TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

BÁO CÁO MÔN HỌC

Tìm hiểu về các mô thức lập trình

Sinh viên thực hiện: Lê Kiều Anh

MSSV: 20193979

Học phần: Kỹ thuật lập trình

Mã lớp: 128698

Giảng viên hướng dẫn: Thầy Vũ Đức Vượng

Hà Nội, ngày 09 tháng 10 năm 2021

Mục lục

Phần 1: Tóm tắt về các mô thức lập trình

1.1. Visual paradigm – Mô thức lập trình trực quan.

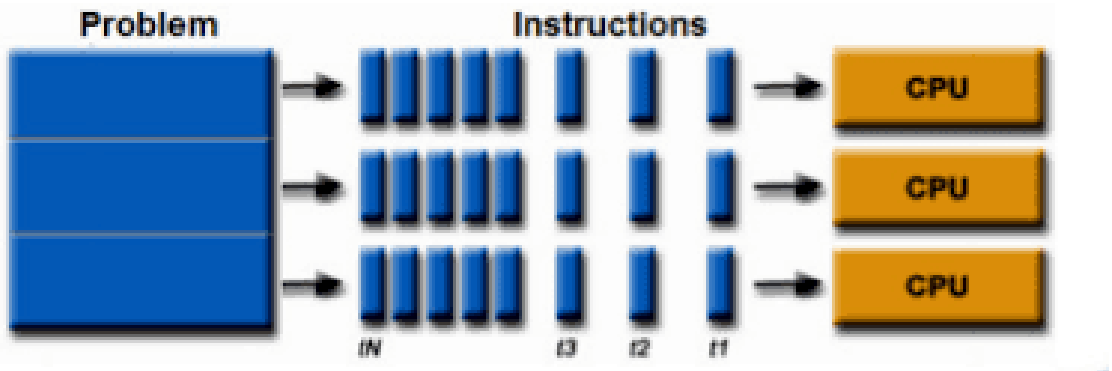
Lập trình trực quan được hiểu một cách đơn giản là sự kết hợp giữa đồ họa và máy tính. Ở đó máy tính và ngôn ngữ lập trình tương tác với nhau. Đây là chương trình ứng dụng được thực hiện dựa trên tình huống xảy ra. Nghĩa là lập trình viên sẽ sử dụng những cú pháp hay câu lệnh tương tự. Nhằm đồng thời gọi các câu lệnh song song.

Các khung giao diện hay còn được gọi là Interface sẽ được lập trình viên trực tiếp xây dựng. Đồng thời các ứng dụng được tạo ra thông qua các thao tác đơn giản trên màn hình. Những đối tượng như hộp thoại, nút điều khiển,… sẽ dựa vào những thuộc tính cho sẵn. Lập trình viên chỉ cần lựa chọn những thành phần đó dựa trên bảng có sẵn.

Không giống như những ngôn ngữ khác. Khi lập trình trực quan, lập trình viên chỉ cần khai báo khi gặp tình huống. Đặc biệt sẽ giảm tối thiểu việc lập trình viên tự viết các lệnh hay như chương trình rắc rối. Qua đó máy tính sẽ hiểu được việc cần phải làm thông qua mã lệnh, cú pháp. Sau đó tự động tạo ra những chương trình một cách dễ dàng.

1.2. Parallel paradigm – Mô thức lập trình song song.

Tính toán song song (Parallel Computing) là việc chia một công việc ra thành các công việc nhỏ và cho các công việc này thực hiện đồng thời với nhay bởi các hệ thống có nhiều bộ vi xử lý(multiprocessor) hay bộ vi xử lý đa nhân (multicore) nhằm giảm thời gian thực hiện công việc đó xuống. Việc lập trình để tách nhỏ công việc và sắp xếp để xử lý song song được gọi là **lập trình song song**.



Ứng dụng của việc sử dụng tính toán song song:

Tính toán song song là sự tiến hoá của tính toán tuần tự để cố gắng mô phỏng các trạng thái diễn ra trong thế giới tự nhiên: rất phức tạp, các sự kiện liên quan xảy ra cùng một thời điểm, nhưng trong cùng một chuỗi.

