**CHƯƠNG 6: THIẾT KẾ CẤU TRÚC**

**\* Mục tiêu:**

Mục tiêu chương 6 để giới thiệu những khái niệm của kiến trúc phần mềm và thiết kế cấu trúc. Khi ta tìm hiểu chương này, ta có thể:

Hiểu được tại sao việc thiết kế phần mềm rất quan trọng

Hiểu được những quyết định cần có về kiến trúc hệ thống trong suốt quá trình thiết kế cấu trúc.

Được giới thiệu về mẫu ý tưởng cấu trúc, những cách thử về tổ chức hệ thống kiến trúc mà nó có thể tái sử dụng trong quá trình thiết kế hệ thống.

Hiểu thêm về mẫu kiến trúc thường được sử dụng ở những kiểu hệ thống ứng dụng khác nhau, bao gồm quá trình chuyển đổi hệ thống và quá trình thao tác với ngôn ngữ.

**\* Nội dung:**

6.1. Những quyết định thiết kế cấu trúc

6.2. Tầm nhìn về cấu trúc

6.3. Mẫu cấu trúc

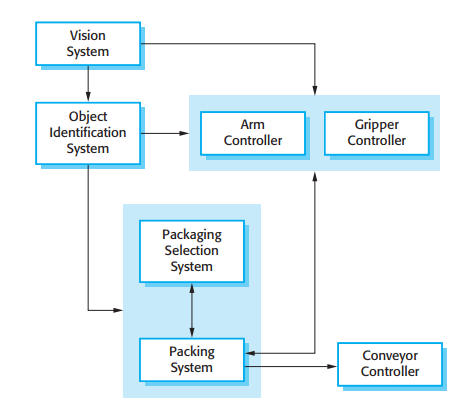
6.4. Ứng dụng cấu trúc

Thiết kế cấu trúc liên quan đến cách mà hệ thống, cấu trúc chung sẽ được tổ chức và thiết kế. Trong mô hình của quá trình phát triển phần mềm, thiết kế cấu trúc là mảng đầu tiên của quá trình thiết kế. Đây là liên kết quan trọng giữa thiết kế và yêu cầu kỹ thuật, vì nó định danh thành phần cấu trúc chính và mối quan hệ của chúng trong hệ thống. Đầu ra của quá trình thiết kế cấu trúc là một mô hình cấu trúc, chúng miêu tả cách mà hệ thống được tổ chức như là một tập hợp các thành phần có giao tiếp với nhau.

Ta có thể thiết kế kiến trúc phần mềm ở 2 cấp trừu tượng (nhỏ và lớn):

Cấu trúc nhỏ: Liên quan đến kiến trúc của những chương trình cho cá nhân. Ta quan tâm đến cách mà chương trình được bóc tách thành các thành phần.

Cấu trúc lớn: Liên quan đến cấu trúc của hệ thống những doanh nghiệp phức tạp. Hệ thống này được phân phối đến nhiều máy tính khác nhau, có thể được quản lý bởi nhiều công ty.



*Hình 6.1: Kiến trúc bên ngoài của một hệ thống điều khiển robot.*

Kiến trúc phần mềm rất quan trọng vì nó ảnh hưởng đến việc trình bày, biểu diễn độ mạnh, khả năng phân phối và khả năng bảo trì của hệ thống. Những thành phần cá nhân triển khai hệ thống các hàm yêu cầu. Những yêu cầu phi chức năng phụ thuộc vào kiến trúc hệ thống – cách mà những thành phần được tổ chức và giao tiếp với nhau. Trong một số hệ thống, những yêu cầu phi chức năng cũng bị ảnh hưởng bởi những thành phần cụ thể, nhưng không nghi ngờ gì khi kiến trúc hệ thống mới là phần ảnh hưởng quan trọng nhất.

Sau đây là 3 ưu điểm của việc thiết kế và ghi lại chính xác kiến trúc phần mềm:

Giao tiếp với các bên liên quan: Kiến trúc là sự biểu diễn cấp cao của hệ thống, có thể được sử dụng tập trung để miêu tả một phạm vi lớn những bên có liên quan.

Khả năng phân tích hệ thống: Làm rõ cấu trúc hệ thống ở ngay những bước đầu trong quá trình phát triển. Thiết kế cấu trúc có ảnh hưởng lớn đến việc liệu hệ thống có đáp ứng được các nhu cầu cần biểu diễn, độ tin cậy và khả năng bảo trì hay không.

Tái sử dụng quy mô lớn: Mô hình cấu trúc hệ thống là bản miêu tả gọn nhẹ, có thể quản lý về cách hệ thống được tổ chức, cách mà các thành phần tương tác. Nó thường giống với hệ thống ở mức độ yêu cầu đơn giản, vì vậy nó có thể hỗ trợ tái sử dụng những phần mềm quy mô lớn.

Hofmeister et al. (2000) đã đề xuất một kiến trúc phần mềm có thể sử dụng như là bản thiết kế cho việc đàm phán những yêu cầu hệ thống, thứ hai nó giúp xây dựng lên những quyết định với khách hàng, nhà phát triển và nhà quản lý. Nó che giấu chi tiết và cho phép nhà thiết kế tập trung vào phần mấu chốt của hệ thống.

Cấu trúc hệ thống thường được xây dựng thành các sơ đồ khối, mỗi hộp trong mô hình biểu diễn 1 thành phần. Những hộp bên trong chỉ ra thành phần nào đã được phân tách thành các thành phần phụ. Mũi tên nghĩa là dữ liệu và tín hiệu điều khiển được truyền từ thành phần này qua thành phần khác có hướng.

Sơ đồ khối trình bày bức tranh cấp cao về cấu trúc hệ tống, mà mọi người từ các ngành khác nhau, những người tham gia vào quá trình phát triển hệ thống đều dễ hiểu. Tuy nhiên, bất chấp việc chúng được sử dụng rộng rãi, Bass et al. (2003) không thích các sơ đồ khối không chính thức để mô tả một kiến ​​trúc. Họ cho rằng những sơ đồ không chính thức là những biểu diễn kiến ​​trúc kém, vì chúng không hiển thị loại mối quan hệ giữa các thành phần hệ thống cũng như các thành phần bên ngoài thuộc tính nhìn thấy được.

Những mâu thuẫn rõ ràng giữa thực hành và lý thuyết kiến ​​trúc nảy sinh bởi vì có hai cách mà mô hình kiến ​​trúc của một chương trình được sử dụng:

1. Như một cách để tạo điều kiện cho việc thảo luận về thiết kế hệ thống. Một mức cao khung nhìn kiến ​​trúc của một hệ thống rất hữu ích cho việc giao tiếp với các bên liên quan đến hệ thống và lập kế hoạch dự án vì nó không lộn xộn với các chi tiết. Các bên liên quan có thể liên quan đến nó và hiểu một cái nhìn trừu tượng về hệ thống. Họ sau đó có thể thảo luận về toàn bộ hệ thống mà không bị nhầm lẫn chi tiết. Các mô hình kiến ​​trúc xác định các thành phần chính sẽ được phát triển. Vì vậy người quản lý có thể bắt đầu chỉ định mọi người lập kế hoạch để phát triển các hệ thống.

