinish (), do đó tạo ra một sự kiện. Sự kiện này được gửi tới các đối tượng quan sát và sau đó xử lý chúng. mối quan hệ này được quy định tại các lớp trừu tượng sau đó được thừa hưởng nó.



Hình 4.15. *Observer* [8]

Các đối tượng và mối quan hệ quan sát được thiết lập bằng cách đăng ký đối tượng quan sát thông qua phương thức UpdateDisplay () tới sự kiện ExplorationStrategy.Finish, như hình dưới đây:

```
public abstract class ExplorationObserver
{
 public void SubscribeToExplorationEvent (ExplorationStrategy obj)
 {
  obj.Finish += new ExplorationFinishEventHandler (UpdateDisplay);
 }
}
```

Và trong quá trình khởi tạo ứng dụng trong các máy khách (mẫu đơn), đối tượng quan sát cụ thể sẽ gọi phương thức SubscribeToExplorationEvent () và thông qua các ví dụ của strategy cụ thể như là tham số. Kể từ đó, mối quan hệ của đối tượng quan sát- đối tượng được quan sát đã được thiết lập và nó cho thấy sự chấp thuận:

```
// Initialization in the client:
// Create the concrete subject
ExplorationStrategy folderStrategy = new FolderStrategy ();
ExplorationStrategy fileTypeStrategy = new FileTypeStrategy ();
// Create the concrete observer
ExplorationObserver pieChart = new PieChartAdapter ();
// Subscribe the concrete observer object to the concrete strategy object
pieChart.SubscribeToExplorationEvent (folderStrategy);
pieChart.SubscribeToExplorationEvent (fileTypeStrategy);
```

Bây giờ hãy xem mô hình này. Giả sử chúng ta muốn thay đổi các ứng dụng để chứa một yêu cầu mới. Chúng ta muốn lưu kết quả thăm dò vào một tập tin ngoài việc hiển thị nó trên màn hình. Đối với mục đích đó, kế thừa một lớp mới từ lớp ExplorationObserver và đưa đoạn code đó lưu kết quả vào một tập tin bên trong phương thức UpdateDisplay(). Tạo thể hiện của lớp mới rồi gọi SubscribeToExplorationEvent() với đối tượng cụ thể ExplorationStrategy như là tham số. Khi ứng dụng được chạy, thông tin sẽ được gửi đến và hiển thị các tập tin như là tốt. Các mã được hiển thị dưới đây.

```
public class NewConcreteObserver : ExplorationObserver
{
  public override void UpdateDisplay (object o, ExplorationFinishEventArgs e)
  {
    Hashtable result = e.ExplorationResult;
    // Write the result to a file
  }
}
// Somewhere, during the initialization in the client
...
ExplorationObserver fileSaver = new NewConcreteObserver ();
fileSaver.SubscribeToExplorationEvent (folderStrategy);
fileSaver.SubscribeToExplorationEvent (fileTypeStrategy);
```

c. Adapter

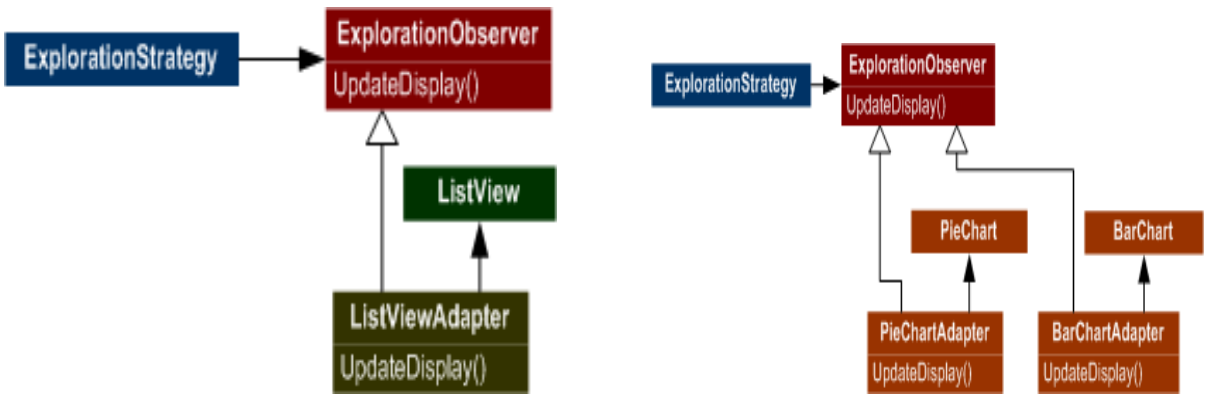
Mục đích: Chuyển đổi giao diện của một lớp vào một giao diện khách mong đợi. Các lớp Adapter cho phép làm việc với nhau rằng dù không có giao diện tương thích (GOF).

Bây giờ kết quả thăm dò sẽ được hiển thị trong một mạng lưới và lớp ListView được sử dụng. Tuy nhiên chúng ta cần phải sửa đổi các ListView để khả năng nó có thể nhận được thông báo tự động khi một thăm dò kết thúc. Cần phải thay đổi lớp ListView phải có xác nhận của lớp ExplorationObserver và do đó có thể ghi đè lên các phương thức UpdateDisplay () được gọi khi các FinishEvent đi kèm. Giải pháp sửa đổi lớp ListView là tạo ra một lớp mới, cụ thể là ListViewAdapter, có khả năng sử

dụng ListView trong khi vẫn có xác nhận của ExplorationObserver. Các lớp ListViewAdapter là một Adapter. Trong điều kiện GOF: các lớp ListViewAdapter thay đổi giao diện của lớp ListView để có giao diện ExplorationObserver mà nó hy vọng.