Ví dụ: Quỹ đạo hành tinh và thiên hà, Các mô hình thời tiết và đại dương, Kiến tạo địa chất, Giờ cao điểm ở Hà Nội, Dây truyền lắp ghép ô tô, Các hoạt động hàng ngày trong một doanh nghiệp, Xây dựng một trung tâm mua sắm…

Tính toán song song có thể được coi là "tính toán hiệu năng cao" và là động lực để mô phỏng cho các hệ thống phức tạp và giải quyết "các bài lớn" như: Dự báo thời tiết và khí hậu, Các phản ứng hoá học và hạt nhân, Các bài toán sinh học và gen người, Các quy trình sản xuất,…

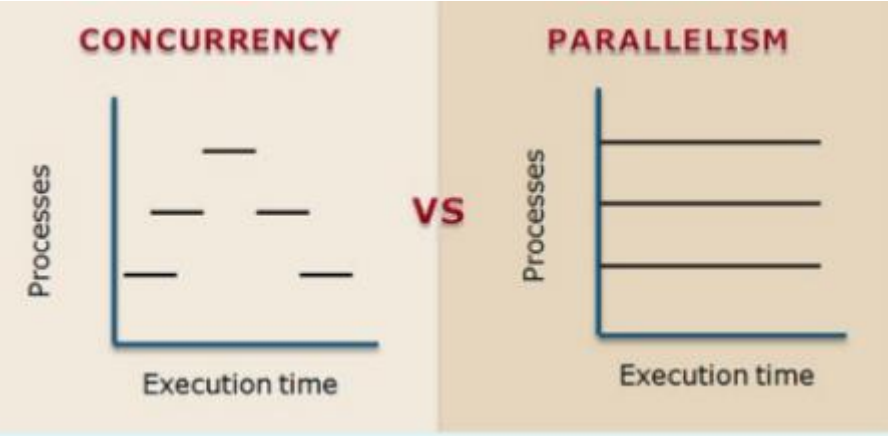
1.3. Concurrent paradigm – Mô thức lập trình tương tranh.

Lập trình tương tranh (lập trình đồng thời) là một hình thức điện toán trong đó một số tính toán được thực hiện trong các khoảng thời gian chồng chéo, đồng thời là một phần tử liên tục (một hoàn thành trước khi bắt đầu tiếp theo). Đây là một thuộc tính của một hệ thống, đây có thể là một chương trình riêng lẻ, một máy tính hoặc một mạng và có một điểm thực thi riêng hoặc luồng điều khiển cho mỗi tính toán (quy trình). Một hệ thống đồng thời là một hệ thống trong đó một tính toán có thể tiến lên mà không cần chờ tất cả các tính toán khác hoàn thành.

Cũng như một mô hình lập trình, lập trình tương tranh là một dạng lập trình module, cụ thể là chia nhỏ một tính toán tổng thể thành các tính toán con có thể được thực hiện đồng thời. Theo lời của Edsger Dijkstra: "Tương tranh xảy ra khi nhiều hơn một luồng thực thi có thể chạy đồng thời." Việc cùng sử dụng các tài nguyên dùng chung, chẳng hạn bộ nhớ hay file dữ liệu trên đĩa cứng, là nguồn gốc của nhiều khó khăn. Các tranh đoạt điều khiển (race condition) liên quan đến các tài nguyên dùng chung có thể dẫn đến ứng xử không đoán trước được của hệ thống. Việc sử dụng cơ chế loại trừ lẫn nhau (mutual exclusion) có thể ngăn chặn các tình huống chạy đua, nhưng có thể dẫn đến các vấn đề như tình trạng bế tắc (deadlock) và đói tài nguyên (resource starvation). Thiết kế của các hệ thống tương tranh thường là kết quả của việc tìm kiếm các kĩ thuật đáng tin cậy cho việc phối hợp hoạt động của thực thi, trao

đổi dữ liệu, cấp phát bộ nhớ và lập lịch thực thi để giảm tối thiểu thời gian phản ứng (response time) và tăng tối đa thông lượng (throughput).

Lập trình tương tranh và lập trình song song:



Lập trình tương tranh (đồng thời)	Lập trình song song
Đồng thời là khi hai hoặc nhiều nhiệm vụ chạy có thể bắt đầu, chạy và hoàn thành trong khoảng thời gian chồng chéo. Nó không nhất thiết là tất cả đều được chạy ngay lập tức. Ví dụ đa nhiệm trên một bộ xử lý đơn lõi.	Song song là khi các nhiện vụ chạy cùng một lúc. Ví dụ nhiều nhiệm vụ chạy cùng lúc trên bộ xử lý đa lõi.