2. Như một cách ghi lại một kiến ​​trúc đã được thiết kế Mục đích ở đây là tạo ra một mô hình hệ thống hoàn chỉnh cho thấy các thành phần khác nhau trong hệ thống, giao diện của chúng và kết nối của chúng. Lập luận cho điều này là mô tả kiến ​​trúc chi tiết như vậy giúp dễ hiểu và phát triển hơn hệ thống.

Sơ đồ khối là một cách thích hợp để mô tả kiến ​​trúc hệ thống trong quá trình thiết kế, vì chúng là một cách tốt để hỗ trợ truyền thông giữa những người tham gia vào quá trình. Trong nhiều dự án, chúng thường là chỉ tài liệu kiến ​​trúc tồn tại. Tuy nhiên, nếu kiến ​​trúc của một hệ thống được ghi chép kỹ lưỡng, sau đó tốt hơn là sử dụng ký hiệu với ngữ nghĩa để mô tả kiến ​​trúc.

**6.1 Quyết định liên quan đến thiết kế cấu trúc**

Thiết kế cấu trúc là quá trình sáng tạo, ta thiết kế một hệ thống sẽ đáp ứng những yêu cầu chức năng và phi chức năng. Những hoạt động trong quá trình phụ thuộc vào kiểu hệ thống, nền tảng và kinh nghiệm của kiến trúc kỹ thuật, những yêu cầu cụ thể hệ thống Vì vậy, có thể thấy rằng thiết kế kiến trúc như một chuỗi các quyết định cần đưa ra để hoàn thành mục tiêu.

Trong suốt quá trình thiết kế kiến trúc, ta cần đưa ra nhiều quyết định dựa trên cả kiến thức và kinh nghiệm, phải xem xét những câu hỏi về chức năng như sau:

*Có phần mềm nào có thể làm khung cho hệ thống không?*

*Hệ thống sẽ được phân phối ra sao trên một số lõi và bộ vi xử lý?*

*Kiến trúc mẫu nào sẽ được dùng?*

*Cách tiếp cận chức năng nào sẽ được cấu trúc cho hệ thống?*

*Các thành phần cấu trúc nào sẽ được phân tách thành thành phần con?*

*Chiến lược nào sẽ được dùng để điều khiển hệ thống?*

*Cách tổ chức kiến trúc nào tốt nhất để đáp ứng những yêu cầu phi chức năng?*

*Đánh giá thiết kế cấu trúc bằng cách nào?*

*Ghi lại kiến trúc hệ thống như thế nào?*

Mặc dù mỗi phần mềm là duy nhất, hệ thống trong một miền ứng dụng sẽ có những kiến trúc giống nhau, phản ánh khái niệm chức năng chung.

Những hệ thống nhúng được thiết kế cho máy tính cá nhân, nó thường chỉ có một bộ xử lý và ta không cần thiết kế hệ thống phân phối. Tuy nhiên, hầu như các hệ thống lớn là các hệ phân phối qua nhiều máy, việc lựa chọn kiến trúc phân phối là quyết định quan trọng, ảnh hưởng đến chất lượng của hệ thống.

Vì mối quan hệ gần gũi giữa yêu cầu chức năng và phi chức năng, kiến trúc phần mềm, những kiểu kiến trúc cụ thể nên ta sẽ phải chọn hệ thống phụ thuộc vào một số yêu cầu phi chức năng:

+ Hiệu suất: Nếu hiệu suất là một yêu cầu quan trọng, kiến trúc phải được thiết kế để xác định các hoạt động quan trọng trong một số thành phần, tất cả các thành phần này được triển khai trên một máy tính thay vì trên hệ thống. Làm giảm sự giao tiếp giữa các thành phần.

+ Bảo vệ: Cấu trúc lớp và việc tạo lớp nên được sử dụng, lớp quan trọng nhất cần được bảo vệ cao.

+ Độ an toàn: Kiến trúc nên được thiết kế để những hoạt động cần độ an toàn được nằm trong một thành phần hoặc giới hạn trong một số thành phần. Nó giảm chi phí và các vấn đề liên quan đến an toàn và ta có thể cung cấp những hệ thống bảo vệ liên quan, hạn chế tối đa rủi ro.

+ Sẵn có: Hệ thống cần được thiết kế bởi nhiều thành phần để dễ dàng thay thế và cập nhật mà hệ thống không bị gián đoạn.

+ Khả năng bảo trì: Kiến trúc hệ thống cần được thiết kế các thành phần khép kín để sẵn sàng thay đổi. Nhà cung cấp dữ liệu sẽ phải được tách biệt với người dùng và phải hạn chế việc chia sẻ cấu trúc.

+ Ta cần đánh giá và chọn chiến lược cho đúng vì hiển nhiên những xung đột tiềm năng có thể xảy ra. Ta cần cân nhắc và có thể chia hệ thống thành phần nhỏ với mỗi phần là một kiểu đặc trưng riêng.

**6.2. Những quan điểm về kiến trúc**

Trong phần này tập trung vào 2 vấn đề như sau:

*Quan điểm khi thiết kế và ghi lại kiến trúc hệ thống?*

*Ký hiệu nào sẽ được dùng để miêu tả hệ thống?*

Ta không thể biểu diễn hết các thành phần liên quan đến hệ thống trong 1 mô hình, mỗi mô hình là một cách nhìn về hệ thống. Nó sẽ chỉ ra cách mà hệ thống được chia ra thành các module, cách mà thời gian chạy sẽ tương tác hoặc những cách khác nhau mà các thành phần chạy qua mạng.

Có những quan điểm khác nhau, Krutchen (1995), trong quan điểm về mô hình nổi tiếng 4 +1, đề xuất 4 quan điểm khác nhau về kiến trúc chức năng:

+ Quan điểm logic: Chìa khoá của hệ thống như các đối tượng hoặc các lớp đối tượng. Có thể liên kết yêu cầu hệ thống đến toàn bộ quan điểm logic.

+ Quan điểm về quá trình xử lý: Nó chỉ ra cách mà hệ thống hoạt động trong run-time. Cách nhìn hữu hiệu để đánh giá về những đặc tính phi chức năng hệ thống như hiệu suất và khả năng sẵn có.

+ Quan điểm về sự phát triển: Chỉ ra cách mà phần mềm sẽ được phân tách để phát triển. Có thể các thành phần được chia ra bởi nhiều nhóm phát triển. Quan điểm này hữu dụng cho kiến trúc hệ thống và phát triển hệ thống.

