**18.2.3. Máy ảo**

Kiến trúc máy ảo xem ứng dụng là chương trình được viết bằng ngôn ngữ sử dụng cho mục đích đặc biệt. Vì trình thông dịch phải có sẵn cho ngôn ngữ này nên kiến trúc chỉ xử lý thành công nếu chương trình được viết bằng ngôn ngữ đa chức năng.

Quá trình hiện thực hóa hệ thống máy ảo hoàn chỉnh yêu cầu việc xây dựng trình thông dịch. Thông dịch yêu cầu chúng ta gọi interpret() khi thực thi. Quá trình thông dịch những thành phần nguyên thủy (như CPU trong ví dụ chương 17, là những phần của một CPU không liên quan?) rất đơn giản (ví dụ, lấy CPU ra khỏi hộp chứa nó). Vấn đề là làm thế nào để thực thi hàm interpret() khi áp dụng với chương trình phức tạp hơn. Để làm được điều đó, chúng ta phải thiết kế pattern cho trình thông dịch.

Kiến trúc máy ảo có lợi nếu ứng dụng chứa nhiều tiến trình xử lý phần tử phức tạp, và nếu những phần tử này như trong ví dụ bên dưới có thể dễ dàng mô tả thông qua cấu trúc, ngữ pháp nào đó.

Ví dụ khác yêu cầu máy ảo là một ứng dụng cung cấp cho người dùng bình thường. Đối với người dùng bình thường sẽ có khả năng viết một đoạn script như sau:

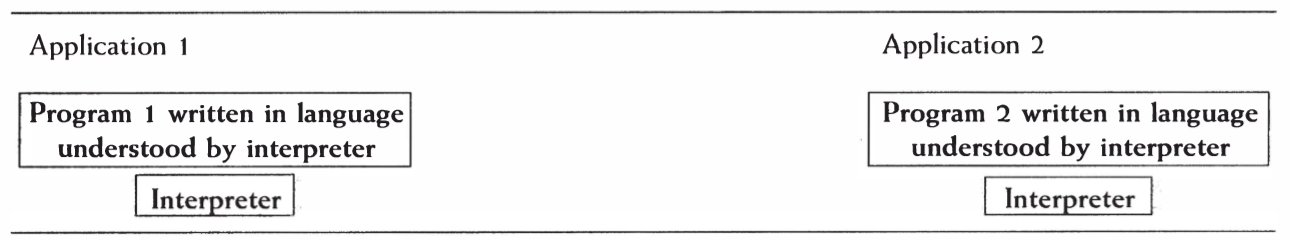
Balance checking/add excess to account + subtract deficit from saving;

Save report / c :Reports + standard headings + except replace "Ed' by "Al"

PrintReport / standard headings

e-mail report to Jayne@xyz. net .

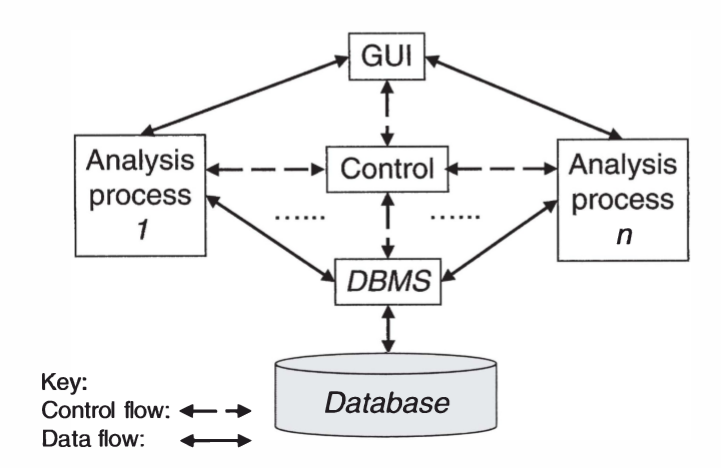
Kiến trúc máy ảo sẽ chuyển đổi và thông dịch đoạn script trên. Ý tưởng về kiến trúc được minh họa trong hình 18.16.



