ỨNG DỤNG MẪU THIẾT KẾ TRONG QUÁ TRÌNH PHÁT TRIỂN PHẦN MỀM

APPLICATION OF DESIGN PATTERNS TO SOFTWARE DEVELOPMENT

Trương Đình Huy
Công ty Cổ phần Công nghệ Phần mềm Đà Nẵng - SOFTECH

Võ Trung Hùng
Trường Đại học Bách khoa, Đại học Đà Nẵng

TÓM TẮT

Trong bài báo này, chúng tôi giới thiệu những kết quả nghiên cứu về các Mẫu thiết kế (Design Patterns) được đề xuất bởi Erich Gamma cùng các đồng sự và ứng dụng các Mẫu thiết kế này vào xây dựng phần mềm quản lý điểm sinh viên. Trong qui trình phát triển phần mềm quản lý điểm sinh viên, chúng tôi đã vận dụng các Mẫu thiết kế trong giai đoạn thiết kế. Kết quả ứng dụng thể hiện trong *biểu đồ lớp* (class diagram) của ứng dụng. Khả năng vận dụng của DP trong phát triển phần mềm là rất lớn và trong hệ thống quản lý điểm sinh viên còn có khả năng vận dụng nhiều DP hơn nữa trong quá trình phát triển. Tuy nhiên, chúng tôi chỉ dừng lại ở việc ứng dụng một số DP cơ bản nhất, chưa vận dụng hết tất cả các DP đã được thiết kế, cho những trường hợp điển hình nhất để minh họa.

ABSTRACT

In this paper, we present a study on the Design Patterns (DPs) proposed by Erich Gamma and his colleagues and apply these patterns to the development of a software for the student's grade administration. In the process of software development for grade record, we used these design patterns in all phases. The results could be observed in the class diagrams (class diagram) of the application. The application of the DPs in software development was greatly significant and the student's grade management system which could also be applied with more DPs in the development process. However, we just dealt with the application of some of the most basic DPs and we have not used all the DPs in the application of the most typical cases for illustration.

1. Giới thiệu

Hiện nay, nhiều phần mềm được phân tích và thiết kế theo phương pháp hướng đối tượng (HĐT), sự phát triển của phương pháp này đã tạo nên một bước chuyển mang tính cách mạng trong lĩnh vực công nghệ phần mềm. Bên cạnh sự phát triển đó, khả năng tái sử dụng là một trong những tiêu chí quan trọng để đánh giá chất lượng phần mềm [4].

Erich Gamma cùng các đồng sự đã đề xuất 23 mẫu thiết kế (DP), nhằm đáp ứng được nhiều yêu cầu đối với sản phẩm phần mềm: dễ bảo trì, dễ nâng cấp, chất lượng cao và tiết kiệm thời gian… Trong đó, đáp ứng được tiêu chí quan trọng là khả năng sử dụng lại các đơn thể, thừa kế được các kinh nghiệm của các chuyên gia trong quá trình phát triển phần mềm [1], [5].

DP được phổ biến, giới thiệu rộng rãi tại hầu hết các trường đại học trên thế

giới, và đã được áp dụng trong phát triển các hệ thống phần mềm theo phương pháp HĐT. Tại Việt Nam, trong những năm gần đây DP là chủ đề rất được quan tâm tại một số doanh nghiệp phần mềm và nhiều trường đại học đã đưa DP vào giảng dạy. Tuy nhiên, thực tế áp dụng các DP vào trong phát triển các hệ thống HĐT gặp nhiều khó khăn, đó là: Việc nắm bắt nội dung của từng mẫu DP, cách áp dụng từng mẫu DP trong từng trường hợp, trong từng bài toán cụ thể [1].

Trong bài báo này, chúng tôi tập trung giới thiệu một số nội dung về DP, đặc biệt cách vận dụng DP để phát triển ứng dụng thực tế: Hệ thống quản lý điểm sinh viên của một khoa của trường đại học. Chúng tôi đã vận dụng bằng cách phối hợp và liên kết nhiều DP để phát triển ứng dụng, kết quả vận dụng cụ thể được thể hiện trong *biểu đồ lớp*.