+ Quan điểm vật lý: Chỉ ra phần cứng và cách mà phần mềm được phân phối qua những bộ vi xử lý khác nhau.

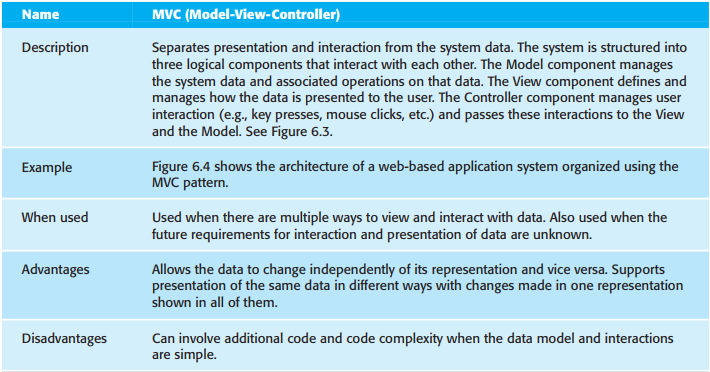
+ Trong thực tế, những quan điểm về mặt khái niệm luôn được phát triển trong suốt quá trình thiết kế. Đó là cách kết nối các thành phần quan trọng của hệ thống đến các bên liên quan. Trong quá trình thực thi, những quan điểm khác có thể được phát triển, những khía cạnh khác được thảo luận nhưng không cần phải thảo luận tất cả các quan điểm.

**6.3. Mô hình kiến trúc**

Ý tưởng về xây dựng mô hình như một cách để biểu diễn, chia sẻ và tái sử dụng kiến thức về hệ thống phần mềm được sử dụng rộng rãi ngày nay. Việc kích hoạt điều này chính là sử dụng các mẫu thiết kế hướng đối tượng.

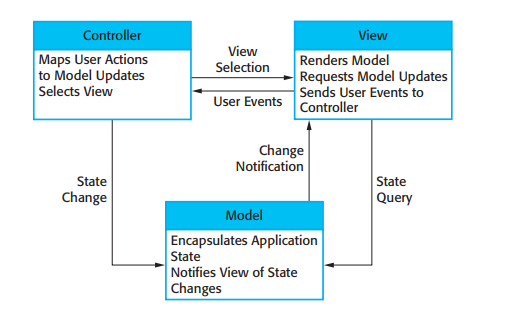
Phần này giới thiệu một vài mô hình kiến trúc và cơ bản miêu tả việc chọn mô hình kiến trúc cho một số loại hệ thống khác nhau. Ta có thể nghĩ về việc chọn mô hình kiến trúc như là bản miêu tả, có thể được kiểm tra và thử nghiệm ở nhiều môi trường và hệ thống khác nhau. Nhưng, một mô hình kiến trúc cần miêu tả thành công tổ chức của hệ thống trước đó. Nó bao gồm thông tin về sự phù hợp hay không phù hợp, cũng như điểm mạnh yếu của mẫu.

Ví dụ như mô hình MVC dưới đây. Đây là mẫu cơ bản hiện nay để tương tác và quản lý các hệ thống web. Nó bao gồm tên mô hình, bản miêu tả vắn tắt (mô hình đồ hoạ liên quan), ví dụ về loại hệ thống được áp dụng mẫu. Có thể thêm thông tin về khi nào mẫu được sử dụng, mặt lợi và hại.



*Hình 6.2: Mô hình chung MVC.*

Mô hình được miêu tả như dưới đây, biểu diễn kiến trúc từ những quan điểm khác nhau.



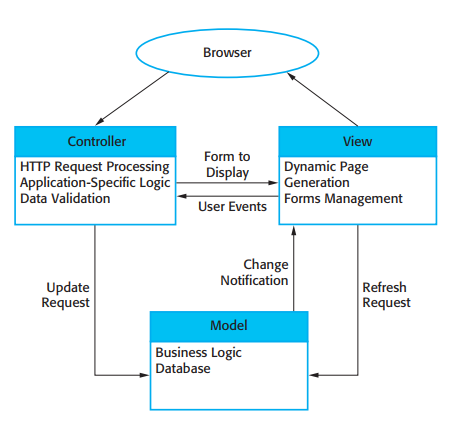
*Hình 6.3: Cách thức tổ chức của mô hình MVC.*

Hình trên miêu tả quan điểm về mặt khái niệm ngữ nghĩa.

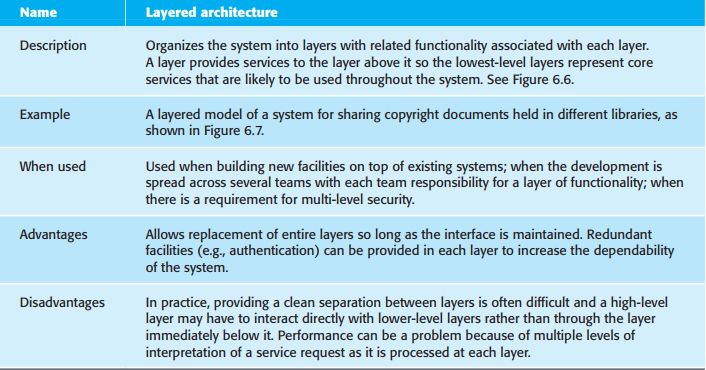
Còn hình 6.4 dưới đây miêu tả kiến trúc trong thời gian chạy, khi mô hình được sử dụng để tương tác cũng như quản lý hệ thống web.

**6.3.1. Kiến trúc phân lớp**

Quan điểm về việc chia tách và độc lập các phần là những quan điểm cơ bản trong thiết kế kiến trúc. Nó cho phép sự thay đổi. Mô hình MVC chia hệ thống ra làm nhiều phần tử, cho phép chúng thay đổi độc lập nhau. Ví dụ, thêm một view mới hoặc thay đổi một view đã có sẽ không ảnh hưởng, làm thay đổi đến dữ liệu trong model. Kiến trúc phân lớp là một cách để chia tách và độc lập các thành phần. Mô hình dưới đây chia hệ thống thành nhiều lớp riêng biệt, mỗi lớp chỉ phụ thuộc vào cơ sở vật chất, dịch vụ của lớp ngay dưới nó cung cấp.