Mẫu adapter có hai hình thức: Lớp adapter và bộ chuyển đổi đối tượng. Các bộ chuyển đổi lớp được thực hiện bằng cách tạo ra ListViewAdapter thừa hưởng từ ListView và lớp ExplorationObserver. Thực tế này có thể được thể hiện nếu ExplorationObserver là một giao diện thay vì một lớp trừu tượng. ListViewAdapter sau đó có thể thực hiện các giao diện và kế thừa từ các lớp ListView. Nhưng trong trường hợp này ExplorationObserver có một số thực hiện bên trong các lớp mà làm cho một giao diện không phải là một lựa chọn.

Hình thức thứ hai, các bộ chuyển đổi đối tượng, được sử dụng trong các ứng dụng này, sử dụng thành phần đối tượng (xem hình bên dưới).



Hình 4.16. *Adapter* [8]

Các lớp ListViewAdapter kế thừa lớp ExplorationObserverl. Để có một ListView có khả năng, bộ chuyển đổi này tạo ra một đối tượng ListView và sử dụng khi cần thiết. Ưu điểm của phương pháp này là các đối tượng ListView và các thành viên của nó là ẩn từ ListViewAdapter người dùng. Các mã được hiển thị dưới đây:

```
public class ListViewAdapter : ExplorationObserver
{
  protected ListView listView;
  public ListViewAdapter()
  {
    // Create a ListView object and initialze it
    listView = new ListView();
    ...
  }
}
```

Phương pháp tương tự được áp dụng cho PieChartAdapater và lớp BarChartAdapter. Giả sử đã có những lớp biểu đồ và không muốn thay đổi chúng. Giải pháp là một lần nữa để tạo ra một lớp adapter kế thừa từ lớp ExplorationObserver và tạo các đối tượng biểu đồ bên trong bộ chuyển đổi.

Ngoài việc sử dụng các mẫu trên thì Storage explorer còn sử dụng thêm các mẫu thiết kế Template Method, Singleton, Wrapper Façade

KẾT LUẬN

Qua quá trình thực hiện đồ án tốt nghiệp với đề tài "Phương pháp phân tích thiết kế hướng mẫu và ứng dụng" em thấy mình đã đạt được một số kết quả nhất định và đã thu được một số kết quả sau:

1. Nắm bắt được những kiến thức cơ bản về phương pháp phân tích thiết kế hướng đối tượng. Đó cũng chính là bước đệm quan trọng mở đường cho việc tiếp cận và làm quen bước đầu với phương pháp phân tích thiết kế hướng mẫu.

2. Rút ra được một số kinh nghiệm và bài học cơ bản về cách làm việc khoa học, chủ động nghiên cứu vấn đề công nghệ mới. Có thể áp dụng kiến thức đã được học vào thực tiễn, đồng thời thu thập được rất nhiều những kiến thức khác từ quá trình làm đồ án.

3. Vận dụng những kiến thức đã được học vào việc nghiên cứu áp dụng cho những vấn đề mới gặp phải trong quá trình làm việc một cách hiệu quả. Thử nghiệm ngôn ngữ C# để tạo ra ứng dụng minh họa cho đề tài

Với những lợi ích mà phương pháp phân tích thiết kế hướng mẫu mang đến trong việc thiết kế phát triển và bảo trì phần mềm, thì phương pháp này sẽ là phương pháp mang lại hiệu quả công việc cao. Đồng thời phương pháp phân tích hướng mẫu sẽ là phương pháp luận chính được áp dụng trong thời gian tới, và có thể được áp dụng ngày càng rộng rãi trong lĩnh vực phát triển công nghệ phần mềm.

TÀI LIỆU THAM KHẢO

TÀI LIỆU TIẾNG VIỆT

[1] Nguyễn Văn Vỵ- *Phân tích thiết kế các hệ thống thông tin hiện đại, hướng cấu trúc và hướng đối tượng*, NXB Thống kê , Hà Nội, 2002

[2] Nguyễn Thị Thanh Thoan- *Hệ thống thời gian thực và ứng dụng các mẫu thiết kế*, Luận văn thạc sĩ chuyên nghành công nghệ phần mềm, ĐHQG Hà Nội 2007- Mã số 68.40.10

[3]Trần Đan Thư, Huỳnh Thụy Bảo Trân- *Áp dụng mẫu thiết kế hướng đối tượng trong phát triển phần mềm web*, Trường ĐHKHTN- ĐHQG HCM, Tạp chí phát triển khoa học và công nghệ 2007

[4] http://www.ebook.edu.vn/?page=1.39&view=2387

[5] http://cntt.tv/nodes/show/164

[6] http://www.fpt.edu.vn/technology-news/design-pattern-thiet-ke-theo-mo-hinh-mau

TÀI LIỆU TIẾNG ANH

[7]Graig Larman (1998), *Applying UML and Patterns (An Introduction to Object-Oriented Analysis and Design)*, Prentice Hall, New Jersey.

[8] http://www.codeproject.com/KB/architecture/sinagastorageexplorer.aspx

[9]http://www.as3dp.com/2008/08/26/actionscript-proxy-design-pattern-the-virtual-proxy-a-minimal-abstract-example/comment-page-1/#comment-5198

[10] Design Patterns Libraray- http://www.dofactory.com/Patterns/Patterns.aspx