BM Hệ thống thông tin

Nguyên lý Kiến trúc Phần mềm

Giới thiệu về Thiết kế Phần mềm

- Định nghĩa Thiết kế Phần mềm:
  - Là quá trình xác định kiến trúc, các thành phần và các tương tác giữa chúng để tạo ra một sản phẩm phần mềm có hiệu suất cao, dễ bảo trì và mở rộng.
  - Là quá trình xác định kiến trúc, giao diện, module, các thuật toán, cấu trúc dữ liệu, hệ thống cơ sở dữ liệu và các chi tiết khác của phần mềm để đáp ứng các yêu cầu chức năng, hiệu năng, bảo mật, khả năng mở rộng và duy trì của phần mềm.

Giới thiệu về Thiết kế Phần mềm

Tầm quan trọng của Thiết kế Phần mềm trong quá trình phát triển phần mềm: Thiết kế phần mềm đóng vai trò quan trọng trong quá trình phát triển phần mềm bởi vì nó ảnh hưởng đến chất lượng của sản phẩm, tính bảo trì và sửa chữa của sản phẩm và độ linh hoạt để mở rộng phần mềm trong tương lai.

Nguyên lý Thiết kế Phần mềm

- Định nghĩa Nguyên lý Thiết kế phần mềm: Là các quy tắc, hướng dẫn và tiêu chuẩn được áp dụng trong quá trình thiết kế phần mềm để đảm bảo tính linh hoạt, khả năng mở rộng, tái sử dụng và hiệu suất của phần mềm.

Nguyên lý SOLID : Được đề xuất bởi Robert C. Martin để xác định các nguyên tắc cơ bản cho thiết kế phần mềm, bao gồm:

- Interface Segregation Principle (ISP)
- Liskov Substitution Principle (LSP)
- Open-Closed Principle (OCP)
- Single Responsibility Principle (SRP)
- Dependency Inversion Principle (DIP)

Tính tách biệt (Separation of Concerns)

- Tính tách biệt (Separation of Concerns) là một trong những nguyên tắc thiết kế phần mềm quan trọng để đảm bảo tính linh hoạt, tái sử dụng và bảo trì của phần mềm.

Tách biệt giữa dữ liệu và logic

- Tách biệt giữa dữ liệu và logic: Thông qua việc sử dụng các thuộc tính và phương thức truy cập để tách riêng phần dữ liệu và phần logic xử lý trên dữ liệu.

Tách biệt giữa lớp và đối tượng

- Tách biệt giữa lớp và đối tượng: Sử dụng các lớp để tách biệt logic của phần mềm thành các phần nhỏ hơn, đơn giản hơn và dễ bảo trì hơn.

Tách biệt trong thiết kế kiến trúc (Architectural Design)

- Tách biệt giữa lớp kiến trúc: Chia kiến trúc phần mềm thành các lớp tách biệt nhau như: giao diện người dùng, xử lý dữ liệu, kết nối cơ sở dữ liệu, các module phần mềm khác.
- Tách biệt giữa kiến trúc và chức năng: Thiết kế kiến trúc của phần mềm không nên phụ thuộc vào các chức năng cụ thể, mà nên tập trung vào các đặc tính cấu trúc như độ ổn định, độ bảo mật, tính mở rộng, tính di động, ...

Tách biệt trong thiết kế hướng sự kiện (Event-Driven Design)

- Tách biệt giữa trình điều khiển (Controller) và giao diện người dùng: Tránh cho trình điều khiển trực tiếp thao tác với giao diện người dùng. Thay vào đó, sử dụng các sự kiện (Event) để tách biệt hoạt động của trình điều khiển và giao diện người dùng.
- Tách biệt giữa sự kiện và xử lý sự kiện: Tách riêng phần sử lý sự kiện khỏi quá trình phát sinh sự kiện để tăng tính tái sử dụng và khả năng mở rộng của phần mềm.

Tách biệt trong thiết kế lớp (CLASS)

- Tách biệt giữa các lớp: Phân chia phần mềm thành các lớp chức năng tách biệt để giảm sự phụ thuộc giữa các thành phần của phần mềm.

- Tính linh hoạt: Tách biệt giúp phần mềm dễ dàng thay đổi, bổ sung, cập nhật các chức năng mà không làm ảnh hưởng đến các phần khác trong phần mềm.
- Tính tái sử dụng: Tách biệt giúp phần mềm dễ dàng sử dụng lại các phần đã được thiết kế và phát triển trước đó.
- Dễ bảo trì: Tách biệt giúp dễ dàng bảo trì và sửa lỗi phần mềm vì các thành phần của phần mềm được phân tách rõ ràng.