2. Tổng quan

2.1. Các định nghĩa về DP

- DP là bài toán (vấn đề) thông dụng cần giải quyết và cách giải quyết bài toán đó trong một ngữ cảnh cụ thể.
- DP không đơn thuần là một bước nào đó trong các giai đoạn phát triển phần mềm mà nó đóng vai trò là sáng kiến để giải quyết một bài toán thông dụng nào đó. Sự góp mặt của DP sẽ giúp cho việc xác định bài toán cần giải quyết nhanh gọn hơn, từ đó đưa ra cách giải quyết hợp lý.
- Không chỉ được sử dụng để xác định bài toán và cách giải quyết mà DP còn được sử dụng nhằm cô lập các thay đổi trong mã nguồn, từ đó làm cho hệ thống có khả năng tái sử dụng cao. Điều này là tất yếu vì DP tuân thủ nghiêm ngặt các nguyên lý thiết kế HĐT [1].

2.2. Bốn yếu tố xác định một DP

- *Tên mẫu* (pattern *name*): Pattern name giúp ta có thể hình dung một cách nhanh chóng và hiệu quả về mẫu. Nắm được các Pattern name thông dụng sẽ giúp chúng ta nhìn được các bảng thiết kế ở mức trừu tượng hơn. Việc tìm kiếm một tên tốt, phù hợp với trường hợp với ngữ cảnh áp dụng là một trong những phần công việc khó khăn nhất trong việc xây dựng danh mục thiết kế.
- *Bài toán/vấn đề (Problem):* Mô tả tình huống áp dụng DP. Ngoài ra, Problem cũng đề cập các điều kiện trước khi áp dụng mẫu, giúp chúng ta nhìn nhận vấn đề một cách thận trọng, biết khi nào nên, khi nào không nên áp dụng DP.
- *Giải pháp (Solution – cách giải quyết bài toán):* Thể hiện các nhân tố (classes – objects) tham gia giải quyết vấn đề và mối quan hệ, trách nhiệm, các sự cộng tác. Giải pháp được mô tả không phải cho một trường hợp thiết kế hay cài đặt

cụ thể nào mà là áp dụng cho nhiều trường hợp tương tự. Mẫu cung cấp một mô tả trừu tượng của vấn đề (bài toán) được thiết kế và việc sắp xếp các phần tử (các lớp – classes và các đối tượng – objects trong các tình huống) cần thiết để giải quyết bài toán đó.

- *Kết quả (Consequences – các kết quả sẽ đạt được sau khi áp dụng mẫu thiết kế):* Cho chúng ta thấy việc áp dụng các giải pháp (solution) để giải quyết các vấn đề (problem) đặt ra có hiệu quả hay không. Kết quả sẽ đặt ra cho chúng ta các lựa chọn, để từ đó có thể xem xét lựa chọn nào là phù hợp nhất, tốt nhất. Thông thường trong thiết kế HĐT, khi lựa chọn giải pháp để áp dụng các mẫu, chúng ta quan tâm đến: Chi phí và hiệu quả mang lại sau áp dụng mẫu [1], [5].