Hình 18.16 Kiến trúc máy ảo tận dụng khái niệm trình thông dịch để tạo điều kiện cho việc thực thi đa ứng dụng

**18.2.4. Kiến trúc repository**

Kiến trúc dùng để lưu trữ dữ liệu được gọi là repository (kho). Phiên bản phổ biến nhất là những hệ thống được thiết kế để thực thi giao dịch (trái lại với database). Ví dụ, một công ty điện duy trì database khách hàng bao gồm những thông tin chi tiết như lượng tiêu thụ / tháng, lịch sử thanh toán, sửa chữa, … Khi những nghiệp vụ này diễn ra có thể gây vấn đề cho database như thêm một khách hàng, truy vấn lịch sử chi trả, truy vấn danh sách khách hàng chầy cối hơn 3 tháng, … Một thiết kế phổ biến cho kiến trúc repository trong hình 18.17. Thiết kế này kết hợp luồng dữ liệu giữa các thực thể (nét liền) và quá trình điều khiển (nét đứt). “Điều khiển” nghĩa là thực thể này thao tác với thực thể khác, ví dụ tắt hoặc bật.



Hình 18.17. Kiến trúc repository điển hình

Ví dụ khác về ứng dụng kiến trúc repository là phát triển môi trường tương tác (IDE). IDE áp dụng dùng dể chỉnh sửa và biên dịch database (chính là các file và resource).

Một số ví dụ đơn giản của chúng ta không có database. Tuy nhiên, nếu có một game nhiều nhân vật, thì phải sử dụng database thay vì file để lưu trữ trạng thái các nhân vật. Điều này cần thiết khi chúng ta cho người chơi tương tác với game nhiều hơn, ví dụ in ra thống kê sức mạnh. SQL là cách phổ biến dùng để mô tả những truy vấn tương tự.

Kiến trúc blackboard, phát triển cho các ứng dụng AI, là kiểu repository hoạt động theo cơ chế post. Bạn đọc có thể xem [5] và [6] để biết chi tiết về kiến trúc này.

Kiến trúc repository cuối cùng chúng ta đề cập là hypertext. Ứng dụng phổ biến nhất chính là web, hoặc một ứng dụng quản lý ứng dụng khác.

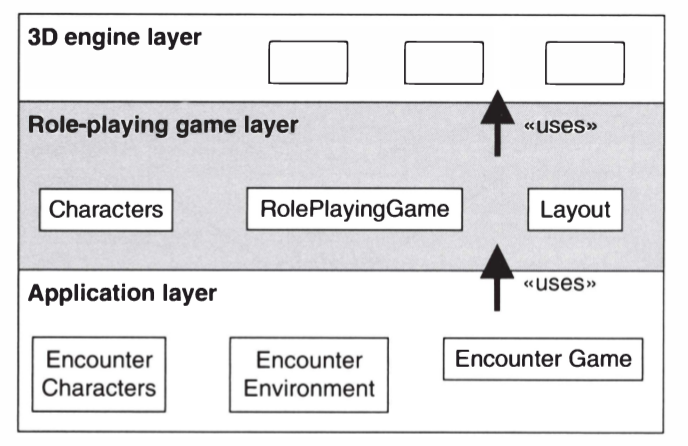
Từ “repository” thường được dùng trong công nghiệp để chứng tỏ một ứng dụng có thể cung cấp cái nhìn tổng quản về tập các database. Nó có thể liên quan đến một nhà kho chứa nhiều database. Respository không thay đổi cấu trúc của những database, nhưng chúng cung cấp cách truy cập thống nhất vào chúng. Đây là một trường hợp đặc biệt trong kiến trúc repository được định nghĩa bởi hai cụ tổ kiến trúc phần mềm Garlan và Shaw.

