**CHƯƠNG I: TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG KHÍA CẠNH**

1. **Tổng quan về lập trình hướng khía cạnh**
2. **Tổng quan**

Hệ thống phần mềm có thể được xem là một thể hiện kết hợp nhiều vấn đề. Một hệ thống tiêu biểu có thể gồm nhiều dạng vấn đề như xử lý nghiệp vụ, hiệu suất, bảo toàn dữ liệu, bảo mật, bẫy lỗi, … và còn có những vấn đề của quá trình phát triển hệ thống như tính dễ quản lý, dễ bảo trì và phát triển. Trong việc thiết kế phần mềm, các nhà thiết kế phần mềm thường tập chung sự quan tâm vào các chức năng chính, cơ bản của hệ thống. Trong ứng dụng của các doanh nghiệp, đó chính là các logic nhiệp vụ cơ bản. Các mô-đun xử lý nghiệp vụ chính có thể được thực hiện riêng rẽ nhưng có những mối quan tâm liên quan đến nhiều mô-đun như tính năng lưu vết, xác thực quyền hạn, truy cập cơ sở dữ liệu, bẫy lỗi, … Các tính năng đó là cần thiết đối với mỗi mô-đun chính của hệ thống và dàn trải trên nhiều môđun của hệ thống. Các mối quan tâm dàn trải trên nhiều mô-đun của hệ thống đó được gọi là các mối quan tâm cắt ngang (cross-cutting concerns). Do vậy đối với mỗi mô-đun, lập trình viên ngoài việc quan tâm đến những vấn đề cơ bản còn phải tính đến những mối quan tâm cắt ngang và sự trộn lẫn nhiều đoạn mã để xử lý những yêu cầu khác nhau. Sự trộn lẫn nhiều đoạn mã như vậy gọi là vấn đề đan nhau. Những vấn đề này ảnh hưởng đến vấn đề thiết kế và phát triển phần mềm như hệ thống khó xây dựng, khó quản lý, phát triển và tái sử dụng kém.

Lập trình hướng đối tượng (Object-Oriented Programming - OOP) là phương pháp lập trình phổ biến nhất hiện nay được sử dụng để quản lý các mối quan tâm nghiệp vụ chính, tuy nhiên phương pháp này chưa đủ mạnh để xử lý hiệu quả rất nhiều các mối quan tâm cắt ngang hệ thống, đặc biệt là các ứng dụng phức tạp. Phương pháp lập trình hướng khía cạnh (Aspect-Oriented Programming - AOP) là phương pháp lập trình mới cho phép chúng ta thực hiện các vấn đề riêng biệt một cách linh hoạt và kết hợp chúng lại để tạo nên hệ thống sau cùng. Với AOP, ta có thể cài đặt các mối quan tâm cắt ngang hệ thống trong các aspect thay vì dàn trải chúng trên các mô-đun nghiệp vụ chính liên quan. Quá trình bộ đan aspect (weaver) kết hợp các mô-đun nghiệp vụ chính với các aspect thành hệ thống cuối cùng được gọi là quá trình đan (weaving).

1. **Giải quyết các mối quan tâm cắt ngang bằng OOP**

Trong thực tế, phương pháp OOP thường được sử dụng để mô-đun hóa các mối quan tâm của một hệ thống phần mềm. Tuy nhiên, thực tế thì, mặc dù OOP giải quyết rất tốt trong việc mô-đun hóa các mối quan tâm nghiệp vụ chính nhưng lại gặp khó khăn trong việc mô-đun hóa các mối quan tâm cắt ngang hệ thống.

Trong OOP, các mô-đun chính có thể được kết nối lỏng với nhau thông qua giao diện, nhưng không dễ gì làm việc này với các mối quan tâm cắt ngang hệ thống. Lý do chính là một mối quan tâm được thực thi thành hai phần: Phần phía server (phần phục vụ) và phần phía client (phần được phục vụ). OOP mô-đun hóa phần client khá tốt bằng các lớp và các giao diện. Tuy nhiên, ở phía server, một mối quan tâm lại là một loại cắt ngang hệ thống, bao gồm các yêu cầu gửi đến server, dàn trải trên các client.