3. Vận dụng DP phát triển ứng dụng

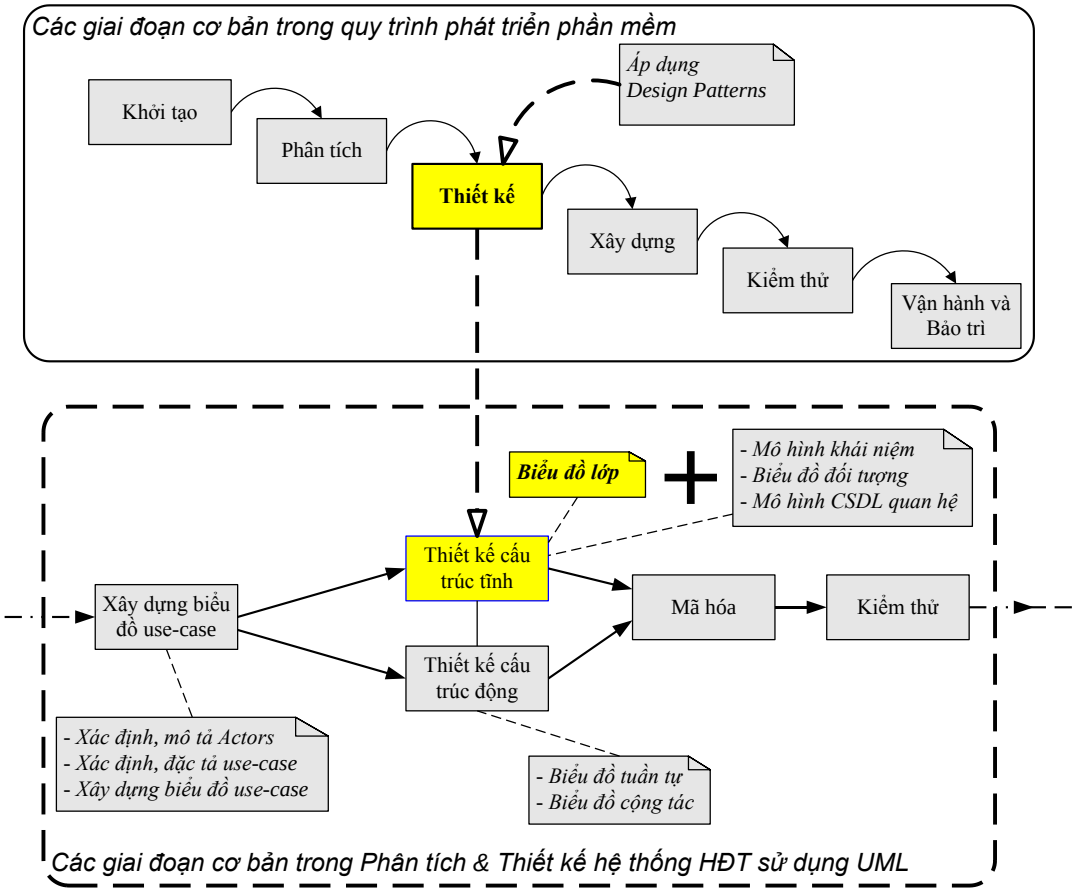
3.1. Mô tả yêu cầu ứng dụng

Khi phát triển Hệ thống hỗ trợ quản lý điểm sinh viên, chúng tôi giới hạn việc quản lý điểm tại một khoa của một trường đại học và phục vụ trực tiếp cho giáo vụ, thông tin quản lý gồm:

- Thông tin về khóa học:
  - Khoa quản lý sinh viên theo các khóa học, ví dụ: KHOA1993, KHOA1994, KHOA1995…
  - Mỗi khóa học bao gồm: Tên khóa học, năm bắt đầu khóa học, năm kết thúc khóa học.
- Thông tin về lớp học:
  - Mỗi khóa học gồm nhiều lớp, ví dụ: 19931998-93T1, 19931998-93T2, 19931998-93TM1, 19931998-93TM2…
  - Mỗi lớp học bao gồm: Mã lớp học, tên lớp học, lớp học thuộc khóa học nào.
- Thông tin về sinh viên:
  - Khoa quản lý thông tin sinh viên theo lớp học.
  - Mỗi sinh viên bao gồm: Mã sinh viên, họ tên, ngày sinh, lớp sinh viên đang học.
- Thông tin môn học, gồm: Mã môn học, tên môn học, số đơn vị học trình.
- Thông tin về điểm:
  - Điểm của mỗi sinh viên được tính theo các môn học
  - Điểm thi mỗi môn học, gồm: Điểm lần 1, điểm lần 2, điểm lần 3, điểm trung bình.

3.2. Mô hình ứng dụng chung

Trong quá trình phát triển hệ thống, chúng tôi sẽ vận dụng các DP trong giai đoạn thiết kế hệ thống, cụ thể áp dụng các DP để thiết kế *biểu đồ lớp* (class diagram) trong giai đoạn *thiết kế cấu trúc tĩnh*, nội dung thể hiện như sau:



Hình 1. *Giai đoạn vận dụng DP trong phát triển ứng dụng*

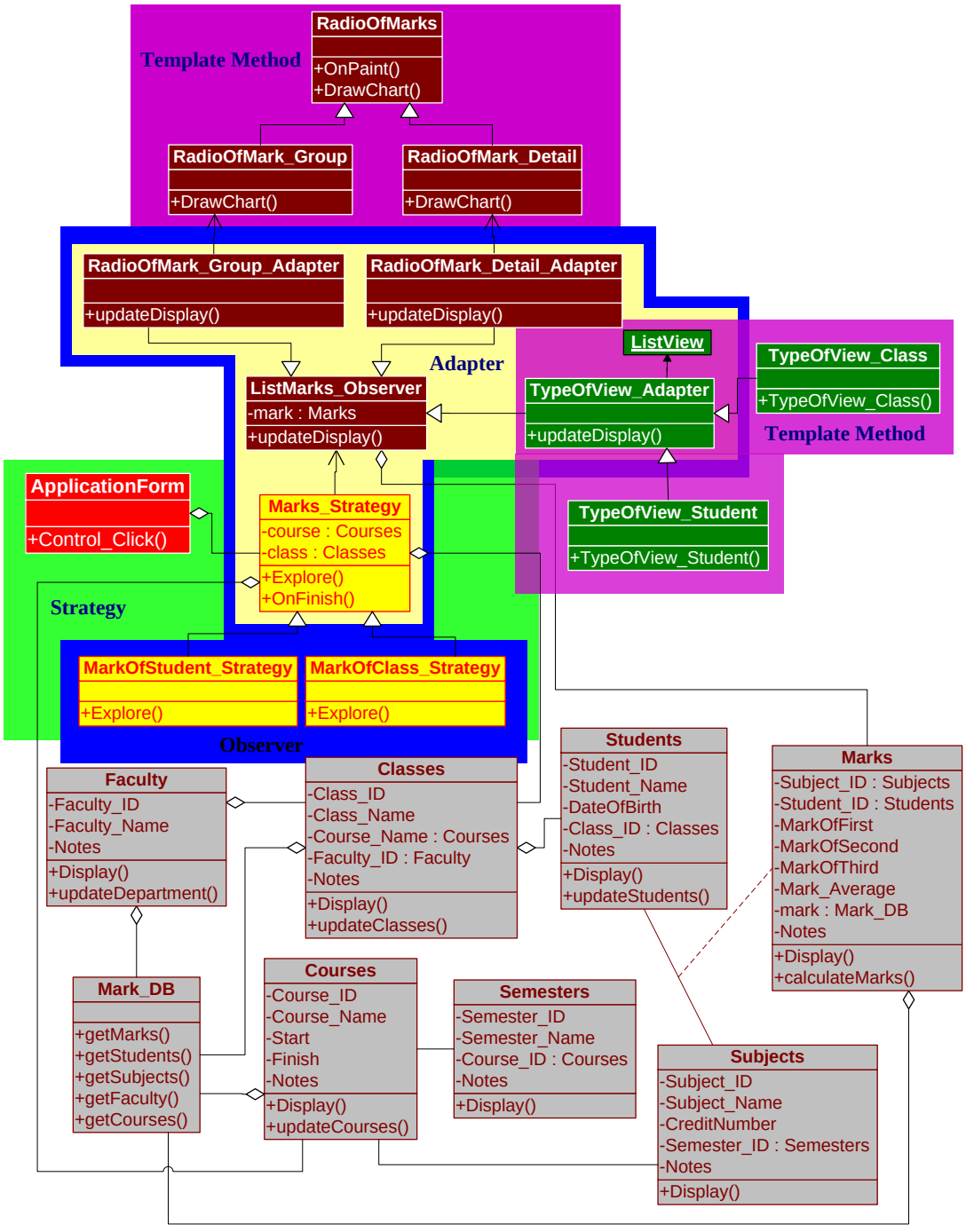
3.3. Các bước để lựa chọn DP

- Việc đầu tiên để lựa chọn mẫu thích hợp là phải xác định được cần thiết kế gì, các vấn đề phát sinh trong khi thiết kế, các vấn đề đó thuộc loại nào và tham khảo xem ứng với loại vấn đề đó có thể dùng mẫu nào để giải quyết.
- Tham khảo lần lượt từng mẫu được chọn, xem mục đích sử dụng, rồi đối chiếu với các vấn đề thiết kế gặp phải (Design Problems). Bước này giúp ta hiểu rõ hơn về các mẫu và tác dụng thực của mẫu lên vấn đề cần giải quyết.
- Tìm hiểu sự tương tác giữa các mẫu thiết kế để xác định được các nhóm mẫu phù hợp với bài toán của mình.
- Các mẫu được phân loại theo mục đích sử dụng: Creational (Kiến tạo), Behavioral (hành vi), Structural (cấu trúc). Sự phối hợp và tính lồng ghép làm cho nhiều mẫu có đặc tính và cấu trúc khá giống nhau, do đó chúng ta cần phải tìm hiểu cẩn thận mục đích của mẫu để phát hiện ra điểm giống nhau và khác nhau nhằm áp dụng một cách hiệu quả nhất, tránh cài đặt sai lầm về ngữ nghĩa khi thiết kế.
- Cố gắng tìm kiếm, khảo sát các nguyên nhân có thể dẫn tới việc thiết kế lại (redesigning) để cô lập hóa, nhằm tránh các thay đổi không cần thiết.