Kiến trúc repository chiếm một phần đáng kể trong ứng dụng, vì nhiều kiến trúc lấy databse làm trung tâm. Khi quá trình xử lý không đáng kể so với quá trình định dạng dữ liệu từ database, kiến trúc repository sẽ được đánh giá cao. Mặt khác, sự hiện diện của database lớn đôi khi che dấu sự thật rằng một lượng lớn xử lý có thể điều khiển cả kiến ​​trúc. Lập trình cơ sở dữ liệu có thể dễ dàng khiến ứng dụng lộn xộn, có lẽ có quan niệm khác với mô hình repository.

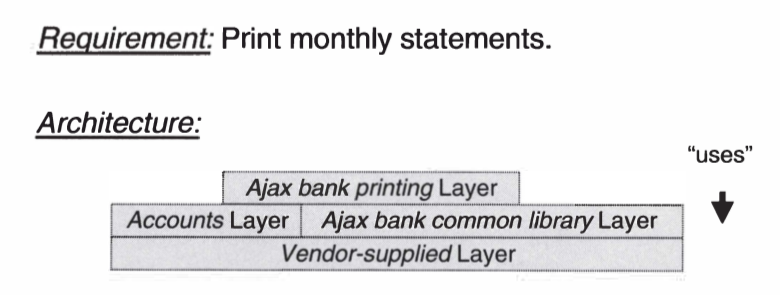
**18.2.5 Kiến trúc đa lớp**

Layer là tập các thành phần rõ ràng của phần mềm, điển hình là package. Ở dạng phổ biến, một layer sử dụng và được sử dụng bởi tối đa một layer khác. Xây dựng các ứng dụng theo từng layer có thể đơn giản hóa về mặt xử lý hơn. Một số layer, chẳng hạn như framework, có thể dùng để tạo ra ứng dụng khác.

Chúng ta đã thấy cách tiếp cận kiểu layer được áp dụng cho ứng dụng Giao tiếp, trong đó class được kế thừa từ class trong framework. Điều này được thể hiện trong Hình 18,18. Hình vẽ cho thấy cách chúng ta có thể tổ chức các class thuộc 3D Engine thành một layer có thể truy cập từ layer Role-playing.



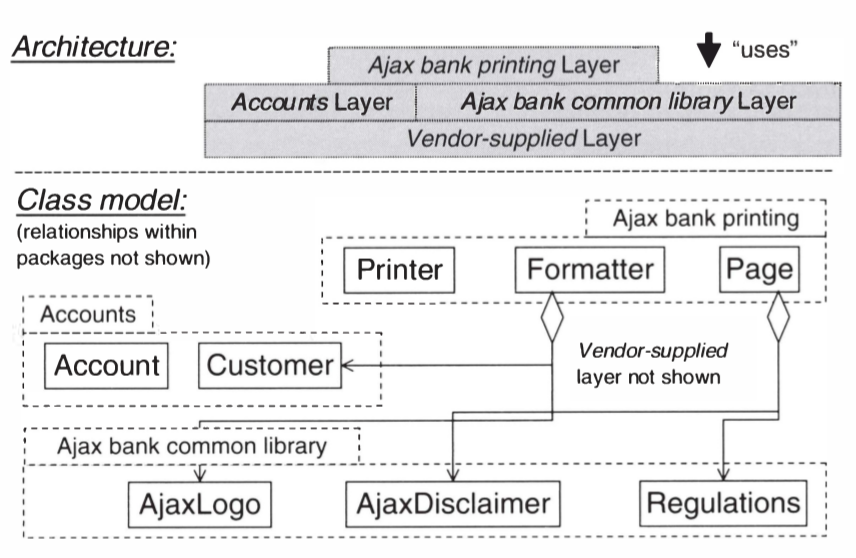
Hình 18.18. Kiến trúc đa lớp, ví dụ về 3D engine và video game.