**3. Các vấn đề gặp phải khi thực thi các mối quan tâm cắt ngang bằng OOP**

Khi thực thi các mối quan tâm cắt ngang bằng phương pháp OOP, hệ thống sẽ gặp phải một số vấn đề trong đó có hai vấn đề chính là sự chồng chéo mã nguồn và dàn trải mã nguồn.

• Chồng chéo mã nguồn (Code tangling): Các mô-đun trong hệ thống có thể đồng thời phải thực hiện nhiều yêu cầu khác nhau.

• Dàn trải mã nguồn (Code scattering): Mã nguồn của các yêu cầu mức hệ thống (an ninh, lưu vết, lưu trữ,...) được cài đặt ở trong rất nhiều mô-đun.

Với các vấn đề đã phân tích ở trên, quá trình phát triển phần mềm sẽ bị ảnh hưởng với một số vấn đề sau:

• Ánh xạ giữa các mối quan tâm và cài đặt kém.

• Hiệu suất làm việc thấp.

• Khả năng sử dụng lại mã nguồn kém.

• Chất lượng mã nguồn thấp.

• Khó bảo trì hệ thống.

**4. Lập trình hướng khía cạnh**

**4.1. Lịch sử hình thành**

Vào năm 2001, Ông Gregor Kiczales và các đồng nghiệp tại Xerox PARC phát triển khái niệm rõ ràng về lập trình hướng khía cạnh AOP. Kế tiếp vào sự nghiên cứu đó, nhóm nghiên cứu của IBM theo đuổi trên một cách tiếp cận ngôn ngữ thiết kế và trong đã đề xuất để sử dụng sử dụng rộng rãi về mô hình lập trình hướng khía cạnh này.

**4.2. Cú pháp của AOP và mô hình lập trình**

Cắt ngang là một kỹ thuật AOP cho phép nhận dạng các mối quan tâm và cấu trúc chúng vào trong các mô-đun bằng cách chúng có thể được triệu gọi tại các điểm khác nhau thông qua ứng dụng. Có hai loại cắt ngang, tĩnh và động. Cắt ngang động sửa đổi hành vi thực thi của một đối tượng bằng việc đan kết hành vi mới tại các điểm quan tâm cụ thể. Cắt ngang tĩnh thay thế cấu trúc tĩnh của một thành phần bằng việc xen thêm các phương thức và các thuộc tính tại thời điểm thực thi. Các cấu trúc ngôn ngữ cơ bản và cú pháp được sử dụng để khai báo cắt ngang trong AOP bao gồm: joint point, pointcut, advice, introduction, aspect. Tùy thuộc vào các công cụ AOP khác nhau mà có các quy định khác nhau về cú pháp.

**4.3. Quản lý các mối quan tâm bằng lập trình hướng khía cạnh**

Mỗi mối quan tâm cắt ngang được cài đặt một cách riêng rẽ trong một thành phần là aspect. Aspect định nghĩa các điểm thực thi trong các thành phần phía client đòi hỏi sự thực hiện mối quan tâm cắt ngang. Đối với mỗi điểm thực thi, aspect định nghĩa các hành vi cần thiết để thực hiện hành vi của aspect, ví dụ như gọi một API lưu vết. Các mô-đun phía client không chứa bất kỳ mã để triệu gọi sự thực thi aspect nào. Điều này dẫn đến các thành phần phía client không bị xen lẫn bởi các lời gọi để thực thi một hoặc nhiều mối quan tâm.

**4.4. Đan kết**

Một bộ đan kết hợp mã ứng dụng với các aspect để tạo ra hệ thống cuối cùng.

**4.5. Phương pháp lập trình hướng khía cạnh**

Về cơ bản quá trình phát triển phần mềm theo AOP gồm có 3 bước sau đây:

• Phân tích các yêu cầu để xác định vấn đề chung và vấn đề đan nhau: sử dụng phương pháp phân tích và thiết kế hướng khía cạnh (Aspect-Oriented Analyis and Design) để phân tích các yêu cầu của bài toán nhằm tách các mối quan tâm nghiệp vụ chính và mối quan tâm cắt ngang hệ thống. Sau bước này ta thu được bản phân tích mà các mối quan tâm nghiệp vụ chính và các mối quan tâm cắt ngang được tách riêng.