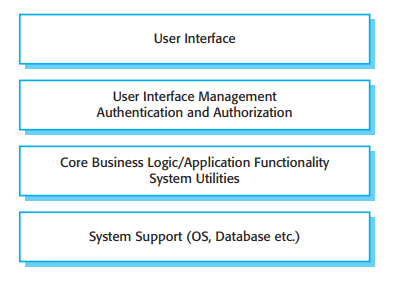
*Hình 6.4: Kiến trúc ứng dụng web sử dụng mô hình MVC*



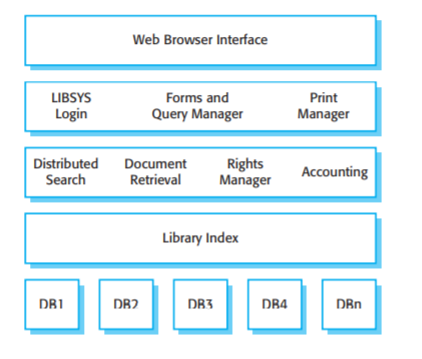
*Hình 6.5: Các lớp mô hình kiến trúc chung*

Cách tiếp cận theo lớp hỗ trợ khả năng phát triển hệ thống. Khi một lớp được phát triển, một số dịch vụ cung cấp bởi lớp đó sẵn sàng phục vụ người dùng. Kiến trúc cũng có thể thay đổi và dễ đóng gói. Miễn là giao diện không thay đổi, một lớp có thể được thay thế bởi lớp khác tương đương. Hơn thế, khi lớp giao diện thay đổi hoặc một vài thiết bị được thêm vào lớp, chỉ lớp kề với lớp đó bị ảnh hưởng. Vì vậy dễ dàng hơn cho việc triển khai trên nhiều nền tảng và ứng dụng. Chỉ có bên trong, những lớp phụ thuộc vào phần cứng cần phải triển khai lại để phù hợp với hệ điều hành khác nhau hoặc cơ sở dữ liệu khác nhau.

Hình dưới đây là một ví dụ về kiến trúc phân lớp với 4 lớp. Lớp thấp nhất bao gồm hệ thống hỗ trợ phần mềm, điển hình là cơ sở dữ liệu và hệ điều hành. Lớp tiếp theo là lớp ứng dụng, bao gồm những thành phần liên quan đến ứng dụng chức năng và thành phần tiện ích, được sử dụng bởi những thành phần ứng dụng khác. Lớp thứ 3 liên quan đến giao diện người dùng quản lý và cung cấp cho người dùng quyền xác thực, lớp trên cùng cung cấp cơ sở giao diện người dùng. Tất nhiên, số lượng lớp là ngẫu nhiên. Mỗi lớp có thể được chia ra thành nhiều lớp nhỏ hơn, tuỳ vào quan điểm mỗi người.



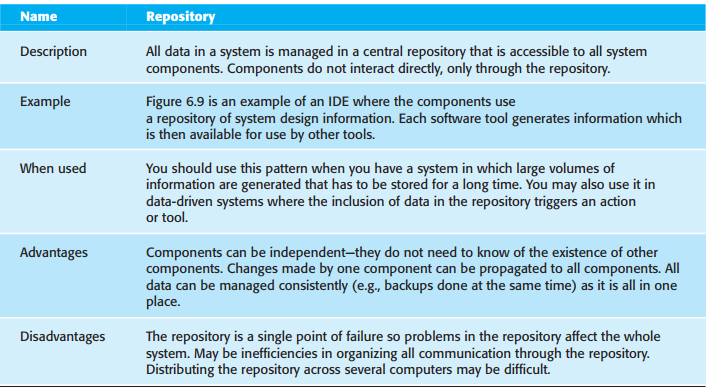
*Hình 6.6: Một kiến trúc chung nhiều lớp.*



*Hình 6.7: Kiến trúc của hệ thống LIBSYS.*

**6.3.2. Kiến trúc kho lưu trữ**

Kiến trúc phân lớp và mô hình MVC là ví dụ cho những mô hình mà quan điểm dựa trên tổ chức khái niệm hệ thống. Ví dụ sau đây về kiến trúc kho lưu trữ, miêu tả cách mà tập hợp các thành phần tương tác chia sẻ dữ liệu.

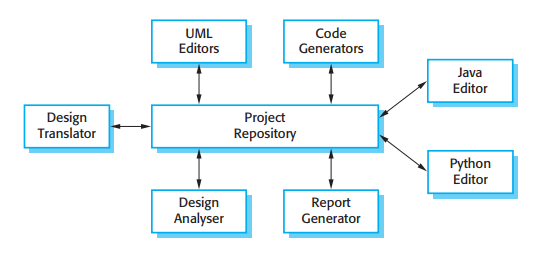


*Hình 6.8: Mô hình kho lưu trữ chung.*

Vấn đề chính của hệ thống là sử dụng nguồn dữ liệu lớn và được tổ chức quanh một cơ sở dữ liệu cũng như kho lưu trữ. Mô hình phải phù hợp với ứng dụng mà dữ liệu được tạo ra từ một thành phần và tiếp tục được sử dụng bởi các thành phần khác. Ví dụ cho phần này là hệ thống bao gồm chỉ huy và điểu khiển, quản lý thong tin, hệ thống CAD, môi trường phát triển tương tác cho phần mềm.

Hình dưới đây minh hoạ cho trường hợp kho lưu trữ được sử dụng. Biểu đồ được trình bày bao gồm những công cụ khác nhau hỗ trợ phát triển mô hình. Kho lưu trữ có thể là phiên bản của môi trường điều khiển, theo dõi những thay đổi của phần mềm, cho phép trở lại phiên bản trước đó.

Việc tổ chức những công cụ quanh kho lưu trữ là cách hiệu quả để chia sẻ lượng lớn dữ liệu. Không cần phải chuyển dữ liệu chính xác từ một thành phần qua thành phần khác. Tuy nhiên, những thành phần phải hoạt động quanh kho lưu trữ. Tuy nhiên không thể tránh khỏi, có những thoả hiệp giữa những yêu cầu cụ thể cho các công cụ, và điều đó gây khó khăn hoặc không thể tích hợp những thành phần mới, nếu như chúng không phù hợp. Trong thực tế, mặc dù có thể phân phối tập trung vào kho lưu trữ, tuy nhiên cũng có thể có những vấn đề với dữ liệu dư thừa và không nhất quán.



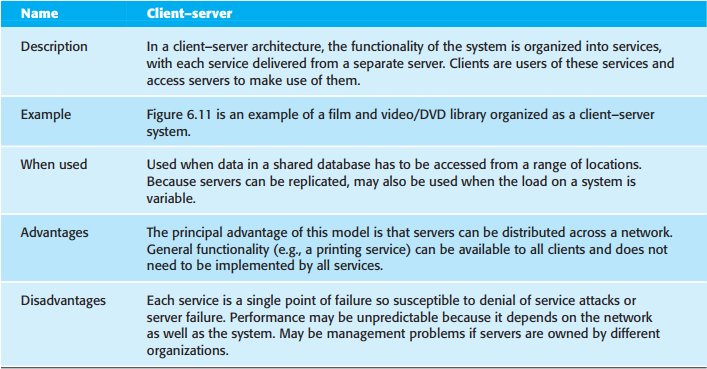
*Hình 6.9: Mô hình kho lưu trữ cho một IDE.*

Trong ví dụ trên, kho lưu trữ là bị động và việc điều khiển kho là trách nhiệm của các thành phần sử dụng dữ liệu. Một cách tiếp cận khác, có nguồn gốc từ hệ thống AI, dùng mô hình bảng đen để kích hoạt các thành phần khi một dữ liệu cụ thể trở nên sẵn có. Nó phù hợp khi dạng của kho lưu trữ có cấu trúc không được tốt lắm. Quyết định về các công cụ để kích hoạt chỉ có thể được đưa ra khi dữ liệu cần phân tích.