- Xác định các thành phần (module, lớp đối tượng…) có thể thay đổi trong tương lai. Từ đó quyết định xem phải làm gì để có thể thay đổi hệ thống mà không cần phải thiết kế lại [1],[4].

3.4. Các DP được áp dụng trong ứng dụng

Từ các hướng dẫn cách chọn DP để áp dụng, chúng tôi đã thiết biểu đồ lớp (class diagram) của ứng dụng như sau:



Hình 2. *Các DP được áp dụng trong biểu đồ lớp của ứng dụng*

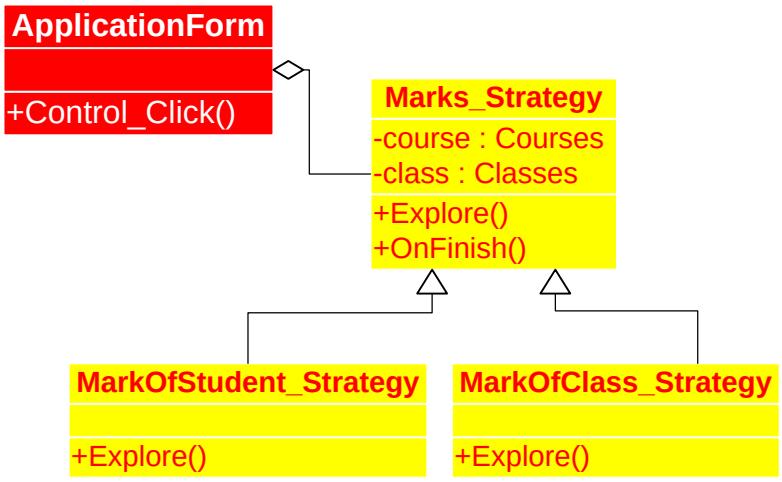
3.5. Mô tả cách vận dụng các DP trong ứng dụng

Trong quá trình phát triển "Hệ thống hỗ trợ quản lý điểm sinh viên" chúng tôi đã vận dụng các DP: *Strategy Pattern, Adapter Pattern, Observer Pattern, Template Pattern*...nội dung của từng DP trong *biểu đồ lớp* của ứng dụng nằm trong từng vùng tô màu khác nhau được thể hiện trong **Hình 2.**

Sau đây chúng tôi mô tả ý nghĩa, nội dung vận dụng và ưu điểm của từng DP trong quá trình vận dụng thiết kế hệ thống.

3.5.1. Mẫu Strategy (Strategy Pattern)

- Ý nghĩa sử dụng: Định nghĩa một họ các thuật toán, đóng gói từng thuật toán và làm cho chúng có khả năng hoán đổi cho nhau [5].
- Nội dung vận dụng:



Hình 3. *Nội dung Strategy Pattern trong ứng dụng*

Ứng dụng hiển thị danh sách điểm sinh viên theo 2 cách (thuật toán - algorithms) khác nhau và được cài đặt bởi 2 lớp khác nhau: MarkOfStudent_Strategy và MarkOfClass_Strategy như trong Hình 3.