• Xây dựng thể hiện từng vấn đề riêng biệt: sau khi các mối quan tâm cắt ngang được tách ra khỏi các mối quan tâm nghiệp vụ chính thì ta có thể dùng phương pháp lập trình truyền thống như lập trình hướng thủ tục hoặc lập trình hướng đối tượng để cài đặt các mối quan tâm nghiệp vụ chính, còn các mối quan tâm cắt ngang thì được cài đặt trong một aspect theo quy tắc mà công cụ AOP được sử dụng quy định. Như vậy, mối quan tâm nghiệp vụ chính được xây dựng độc lập với các mối quan tâm cắt ngang.

• Tổng hợp các thể hiện: sau khi các mối quan tâm nghiệp vụ chính và các mối quan tâm cắt ngang được cài đặt một các riêng rẽ, để hệ thống có thể hoạt động được thì các mối quan tâm cần phải được kết hợp theo một số quy tắc nào đó. Quá trình kết hợp này gọi là quá trình tích hợp hay quá trình đan được thực hiện bằng cách sử dụng các quy tắc đan. Sau quá trình đan ta nhận được hệ thống cuối cùng mà các mối quan tâm nghiệp vụ chính và mối quan tâm cắt ngang đã được đan kết với nhau để thực hiện mục tiêu theo yêu cầu.

**4.6. Lợi ích của lập trình hướng khía cạnh Một số lợi ích chính của AOP:**

• Có thể nhận dạng và miêu tả một cách rõ ràng các mối quan tâm cắt ngang giúp các kiến trúc sư có thể xem xét các hành vi cắt ngang dưới dạng các aspect tại giai đoạn sớm của vòng đời dự án.

• Phân chia trách nhiệm rõ ràng cho các mô-đun riêng lẻ: AOP cho phép tách các mối quan tâm cắt ngang ra khỏi các mối quan tâm nghiệp vụ chính.

• Dễ dàng phát triển hệ thống: AOP mô-đun hóa các aspect riêng lẻ do đó khi hệ thống cần chức năng mới, nhà phát triển chỉ cần tạo ra một aspect thực hiện các yêu cầu mới đó mà không cần thay đổi các mô-đun nghiệp vụ chính.

• Tái sử dụng mã nguồn: AOP cho phép các nhà phát triển dễ dàng sử dụng lại mã nguồn của một aspect trong nhiều thành phần và do đó giảm các nỗ lực thể hiện mã chương trình.

• Khả năng mô-đun hóa và đóng gói: AOP làm tăng tính mô-đun hóa và đóng gói như mã thành phần tốt hơn ngắn gọn và gọn gàng.

• Cho phép để lại quyết định thiết kế tương lai: người thiết kế hệ thống có thể để lại các quyết định thiết kế cho những yêu cầu tương lai nhờ thực hiện theo các aspect riêng biệt.

• Rút ngắn thời gian bàn giao sản phẩm: việc tái sử dụng mã nguồn tốt hơn giúp giảm thời gian phát triển hệ thống.

• Giảm thiểu chi phí thực thi.

**4.7. Nhược điểm của phương pháp lập trình hướng khía cạnh**

Phương pháp lập trình hướng khía cạnh có nhiều ưu điểm kể trên, tuy nhiên nó cũng có một số hạn chế sau:

• Các aspect phá vỡ tính đóng gói của các đối tượng.

• Giảm hiệu suất thực thi trong trường hợp thực hiện đan kết lúc thực thi hoặc tốn thời gian nạp chương trình trong trường hợp thực hiện đan kết lúc nạp chương trình.

• Khi chúng ta mắc sai lầm trong việc diễn đạt thực thi cắt ngang có thể gây lỗi tràn ngập toàn chương trình.

• Việc thay đổi các joint point của một chương trình mà không được lường trước bởi người viết aspect sẽ gây ra các lỗi.

• Vấn đề bảo mật có thể bị phá vỡ bởi việc sử dụng AOP để cài thêm các đoạn mã tại các vị trí phù hợp.