Lợi ích của tính tách biệt trong thiết kế phần mềm

Tính kết dính và phụ thuộc chồng chéo trong thiết kế Class UML

Tính kết dính và phụ thuộc chồng chéo là các khái niệm quan trọng trong thiết kế Class UML.

Tính kết dính liên quan đến mức độ liên kết giữa các phương thức và thuộc tính của một Class.

Phụ thuộc chồng chéo liên quan đến mức độ phụ thuộc giữa các Class trong một hệ thống phần mềm.

Tính kết dính trong thiết kế Class UML

Tính kết dính cao là một ưu điểm của thiết kế Class UML, bởi vì nó giúp tăng tính bảo mật và độ tin cậy của phần mềm.

Tính kết dính thấp có thể dẫn đến vấn đề về quản lý mã nguồn và sửa lỗi, và dễ dàng gây ra các lỗi khác nhau khi phát triển phần mềm.

Các cấp độ kết dính trong thiết kế Class UML

- Các cấp độ kết dính bao gồm kết dính cứng, kết dính chặt chẽ, kết dính lỏng lẻo và kết dính yếu.
- Kết dính cứng là một trường hợp đặc biệt của kết dính chặt chẽ, trong đó một Class phụ thuộc trực tiếp vào một Class khác.
- Kết dính lỏng lẻo và kết dính yếu là các cấp độ kết dính thấp hơn, trong đó các Class có sự phụ thuộc tương đối lớn vào nhau.

Phụ thuộc chồng chéo trong thiết kế Class UML

- Phụ thuộc chồng chéo là sự phụ thuộc giữa các Class trong một hệ thống phần mềm, khi một Class phụ thuộc vào một Class khác và ngược lại.
- Phụ thuộc chồng chéo có thể dẫn đến vấn đề về quản lý mã nguồn và sửa lỗi, và dễ dàng gây ra các lỗi khác nhau khi phát triển phần mềm.

Cách giảm phụ thuộc chồng chéo trong thiết kế Class UML

Sử dụng các nguyên lý thiết kế phần mềm như Đóng gói, Đa hình và Kế thừa để giảm phụ thuộc chồng chéo giữa các Class.

Sử dụng các mẫu thiết kế phần mềm như Dependency Injection (DI), Inversion of Control (IoC) và Observer để giảm phụ thuộc chồng chéo giữa các Class.

Nguyên tắc

Tính kết dính và phụ thuộc chồng chéo là các khái niệm quan trọng trong thiết kế Class UML.

Tính kết dính cao và phụ thuộc chồng chéo thấp là các mục tiêu cần đạt được trong thiết kế phần mềm.

Sử dụng các nguyên lý thiết kế phần mềm và mẫu thiết kế phần mềm để giảm phụ thuộc chồng chéo và tăng tính kết dính trong thiết kế Class UML là rất quan trọng.

- Trong thiết kế phần mềm, điều kiện ràng buộc là các yêu cầu bắt buộc phải tuân thủ trong quá trình phát triển phần mềm.
- Các điều kiện ràng buộc có thể đến từ nhiều nguồn khác nhau, ví dụ như yêu cầu chức năng, yêu cầu phi chức năng, các rào cản kỹ thuật, quy định pháp lý và quy trình nội bộ của công ty.
- Thiết kế phần mềm phải đáp ứng các điều kiện ràng buộc này để đảm bảo tính hoàn thiện và đáp ứng các yêu cầu của khách hàng.

Thiết kế phần mềm theo các điều kiện ràng buộc

Các loại điều kiện ràng buộc

- Điều kiện ràng buộc chức năng: là các yêu cầu về chức năng mà hệ thống phần mềm phải đáp ứng.
- Điều kiện ràng buộc phi chức năng: là các yêu cầu về các thuộc tính phi chức năng của hệ thống, ví dụ như độ tin cậy, hiệu suất và bảo mật.
- Điều kiện ràng buộc kỹ thuật: là các rào cản kỹ thuật mà hệ thống phải vượt qua, ví dụ như hạn chế kỹ thuật, môi trường phát triển và các giới hạn thiết kế.
- Điều kiện ràng buộc pháp lý: là các quy định và quy trình pháp lý mà hệ thống phải tuân thủ, ví dụ như quy định bảo mật, quy định về quyền riêng tư và quy trình kiểm tra phần mềm.
- Điều kiện ràng buộc nội bộ: là các quy định và quy trình nội bộ của công ty mà hệ thống phải tuân thủ.