1.4. Distributed paradigm – Mô thức lập trình phân tán.

Lập trình phân tán là một dạng của lập trình song song (tính toán song song). Lập trình song song tạo ra mối liên hệ giữa máy tính và các đơn vị tính toán, khiến chúng hoạt động đồng thời đối với một vấn đề cụ thể (dự báo thời tiết,…). Các đơn vị tính toán có thể đặt rất gần nhau hoặc tách rời nhau. Khi các đơn vị tính toán được đặt tách rời nhau, ta gọi đó là lập trình phân tán.

Với mô hình lập trình này, các đơn vị tính toán thường rất khác nhau, cũng như sự khác nhau giữa các hệ điều hành và thiết lập mạng máy tính. Những yếu tố đó khiến cho việc lập trình các tính toán của máy tính trở nên tương đối phức tạp và khó khăn. Lập trình mạng phân tán thường có 2 khái niệm chính: Peer to peer và client-server,… Peer to peer là lập trình ngang hàng giữa hai máy tính, client-server là lập trình cho phép n máy client kết nối với m máy server – một mô hình hay gặp trong thực tế.

1.5. Service-oriented paradigm – Mô thức lập trình hướng dịch vụ.

Lập trình hướng dịch vụ (Service-Oriented Programming) được xây dựng dựa trên mô hình hướng đối tượng (OOP), thêm vào đó những tiền đề giúp cho chúng ta có thể mô hình hoác các bài toán bằng những khái niệm của dịch vụ (Services), cái mà được cung cấp và sử dụng bởi các đối tượng.

Dịch vụ (Service) là một hành vi mang tính hợp đồng đã được định nghĩa trước. Các hành vi này có thể được thực hiện bởi bất kỳ thành phần nào, nó cũng có thể được cung cấp bởi bất kỳ thành phần nào dựa trên một hợp đồng duy nhất. Dịch vụ này có thể được sử dụng bởi dịch vụ khác.

Mô thình thành phần quy định rằng những bài toán về lập trình có thể được xem như những hộp đen, chúng được triển khai một cách độc lập và giao tieeps với nhau thông qua những hợp đồng.

Mô hình client-server truyền thống không định nghĩa trước những hợp đồng được public giữa client và server. Việc thực hiện của client và server cũng không thể độc lập với nhau. Một client chỉ có thể kết nối hay giao tiếp với một server duy nhất. Trong mô hình lập trình hướng dịch vụ thì có sự khác biệt, một client không bị bó buộc với bất kỳ server nào. Thay vào đó, vai trò người cung cấp dịch vụ có thể thay đổi qua lại giữa client và server.

Các thành phần của lập trình hướng dịch vụ

- Hợp đồng (contract)
- Thành phần (component)
- Bộ nối hay đầu nối (Connector)
- Bộ chứa (Container)
- Ngữ cảnh (Context)

Các đặc trưng của lập trình hướng dịch vụ:

- Tính liên kết (conjunctive)
- Khả năng triển khai (deployable)
- Tính lưu động (mobile)
- Tính sẵn sàng (available)
- Sự bảo mật (Security)

Phần 2: Tìm hiểu về mô thức lập trình hướng dịch vụ - Service oriented paradigm.

2.1. Các thành phần của lập trình hướng dịch vụ.

- Hợp đồng (contract): là một giao diện (interface) định nghĩa cú pháp và ngữ nghĩa của một hành vi.
- Thành phần (component): là các phần tử tính toán có khả năng triển khai, có khả năng sử dụng lại do tính độc lập cảu nó với các platform, giao thức (protocol) và môi trường triển khai.
- Bộ nối hay đầu nối (Connector): là một bản đóng gói cho phần chi tiết về phương thức vận tải (transport) đặc thù của một contract cụ thể nào đó. Đây cũng là mọt phần tử có khả năng triển khai được.
- Bộ chứa (Container): là môi trường cho việc thực thi các component. Nó chịu trách nhiệm quản lý tính sẵn sảng (availabilily) và bảo mật.
- Ngữ cảnh (Context): là môi trường cho việc triển khai các thành phần (bao gồm plug và play). Nó mô tả chi tiết việc cài đặt, bảo mật, khám phá và tìm kiếm.