**6.3.3. Kiến trúc client – server**

Kho lưu trữ là kiến trúc tĩnh của hệ thống và không được trình bày ra trong tổ chức thời gian chạy. Ví dụ sau đây minh hoạ tổ chức thể hiện trong thời gian chạy phổ biến cho các hệ phân tán. Mô hình client – server:

Hệ thống tuân theo mô hình client-server được tổ chức như một tập hợp các dịch vụ, và các máy chủ được liên kết và các client truy cập và sử dụng các dịch vụ. Các thành phần chính của mô hình này là:



*Hình 6.10: Mô hình client – server.*

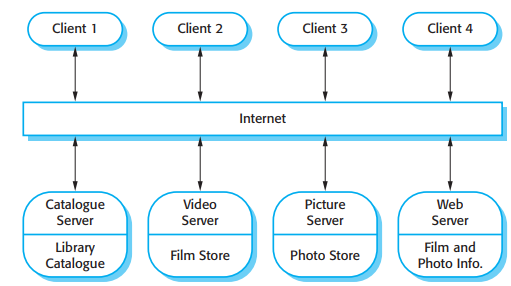
1. Một tập hợp các máy chủ cung cấp dịch vụ cho các thành phần khác. Ví dụ về máy chủ bao gồm máy chủ in cung cấp dịch vụ in, máy chủ file cung cấp dịch vụ quản lý tệp và máy chủ biên dịch cung cấp ngôn ngữ lập trình dịch vụ biên dịch.

2. Một tập hợp các máy khách gọi các dịch vụ do máy chủ cung cấp. Sẽ có bình thường là một số trường hợp của một chương trình khách hàng đang thực thi đồng thời trên các máy vi tính.

3. Một mạng cho phép khách hàng truy cập các dịch vụ này. Hầu hết hệ thống được thực hiện dưới dạng hệ thống phân tán, được kết nối bằng giao thức Internet.

Cấu trúc client-server thường được nghĩ như là hệ thống phân tán, nhưng mô hình logic của những dịch vụ độc lập chạy trên những server riêng biệt có thể được triển khai trên 1 máy. Đó lại là lợi ích quan trọng của việc chia tách và độc lập các thành phần. Dịch vụ và server có thể bị thay đổi mà không ảnh hưởng đến các thành phần khác trong hệ thống.

Client phải biết tên server và dịch vụ nó cung cấp. Tuy nhiên, server không cần phải biết định danh của client hoặc có bao nhiêu client truy cập dịch vụ. Client truy cập dịch vụ thông qua gọi thủ tục từ xa, gửi đến một yêu cầu dựa trên giao thức như http, sử dụng trong www. Client gửi yêu cầu và đợi server phản hồi.



*Hình 6.11: Kiến trúc client – server cho thư viện film.*

Hình trên là minh hoạ cho hệ thống dựa trên mô hình client – server. Nó đa người dùng, hệ thống web cung cấp film và thư viện ảnh. Trong hệ thống, một vài server quản lý việc hiển thị những loại khác nhau của truyền thông. Khung video cần chuyển đổi nhanh và đồng bộ nhưng ở độ phân giải thấp.

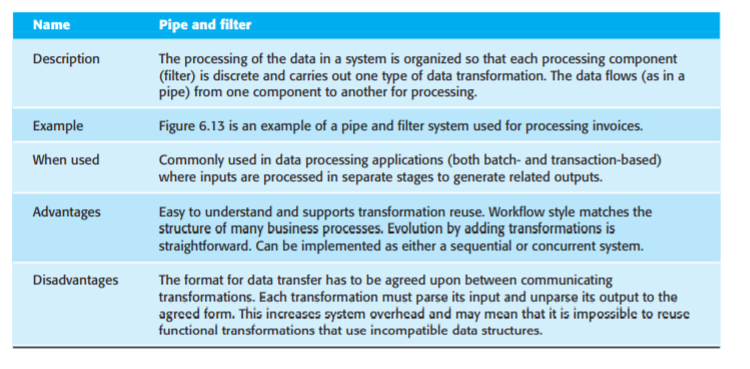
Các danh mục cần có khả năng giải quyết nhiều câu truy vấn và cung cấp link đến thông tin hệ thống web, bao gồm dữ liệu về film và video, thư điện tử hỗ trợ bán ảnh, films và video. Đây là chương trình có giao diện tích hợp người dùng, cấu trúc dựa trên web browser để truy cập dịch vụ.

Ưu điểm quan trọng nhất của mô hình client-server là nó là một kiến ​​trúc phân tán. Việc sử dụng hiệu quả có thể được thực hiện bởi các hệ thống được nối mạng với nhiều bộ xử lý. Dễ dàng thêm một máy chủ mới và tích hợp nó với phần còn lại của hệ thống hoặc để nâng cấp máy chủ một cách minh bạch mà không ảnh hưởng đến các phần khác của hệ thống.

**6.3.4. Kiến trúc ống và bộ lọc**

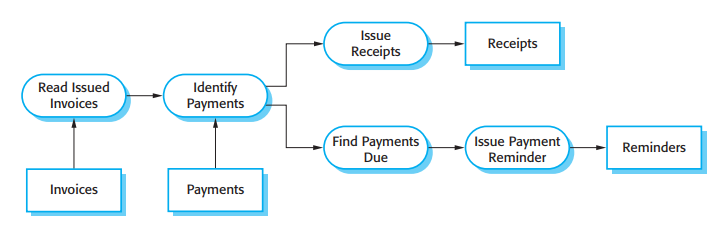
Ví dụ cuối cùng về kiến trúc là mô hình kiến trúc ống và bộ lọc. Đây là một mô hình tổ chức thời gian vận hành của một hệ thống nơi chuyển đổi chức năng xử lý đầu vào của họ và sản xuất đầu ra. Dữ liệu chảy từ nơi này sang vùng khác và được chuyển đổi khi nó di chuyển qua chuỗi. Mỗi bước xử lý được thực hiện dưới dạng biến đổi. Dữ liệu đầu vào chảy qua các biến đổi này cho đến khi được chuyển đổi thành đầu ra. Các phép biến đổi có thể thực hiện tuần tự hoặc song song. Dữ liệu có thể được xử lý bởi từng biến đổi từng mục hoặc trong một lô duy nhất.