Hình 18.9. Ví dụ là kiến trúc đa lớp về ngân hàng Ajax – lớp cao nhất (user)

Hình trên cho thấy ví dụ về kiến ​​trúc đa lớp của ngân hàng Ajax. Có 4 layer trong kiến ​​trúc này và cho thấy sự phụ thuộc theo hướng ngược lại so với Hình 18.18. Layer Ajax Bank Printing thực hiện việc in và định dạng. Nó được xây dựng dựa trên (sử dụng) layer Account và layer Ajax Bank Common Class. Cái thứ hai dựa trên layer vendor-supplied, chứa một số hàm như sắp xếp và tìm kiếm. Thông thường, layer được xem như package. Ví dụ, library Ajax là thư viện giao tiếp trực tiếp với người dùng nên nó bao gồm các class giải quyết vấn đề logo và các quy định.

Mối quan hệ using có thể kế thừa, tổng hợp hoặc tham chiếu. Ví dụ, tổng hợp có thể áp dụng giữa các lớp, như hình 18.20.



Hình 18.20. Kiến trúc đa lớp theo kiểu tổng hợp.

**18.2.6 Kiến trúc hướng Service**

Kiến trúc hướng Service (SOA) đang được sử dụng nhiều nơi. Chúng liên quan chặt chẽ với ý tưởng xem phần mềm như service và điện toán đám mây. SOA là sự kết hợp nhiều service: các component có chức năng khác nhau và interface của chúng. Chúng khác với nhiều kiến ​​trúc còn lại ở chỗ, mô tả tập các component tương tác linh hoạt với nhau, chứ không phải mô tả một ứng dụng duy nhất.

SOA là linh hồn cho các object tầng trên, bao gồm web service. SOA không nhất thiết phải hướng đối tượng. Trong trường hợp web service, các class toàn cục như đã cung cấp trong các ví dụ trước có thể không có. Ví dụ, giả sử rằng SOA trong ứng dụng business to business liên quan đến đặt hàng. Trong SOA, chúng ta sẽ không biết rằng class Order được biết đến và được sử dụng bởi tất cả các nhà cung cấp service và người tiêu dùng hay không. Các web service xử lý điều này bằng cách xác định một lược đồ cho cấu trúc dữ liệu Order và tham chiếu lược đồ khi có web service liên quan đến đặt hàng. Việc này sử dụng khả năng của web service gọi là Web Service Description language và có tác dụng làm class Order được khách hàng biết đến. Điều này tóm tắt trong hình 18.21 và 18.22.

Dựa vào việc component cung cấp chức năng thông qua interface:

* Chủ yếu thông qua web service.
* Theo tinh thần của lớp trên.
* Không nhất thiết phải OO

Ví dụ: Một ứng dụng liên quan đến đặt hàng.

* Không giả sử class Order được tất cả biết đến.
* Thay vào đó: Xác định lược đồ Order; tham chiếu khi web service có gì đó liên quan đến đặt hàng.

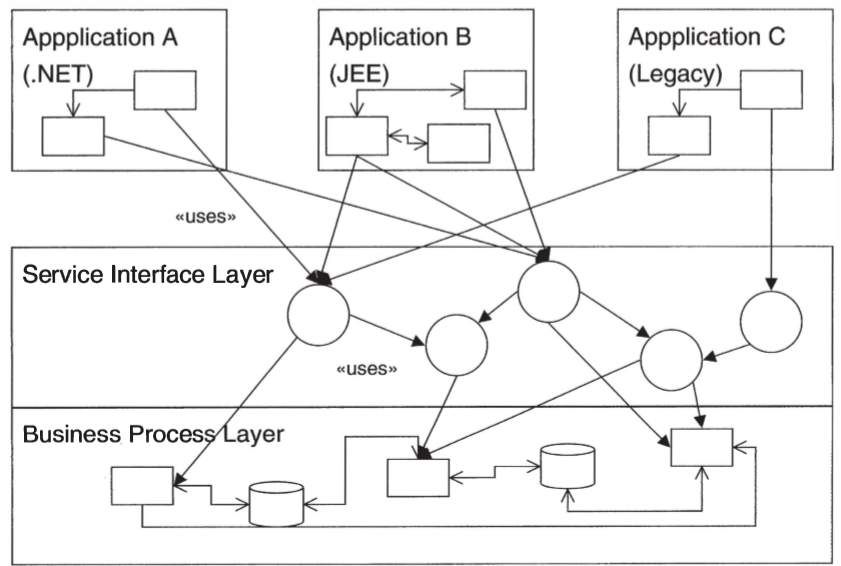
Hình 18.21. Kiến trúc hướng service, 1/2