2.2. Các đặc trưng của lập trình hướng dịch vụ.

- Tính liên kết (conjunctive): điều này nói đến khả năng sử dụng hoặc kết hợp các dịch vụ theo những cách khác với các cách thức đã tạo ra chúng. Nó hàm ý rằng những dịch vụ đó đã tạo ra những interface, cái mà chúng ta có thể dễ dàng nhận ra.
- Khả năng triển khai (deployable): Điều này nói đến khả năng triển khai hoặc sử dụng lại các thành phần trong bất kỳ môi trường nào. Điều này đòi hỏi sự độc lập về môi trường bao gồm sự độc lập về vận tải (transport), sự độc lập về nền tảng (platform) và sự độc lập về ngữ cảnh.
- Tính lưu động (mobile): Điều này nói đến khả năng di chuyển mã xung quanh mạng. Nó được sử dụng để di chuyển các proxy, giao diện người dùng và các dịch vụ di động.
- Tính sẵn sàng (available): Là một trong những tiền đề quan trọng của lập trình hướng dịch vụ, nghĩa là những mạng lưới nguồn tài nguyên dư thừa sẽ đảm bảo sự sẵn sàng cao cho lập trình hướng dịch vụ. Đây cũng là mục tiêu của lập trình hướng dịch vụ để xử lý các lỗi xảy ra trong tính toán phân bố.
- Sự bảo mật (Security): Những khái niệm của mã di động và những mạng lưới dịch vụ có khả năng khám phá đã đưa đến nhứng thách thức về mặt

bảo mật. Trong khi lập trình hướng dịch vụ cho phép giới hạn sử dụng các dịch vụ rộng hơn, nó không thể thành công nếu thiếu việc bảo vệ các dịch vụ trước việc sử dụng không đúng đắn.

2.3. Ngôn ngữ mô tả kiến trúc – ADL

Lập trình hướng dịch vụ định nghĩa ra một ngôn ngữ cho việc mô hình hóa dựa trên ngôn ngữ mô tả cấu trúc (Architecture Description Language). ADL bao gồm các khái niệm về các thành phần (components), các đầu nối (connectors), các vai trò (roles) và các cổng (ports). ADL là một ký hiệu chuẩn cho việc biểu diễn các kiến trúc giúp nâng cấp việc giao tiếp, hình mẫu cho các thiết kế ban đầu, cho việc tạo ra sự trừu tượng cho các hệ thống có thể chuyển nhượng, ADL sử dụng các hình vẽ để biểu diễn các thành phần, các thuộc tính, ngữ nghĩa của các kết nối, hành vi của toàn bộ hệ thống.

ADL là kết quả của việc các tiếp cận mang tính ngôn ngữ cho việc biểu diễn chính thức kiến trúc của hệ thống. ADL tập trung vào việc biểu diễn các thành phần của hệ thống. Các đặc trưng của ADL:

- ADL là một công cụ chính thức cho việc biểu diễn hệ thống.
- Con người và máy đều có khả năng đọc được ADL.
- ADL hỗ trợ việc mô tả hệ thống ở cấp độ cao hơn trước đây.
- ADL cho phép phân tích các kiến trúc – bao gồm sự đầy đủ, nhất quán, nhập nhằng, hiệu năng…
- Không có thỏa thuận chung cho việc ADL nên biểu diễn gì, đặc biệt là các hành vi của các bản kiến trúc.
- Cách biểu diễn hiện tại đang được sử dụng khó để phân tích cú pháp, không có công cụ thương mại nào hỗ trợ cho việc phân tích này.
- Đa số các ADL đều được tối ưu theo chiều dọc, một dạng đặc biệt của phân tích.

2.4. Kiến trúc hướng dịch vụ (SOA)

2.4.1. Khái niệm.

Về cơ bản, SOA là tập hợp toàn bộ các dịch vụ kết nối "mềm dẻo" với nhau và có giao tiếp. Chúng được định nghĩa một cách rõ ràng, hoàn toàn độc lập với nền tảng hệ thống và có thể tái sử dụng. Đây là cấp độ cao hơn của việc

phát triển ứng dụng chú trọng tới quy trình nghiệp vụ và sử dụng giao tiếp chuẩn để che đi sự phức tạp của kỹ thuật bên dưới.

2.4.2. Tính chất của hệ thống sử dụng kiến trúc hướng dịch vụ (SOA)

Loose coupling:

- Các module có tính loose coupling khi có một số ràng buộc được mô tả rõ ràng
- Hầu như mọi kiến trúc phần mềm đều hướng đến tính loose coupling giữa các module.
- Mức độ kết dính của mỗi hệ thống ảnh hưởng trực tiếp đến khả năng chỉnh sửa hệ thống của chính nó.