Tên “đường ống và bộ lọc” xuất phát từ hệ thống Unix ban đầu, nơi có thể liên kết các quy trình bằng cách sử dụng 'đường ống'. Những điều này đã chuyển một luồng văn bản từ một quy trình tới khác. Các hệ thống tuân theo mô hình này có thể được thực hiện bằng cách kết hợp lệnh Unix, sử dụng các đường ống và các phương tiện điều khiển của Unix shell. Thuật ngữ “bộ lọc” được sử dụng vì phép chuyển đổi   
“lọc ra” dữ liệu mà nó có thể xử lý từ đầu vào của nó dòng dữ liệu. Các biến thể của mẫu này đã được sử dụng kể từ lần đầu tiên máy tính được sử dụng để xử lý dữ liệu tự động. Khi các phép biến đổi diễn ra tuần tự với dữ liệu được xử lý trong lô, mô hình kiến ​​trúc ống và bộ lọc này trở thành mô hình tuần tự theo lô, kiến trúc chung cho hệ thống xử lý dữ liệu (ví dụ: hệ thống thanh toán). Kiến trúc của một hệ thống nhúng cũng có thể được tổ chức như một đường dẫn quy trình, với mỗi quá trình thực hiện đồng thời.



*Hình 6.12: Mô hình chung kiến trúc ống và bộ lọc.*

Ví dụ về kiểu kiến ​​trúc hệ thống này, được sử dụng nhiều trong ứng dụng xử lý hàng loạt, được trình bày trong hình dưới đây. Một tổ chức đã xuất hóa đơn cho khách hàng.



*Hình 6.13: Một ví dụ về kiến trúc ống và bộ lọc.*

Mỗi tuần một lần, các khoản thanh toán đã thực hiện được đối chiếu với các hóa đơn. Đối với những hóa đơn đã được thanh toán, một biên lai sẽ được phát hành. Đối với những hóa đơn không được thanh toán trong thời gian thanh toán cho phép, một lời nhắc nhở được đưa ra. Hệ thống tương tác khó viết bằng cách sử dụng mô hình ống và bộ lọc vì nhu cầu về một luồng dữ liệu được xử lý. Mặc dù đầu vào văn bản đơn giản và đầu ra có thể được mô hình hóa theo cách này, giao diện người dùng đồ họa phức tạp hơn.

Định dạng I / O và chiến lược kiểm soát dựa trên các sự kiện như nhấp chuột hoặc lựa chọn menu. Rất khó để có thể dịch nó thành một dạng tương thích với mô hình pipelining.

**6.4. Kiến trúc ứng dụng**

Các hệ thống ứng dụng nhằm đáp ứng nhu cầu của doanh nghiệp hoặc tổ chức. Tất cả các doanh nghiệp đều có nhiều điểm chung – họ cần thuê người, xuất hóa đơn, giữ tài khoản, v.v. Các doanh nghiệp hoạt động trong cùng một lĩnh vực sử dụng các ứng dụng cụ thể của từng lĩnh vực chung. Do đó, cũng như các chức năng kinh doanh chung, tất cả điện thoại các công ty cần hệ thống để kết nối cuộc gọi, quản lý mạng của họ, xuất hóa đơn cho khách hàng, v.v. Do đó, các hệ thống ứng dụng được các doanh nghiệp này sử dụng cũng có nhiều điểm chung. Những điểm chung này đã dẫn đến sự phát triển của kiến ​​trúc phần mềm mô tả cấu trúc và tổ chức của các loại hệ thống phần mềm cụ thể. Các kiến ​​trúc ứng dụng đóng gói các đặc điểm chính của một lớp của hệ thống. Ví dụ: trong các hệ thống thời gian thực, có thể có các mô hình kiến ​​trúc chung của các loại hệ thống khác nhau, chẳng hạn như hệ thống thu thập dữ liệu hoặc hệ thống giám sát. Mặc dù các trường hợp của các hệ thống này khác nhau về chi tiết, cấu trúc kiến ​​trúc chung có thể được tái sử dụng khi phát triển các hệ thống mới của cùng một loại. Kiến trúc ứng dụng có thể được triển khai lại khi phát triển nhưng, đối với nhiều hệ thống kinh doanh, có thể sử dụng lại ứng dụng mà không cần thực hiện lại. Chúng tôi thấy điều này trong sự phát triển của Hoạch định Nguồn lực Doanh nghiệp (ERP) hệ thống từ các công ty như SAP và Oracle, và các gói phần mềm dọc (COTS) cho các ứng dụng chuyên biệt trong các lĩnh vực kinh doanh khác nhau. Trong các hệ thống này, một hệ thống chung được cấu hình và điều chỉnh để tạo ra một ứng dụng kinh doanh cụ thể.

Ví dụ, đối với một hệ thống cho quản lý các chuỗi cung ứng có thể được điều chỉnh cho các loại nhà cung cấp, hàng hóa hay các thỏa thuận hợp đồng khác nhau. Là một nhà thiết kế phần mềm, bạn có thể sử dụng các mô hình kiến trúc ứng dụng theo một số cách sau:

1. Nếu bạn không quen với kiểu cách ứng dụng mà bạn đang phát triển, bạn có thể dựa trên thiết kế ban đầu của mình trên một kiến trúc ứng dụng chung làm điểm khởi đầu cho quá trình thiết kế kiến trúc của bạn. Tất nhiên, điều này sẽ phải chuyên biệt đối với từng hệ thống cụ thể đang được phát triển, nhưng nó sẽ là một điểm khởi đầu tốt cho các thiết kế của bạn.
2. Nếu bạn đã phát triển một cấu trúc thiết kế cho một hệ thống ứng dụng, bạn có thể so sánh điều này với kiến trúc ứng dụng chung. Khi đó, cái bạn có chính là một danh sách sự giống và khác nhau của thiết kế. Bạn có thể kiểm tra xem thiết kế của mình có phù hợp với kiến trúc chung hay không?
3. Các kiến trúc ứng dụng xác định các đặc điểm cấu trúc ổn định của hệ thống kiến trúc và trong nhiều trường hợp, bạn có thể sử dụng để phát triển song song những điều này. Bạn có thể giao việc cho các thành viên trong nhóm để triển khai các thành phần khác nhau trong kiến trúc.
4. Nếu bạn có các thành phần mà bạn có thể sử dụng lại, bạn có thể đưa ra so sánh những cấu trúc này với cấu trúc chung để xem liệu có các thành phần trong kiến trúc có thể sử dụng được hay không? Khi đó, mô hình sẽ là một phương tiện để đánh giá các phần để sử dụng lại.
5. Nếu bạn đang thảo luận về một ứng dụng cụ thể, hãy cố gắng so sánh các ứng dụng cùng loại. Khi đó, bạn có thể sử dụng các khái niệm trong kiến trúc chung để nói về các ứng dụng.