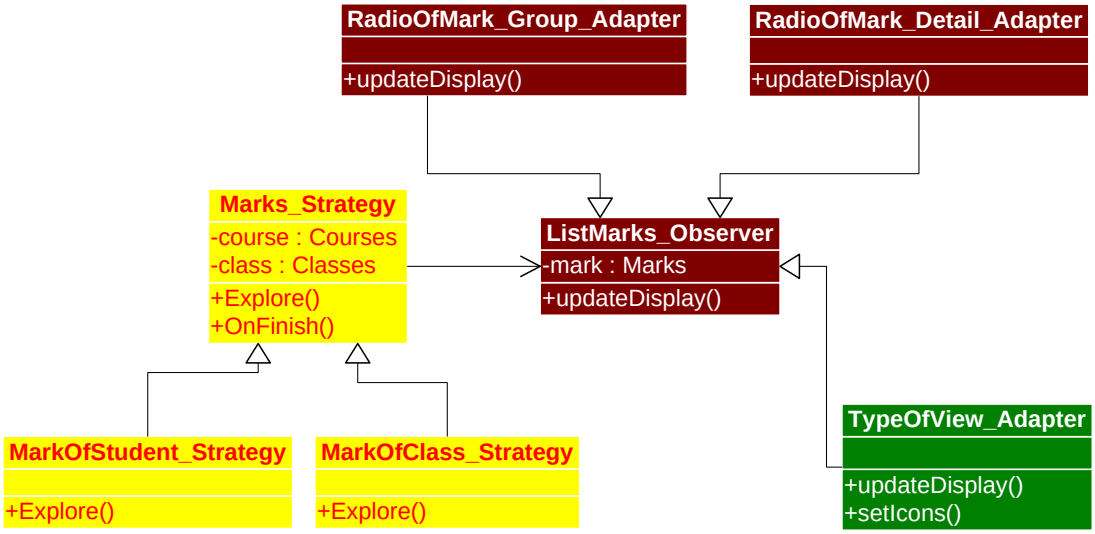
- Lớp MarkOfClass_Strategy thừa kế từ lớp trừu tượng (abstract class) Marks_Strategy, thuật toán hiển thị bảng điểm từng môn học theo danh sách lớp học trong từng khóa học được cài đặt trong lớp MarkOfClass_Strategy.
- Lớp MarkOfStudent_Strategy thừa kế từ lớp trừu tượng (abstract class) Marks_Strategy, thuật toán hiển thị bảng điểm từng môn học theo danh sách sinh viên từng lớp học hoặc theo danh sách sinh viên từng khóa học được cài đặt trong lớp MarkOfStudent_Strategy.

Một trong những thuận lợi khi vận dụng Strategy Pattern là dễ dàng thêm mới một thuật toán. Ví dụ ứng dụng cần thêm mới một thuật toán để hiển thị điểm của sinh viên theo một cách khác ngoài hai thuật toán đã cài đặt, chúng ta có thể thực hiện như sau:

- Tạo một lớp con mới có tên NewAlgorithm thừa kế từ lớp Marks_Strategy và cài đặt phương thức Explore() bằng mã nguồn của thuật toán mới.
- Trong lúc thực thi (run-time) ứng dụng tạo ra một thể hiện (instance) của lớp NewAlgorithm và gán nó một cách động đến đối tượng explore. Khi chúng ta gọi explore.Explore() phương thức của các thuật toán được thực thi. Đó là ý nghĩa sử dụng của Strategy Pattern: Đóng gói và cho phép các thuật toán có khả năng hoán đổi cho nhau.

3.5.2. Mẫu Observer (Observer Pattern)

- Ý nghĩa sử dụng: Định nghĩa mối quan hệ phụ thuộc một – nhiều giữa các đối tượng sao cho khi một đối tượng thay đổi trạng thái thì tất cả các đối tượng phụ thuộc vào nó được thông báo và cập nhật một cách tự động [5].
- Nội dung vận dụng:



Hình 4. *Nội dung Observer Pattern trong ứng dụng*

Để thể hiện đúng ý nghĩa của Observer Pattern, trong trường hợp này cần thực hiện:

- Bước 1: ListMarks_Observer tạo mối quan hệ (đăng ký các chủ đề - Subjects) với các đối tượng của lớp cụ thể Marks_Strategy với (cho) các đối tượng của lớp cụ thể của của nó. Mục đích của việc đăng ký các chủ đề này để mỗi khi tiến trình của MarkOfStudent_Strategy hoặc MarkOfClass_Strategy kết thúc sẽ có thông báo đến cho ListMarks_Observer và có các kết quả hiển thị tương ứng, nội dung thể hiện trong Hình 4.
- Bước 2: Để thực hiện được yêu cầu của Bước 1, thì các mối quan hệ này được định nghĩa ở lớp trừu tượng và các lớp cụ thể của nó thừa kế lại.

Các chủ đề (các sự kiện - events) của lớp cụ thể MarkOfStudent_Strategy và MarkOfClass_Strategy thông báo đến các lớp cụ thể RadioOfMark_Group_Adapter,

RadioOfMark_Detail_Adapter và TypeOfView_Adapter bằng các gọi phương thức OnFinish(), phương thức này lần lượt sinh ra các sự kiện gởi đến các lớp cụ thể của ListMarks_Observer các sự kiện này sẽ được xử lý để hiển thị nội dung tương ứng.