Sử dụng lại dịch vụ:

- Bởi vì các dịch vụ được cung cấp lên trên mạng và được đăng ký ở một nơi nhất định (service registry) nên chúng dễ dàng được tìm thấy và tái sử dụng.
- Tái sử dụng lại các dịch vụ giúp loại bỏ những thành phần trùng lắp, tăng độ vững chắc trong cài đặt, đơn giản hoá việc quản trị.
- Những dịch vụ được dùng chung bởi tất cả các ứng dụng của một hệ thống SOA gọi là những shared infrastructure service.

Khả năng cộng tác:

- Các hệ thống có thể giao tiếp với nhau trên nhiều nền tảng và ngôn ngữ khác nhau.
- Mỗi dịch vụ cung cấp một interface có thể được triệu gọi thông qua một dạng kết nối.
- Một kết nối gọi là interoperable chứa bên trong nó một giao thức và một định dạng dữ liệu mà mỗi client kết nối đến nó đều hiểu.

Tự phục hồi:

- Là một hệ thống có khả năng tự hồi phục sau khi bị lỗi mà không cần sự can thiệp của con người.

- Độ tin cậy (reliability) là mức độ đo khả năng một hệ thống xử lý tốt như thế nao trong tình trạng hỗn loạn; phụ thuộc vào khả năng phụ hồi của phần cứng sau khi bị lỗi.
- Nếu một thể hiện service nào đó không hoạt động thì một thể hiện khác vẫn có thể hoàn tất giao dịch cho khách hàng mà không bị ảnh hưởng gì.

2.4.3. Lợi ích của việc sử dụng kiến trúc hướng dịch vụ.

Mô hình SOA có nhiều ưu thế hơn so với truyền thống (cụ thể như mô hình ứng dụng hoặc mô hình hướng lập trình). Trong khi SOA chủ yếu tập trung nguồn lực để phát triển vào các chức năng và tính năng phục vụ hoạt động cũng như quy trình nghiệp vụ. Điều này cho phép nhà quản lý chỉ cần dựa trên những đặc điểm mang tính nghiệp vụ rà soát, xác định rõ ràng chi tiết, bổ sung các thành phần, sửa đổi hoặc loại bỏ chúng.

Vì vậy, hệ thống phần mềm phát triển phía sau có thể được thiết kế với mục đích đáp ứng các quy trình nghiệp vụ. Thay cho việc quy trình nghiệp vụ phải thay đổi để có thể tận dụng các tính năng phần mềm như trong các mô hình thường thấy ở nhiều cơ quan tổ chức với hạ tầng ứng dụng công nghệ thông tin đã phát triển trước đó.

Khi sử dụng mô hình SOA, các đơn vị cho phép hướng sự tập trung vào xây dựng các tính năng nghiệp vụ trong quá trình phát triển các phần mềm. Điều này mang lại một số lợi ích cho người dùng như sau:

- Giảm thiểu một khoản chi phí trong quá trình phát triển
- Giảm thiểu các yêu cầu về đào tạo và kỹ năng.
- Khoản phí bảo hành thấp
- Chu trình phát triển phần mềm nhanh chóng và dễ dàng hơn.
- Định hướng kinh doanh: SOA được ví như một bức tranh lớn của toàn bộ quy trình kinh doanh và dòng dịch chuyển của một công ty. Theo đó những người làm kinh doanh đầu tiên có thể hình dung toàn bộ quy trình được xây dựng theo quan điểm của công nghệ.
- Nâng cao vị thế của ngành công nghệ thông tin.

Với việc phát triển và tập hợp danh mục những sản phẩm/dịch vụ, các nhà phát triển có một bộ sưu tập những modun phần mềm có sẵn có thể dùng để lắp ghép lên một hệ thống mới. Danh mục này sẽ nhanh chóng được gia tăng về

quy mô và số lượng giúp cho việc phát triển các hệ thống mới thuận tiện và nhanh chóng hơn. Khả năng sử dụng lại dịch vụ này cũng cho phép giảm bớt chi phí phát sinh khi bổ sung thêm các tính năng mới vào hệ thống.

Tài liệu tham khảo

1. https://text.123docz.net/document/3294695-tim-hieu-ve-lap-trinh-huong-dich-vu-service-oriented-programming.htm

2. https://bkhost.vn/posts/soa-la-gi