Có thể thấy, có nhiều loại hệ thống ứng dụng và trong một số trường hợp, chúng dường như khác biệt với nhau. Tuy nhiên, nhiều ứng dụng có bề ngoài không giống nhau này thực sự có nhiều điểm chung với nhau và do đó bạn có thể biểu diễn được chúng bằng một kiến trúc ứng dụng trừu tượng duy nhất. Tôi có thể minh họa điều này bằng cách mô tả các kiến trúc của hai loại ứng dụng:

1. *Các ứng dụng xử lý giao dịch*

Các ứng dụng xử lý giao dịch là các ứng dụng tập trung vào cơ sở dữ liệu, xử lý các yêu cầu của người dùng về thông tin và cập nhật thông tin trong cơ sở dữ liệu. Đây là loại hệ thống kinh doanh tương tác phổ biến nhất. Chúng được tổ chức theo cách mà hành động của người dùng không thể can thiệp lẫn nhau và đảm bảo tính toàn vẹn của cơ sở dữ liệu luôn được duy trì. Loại hệ thống này bao gồm hệ thống tương tác ngân hàng, hệ thống thương mại điện tử, hệ thống thông tin và hệ thống đặt lịch.

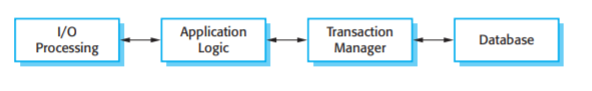
1. *Hệ thống xử lý ngôn ngữ*

Hệ thống xử lý ngôn ngữ là một hệ thống ngầm mà ý định của người dùng được thể hiện bằng một ngôn ngữ chính thức (chẳng hạn như Java). Hệ thống có chức năng xử lý các ngôn ngữ này thành một định dạng nội bộ và sau đó diễn giải đại diện nội bộ này. Các hệ thống xử lý ngôn ngữ nổi tiếng nhất là các trình biên dịch, dịch các chương trình ngôn ngữ bậc cao thành ngôn ngữ máy. Tuy nhiên, các hệ thống xử lý ngôn ngữ cũng được sử dụng để diễn giải các ngôn ngữ lệnh cho cơ sở dữ liệu và hệ thống thông tin, các ngôn ngữ đánh dấu như XML (Harold và Means, 2002; Hunter và cộng sự, 2007).

Tôi đã chọn những loại hệ thống cụ thể này bởi vì một số lượng lớn các hệ thống kinh doanh dựa trên web là hệ thống cử lý giao dịch và tất cả việc phát triển phần mềm đều dựa vào hệ thống xử lý ngôn ngữ.

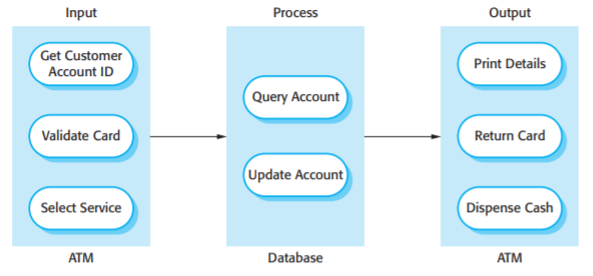
**6.4.1. Hệ thống xử lý giao dịch**

Hệ thống xử lý giao dịch (TP) được thiết kế để xử lý các yêu cầu của người dùng về thông tin từ cơ sở dữ liệu hoặc các yêu cầu cập nhật cơ sở dữ liệu (Lewis và cộng sự, 2003). Về mặt kỹ thuật, một giao dịch cơ sở dữ liệu là một chuỗi các hoạt động được coi như một đơn vị duy nhất (một đơn vị nguyên tử). Tất cả các hoạt động trong một giao dịch phải được hoàn thành trước khi các thay đổi cơ sở dữ liệu được thực hiện sau đó. Điều này sẽ đảm bảo rằng các hoạt động trong giao dịch bị lỗi cũng không dẫn tới sự mâu thuẫn trong cơ sở dữ liệu. Từ góc độ của người dùng, giao dịch được coi là bất kỳ các chuỗi tập hợp mà đáp ứng mục tiêu, chẳng hạn như “tìm giờ của các chuyến bay từ London đến Paris”. Nếu người dùng giao dịch không yêu cầu cơ sở dữ liệu được thay đổi thì có thể không cần đóng gói cơ sở dữ liệu như một giao dịch cơ sở dữ liệu kỹ thuật. Ví dụ về giao dịch là yêu cầu khách hàng rút tiền từ tài khoản sử dụng máy ATM. Điều này liên quan tới việc lấy thông tin chi tiết về tài khoản của khách hàng, kiểm tra số dư theo số tiền đã rút và gửi lệnh đến máy ATM để chuyển tiền mặt. Giao dịch sẽ không hoàn thành và cơ sở dữ liệu tài khoản khách hàng không được thay đổi cho đến khi tất cả các bước được hoàn thành.



*Hình 6.14: Cấu trúc của hệ thống xử lý giao dịch*

Hệ thống xử lý giao dịch thường là hệ thống tương tác trong đó người dùng thực hiện các yêu cầu không đồng bộ cho dịch vụ. Hình 6.14 minh họa cấu trúc kiến trúc của các ứng dụng TP. Đầu tiên, người dùng đưa ra yêu cầu đối với hệ thống thông qua một thành phần xử lý I/O. Yêu cầu được xử lý bởi một số logic cụ thể của ứng dụng. Một giao dịch được tạo và chuyển cho hệ thống quản lý giao dịch, đó là thường được nhúng trong hệ quản trị cơ sở dữ liệu. Sau khi hệ thống quản lý giao dịch đã đảm bảo rằng giao dịch được hoàn thành đúng cách. Nó sẽ báo cho ứng dụng rằng quá trình xử lý đã kết thúc.



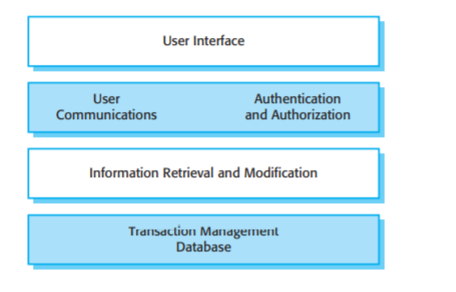
*Hình 6.15: Kiến trúc phần mềm của một hệ thống ATM*

Hệ thống xử lý giao dịch có thể được tổ chức dưới dạng kiến trúc “đường ống và bộ lọc” với các thành phần hệ thống chịu trách nhiệm nhập, xử lý và đầu ra. Đối với ví dụ trên, hãy xem xét một hệ thống ngân hàng cho phép khách hàng truy vấn tài khoản của họ và rút tiền mặt từ máy ATM. Hệ thộng bao gồm hai phần mềm hợp tác: phần mềm dành cho ATM và phần mềm xử lý tài khoản trong máy chủ cơ sở dữ liệu của ngân hàng. Các thành phần đầu vào và đầu ra được triển khai dưới dạng phần mềm trong máy ATM và thành phần xử lý là một phần của máy chỉ cơ sở dữ liệu của ngân hàng. Hình 6.15 cho thấy kiến trúc của hệ thống này, minh họa các chức năng của thành phần đầu vào, quá trình xử lý và đầu ra.