Một trong những thuận lợi khi áp dụng Observer Pattern là dễ dàng thay đổi (bổ sung) yêu cầu mới, ví dụ chúng ta bổ sung yêu cầu mới: Nội dung hiển thị vừa được hiển thị trên màn hình (yêu cầu cũ), vừa được lưu trữ trong tệp tin (yêu cầu mới). Chúng ta thực hiện như sau:

- Tạo một lớp mới NewConcreteObserver thừa kế từ lớp ListMarks_Observer, sau đó cài đặt mã nguồn lưu thông tin vào tệp tin (file) trong phương thức UpdateDislay().
- Tạo thể hiện (instance) của lớp NewConcreteObserver vừa tạo và gọi đăng ký sự kiện SubcribeToExplorationEvent() nhận các đối tượng của lớp cụ thể của Marks_Strategy như là các tham số.

4. Kết luận

Trong quá trình phát triển Hệ thống hỗ trợ quản lý điểm sinh viên, chúng tôi đã ánh xạ những cơ sở lý thuyết về nội dung các DP vào phát triển hệ thống hỗ trợ quản lý điểm sinh viên. Với kết quả này, một phần nào đã khái quát được cơ bản những yêu cầu áp dụng DP trong thực tế.

Hiện tại có nhiều tài liệu đề cập đến DP, nhiều ví dụ minh họa về cách vận dụng DP để thiết kế... Tuy nhiên, hầu hết đó là các giải pháp từng phần, cách vận dụng các DP chỉ mang tính chất đơn lẻ, chỉ dành riêng cho từng DP cụ thể... Hệ thống quản lý điểm sinh viên mà chúng tôi phát triển là một ứng dụng thực tế, vận dụng phối hợp và liên hết đồng thời nhiều DP (*Strategy Pattern, Adapter Pattern, Observer Pattern, Template Pattern*...) trong quá trình thiết kế, kết quả vận dụng thể hiện trong *biểu đồ lớp*. Qua ứng dụng này, nêu lên được khả năng vận dụng của DP để phát triển ứng dụng là rất lớn, chúng ta có thể phối hợp sử dụng đồng thời nhiều DP để phát triển một ứng dụng lớn. Việc vận dụng tốt các DP sẽ mang lại nhiều lợi ích: dễ bảo trì, dễ nâng cấp, chất lượng cao và tiết kiệm thời gian. Đặc biệt đây là giải pháp thừa kế được nhiều kinh nghiệm của các chuyên gia trong phát triển phần mềm một cách hiệu quả nhất.

DP là vấn đề được nhiều trường đại học, cũng như nhiều doanh nghiệp phần mềm quan tâm để nghiên cứu, vận dụng. Nghiên cứu Design Patterns giúp ta hiểu hệ thống một cách nhanh chóng thông qua các mô hình trừu tượng. Áp dụng Design Patterns trong phát triển hệ thống, giúp chúng ta giải quyết các vấn đề được nhanh chóng và giảm nguy cơ phải thiết kế lại. Hơn nữa, Design Patterns là cặp bài toán/giải pháp (problem/solution), do đó, nghiên cứu Design Pattern cũng giúp ta cách phát hiện các lỗi thiết kế (Design Problem) từ đó tìm được giải pháp hợp lý [1].

TÀI LIỆU THAM KHẢO

[1] Nguyễn Quý Minh, Tăng Nguyễn Trung Hiếu, Phạm Anh Vũ, Lê Hải Dương, Phương Lan, *Design Patterns*. Nhà xuất bản Phương Đông 2005

[2] Grady Booch, *Object-oriented Analysis and Design with Applications*, 3rd edition, Addison-Wesley 2007.

[3] Võ Trung Hùng, *Software Engineering*, VOCW, Vietnam Education Foundation, 2008

[4] Nguyễn Văn Phi, *Nghiên cứu và ứng dụng các mẫu thiết kế (Design Patterns) trong phát triển hệ thống hướng đối tượng*, Báo cáo tốt nghiệp Thạc sĩ ngành CNTT, Đại học Đà Nẵng Khóa 2004 – 2007.

[5] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional 1995.

[6] Fer Kroll, Philippe Kruchten, The Rational Unified Process Made Easy – A practitionner's guide to the RUP, Addison- Wesley 2004

[7] Craig Larman, Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design and Iterative Development, Pearson Education 2005

[8] Alan Shalloway, James R.Trott, Design Patterns Explanined – A New Perspective on Object-Oriented Design Second Edition, Addison Wesley Professional – 2004.

[9] Robert C.Martin, *Design Principles and Design Patterns*, Object Mentor – 2000.