* Thực hiện và quên
* Không lưu trữ trạng thái càng nhiều càng tốt
* Có thể mở rộng
* Thêm chức năng dễ dàng
* Có thể khám phá
* Tài khoản trong chất lượng service
* Ví dụ: bảo mật

Hình 18.22. Kiến trúc hướng service, 2/2

Kiến trúc hướng service cần môi trường kết nối mạng và các ứng dụng (hay thành phận của ứng dụng) không kết nối với nhau vĩnh viễn. Thay vào đó, SOA cho phép liên kết động các service. Ví dụ: giả sử bạn muốn viết ứng dụng đặt văn phòng phẩm cho công ty. Bạn sẽ muốn ứng dụng có thể xác định tất cả nhà cung cấp đủ điều kiện, kiểm tra giá cả và số lượng sẵn có, các điều khoản, chọn nhà cung cấp và đặt hàng. Để làm điều này, ứng dụng không thể luôn kết nối với một nhóm các nhà cung cấp. Do đó, SOA được xây dựng xung quanh một hệ thống đăng ký cập nhật. Hình 18.23 minh họa bốn bước liên quan đến việc công khai và truy cập service. 'Truy vấn "giống như tìm kiếm một doanh nghiệp trong danh bạ điện thoại." Binding' có nghĩa là liên hệ với service để gọi nó.

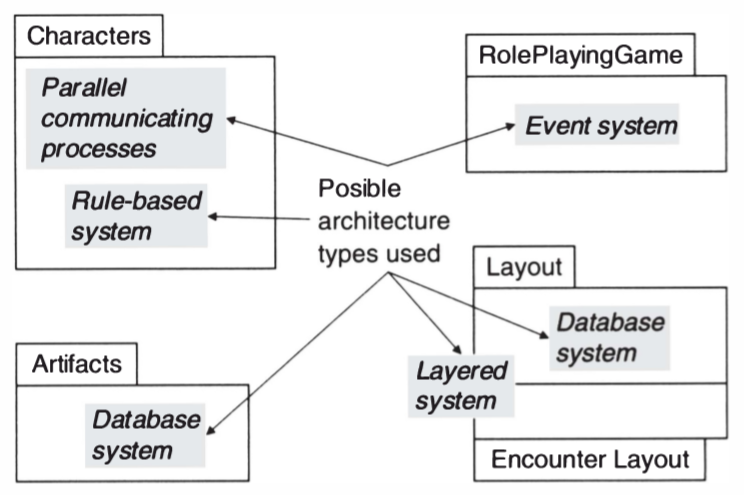
Kiến trúc hướng service thường xuyên sử dụng một layer quy trình nghiệp vụ. Layer này xác định các component của nghiệp vụ như database khách hàng, nghiệp vụ, hay chính sách tín dụng. Interface của service xác định các service sẵn có dựa trên quy trình nghiệp vụ. Nó chỉ định các chức năng như liệt kê tất cả khách hàng trong database và kiểm tra quy tắc kinh doanh trong giao dịch. Layer ứng dụng bao gồm các ứng dụng được xây bằng interface của layer service. Điều này được thể hiện trong Hình 18.24.



Hình 18.24. Layer hóa trong kiến trúc hướng service

**18.2.7 Sử dụng đa kiến ​​trúc trong một ứng dụng**

Các ứng dụng thường sử dụng các số kiến ​​trúc con trong tổng thể. Hình 18.25 cho thấy framework video game nhập vai sử dụng một số kiến ​​trúc được liệt kê bởi Carlin và Shaw. Chẳng hạn, package Artifacts làm database. nhân vật trong game là các tiến trình giao tiếp song song với nhau. Tổng thể trò chơi theo kiểu hệ thống hướng sự kiện.



Hình 18.25. Ví dụ về đa kiến trúc, bản mở rộng của video game.