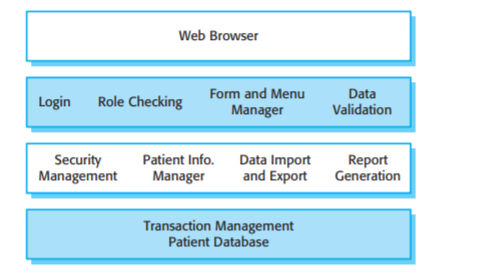
**6.4.2. Hệ thống xử lý thông tin**

Tất cả các hệ thống liên quan đến tương tác với việc chia sẻ cơ sở dữ liệu có thể được coi là hệ thống thông tin dựa trên giao dịch. Một hệ thống thông tin cho phép kiểm soát truy cập vào cơ sở thông tin lớn, chẳng hạn như danh mục thư viện, lịch trình chuyến bay hoặc hồ sơ của bệnh nhân trong bệnh viện. Ngày càng có nhiều hệ thống thông tin dựa trên web hệ thống được truy cập thông qua trình duyệt web.

Hình 6.16 mô tả một mô hình chung của một hệ thống thông tin. Hệ thống được mô hình hóa bằng cách sử dụng phương pháp phân lớp (được thảo luận trong phần 6.3) trong đó lớp trên cùng hỗ trợ giao diện người dùng và lớp dưới cùng là cơ sở dữ liệu của hệ thống. Lớp giao tiếp người dùng xử lý tất cả đầu vào và đầu ra từ giao diện người dùng và thông tin lớp truy xuất bao gồm logic dành riêng cho ứng dụng để truy cập và cập nhật cơ sở dữ liệu. Như chúng ta sẽ thấy, các lớp trong mô hình này có thể ánh xạ trực tiếp lên các máy chủ trong một hệ thống dựa trên Internet.



*Hình 6.16: Cấu trúc các lớp trong hệ thống thông tin*



*Hình 6.17: Kiến trúc của MHC – PMS*

Giống như một ví dụ về sự khởi tạo mô hình phân lớp, hình 6.17 cho thấy kiến trúc của MHC – PMS. Nhớ lại rằng hệ thống này duy trì và quản lý chi tiết thông tin của những bệnh nhân đang tham khảo ý kiến bác sĩ chuyên khoa về các vấn đề sức khỏe tinh thần. Tôi đã thêm chi tiết vào từng lớp trong mô hình bằng cách xác định các thành phần hỗ trợ thông tin liên lạc của người dùng và truy xuất thông tin và truy cập:

1. Lớp trên cùng chịu trách nhiệm triển khai giao diện người dùng. Trong trường hợp này, giao diện của người dùng đã được triển khai bằng các trình duyệt web.
2. Lớp thứ hai cung cấp chức năng giao diện người dùng được phân phối thông qua trình duyệt web. Nó bao gồm các thành phần để cho phép người dùng đăng nhập vào hệ thống và kiểm tra các thành phần đảm bảo rằng các hoạt động mà chúng sử dụng được cho phép bởi vai trò của chúng. Lóp này bao gồm các thành phần quản lý bảng, biểu mẫu và menu trình bày thông tin cho người dùng và các thành phần các thực dữ liệu, kiểm tra tính nhất quán của thông tin.
3. Lớp thứ ba thực hiện chức năng cung cấp các thành phần thực hiện bảo mật hệ thống, tạo và cập nhật thông tin bệnh nhân, nhập và xuất dữ liệu bệnh nhân từ các cơ sở dữ liệu khác và trình tạo báo cáo, tạo các báo cáo quản lý.
4. Cuối cùng, lớp thấp nhất được xây dựng bằng hệ thống quản lý cơ sở dữ liệu thương mại, cung cấp khả năng quản lý giao dịch và lưu trữ dữ liệu liên tục.

Hệ thống quản lý tài nguyên và thông tin hiện nay thường là hệ thống dựa trên web nơi giao diện người dùng được triển khai bằng trình duyệt web. Ví dụ, hệ thống thương mại điện tử là hệ thống quản lý tài nguyên dựa trên internet chấp nhận đơn đặt hàng điện tử cho hàng hóa hoặc dịch vụ và sau đó sắp xếp việc phân phối hàng hóa hoặc dịch vụ này cho khách hàng. Trong hệ thống thương mại điện tử, lớp dành riêng cho ứng dụng bao gồm chức năng bổ sung hỗ trợ “giỏ hàng” trong đó người dùng có thể đặt một số các mục trong các giao dịch riêng biệt sau đó thanh toán cho tất cả chúng cùng nhau trong một giao dịch duy nhất. Việc tổ chức các máy chủ trong hệ thống này thường phản ánh bốn lớp mô hình chung được trình bày trong hình 6.16. Các hệ thống này thường được thực hiện như mô hình client – server với server nhiều tầng:

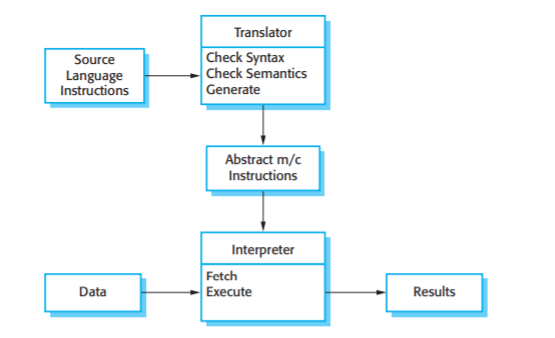
1. Máy chủ web chịu trách nhiệm về tất cả các giao tiếp của người dùng, với giao diện người dùng, được triển khai bằng trình duyệt web.
2. Máy chủ ứng dụng chịu trách nhiệm triển khai ứng dụng cụ thể một cách logic cũng như các yêu cầu lưu trữ và truy xuất thông tin.
3. Máy chủ cơ sở dữ liệu di chuyển thông tin đến và đi từ cơ sở dữ liệu, xử lý quản lý giao dịch thông qua các thủ tục.

Sử dụng nhiều máy chủ cho phép thông lượng cao và có thể xử lý hàng trăm giao dịch mỗi phút. Khi nhu cầu tăng lên, các máy chủ có thể được thêm vào mỗi cấp độ để đối phó với các xử lý bổ sung liên quan.

**6.4.3. Hệ thống xử lý ngôn ngữ**

Hệ thống xử lý ngôn ngữ dịch một ngôn ngữ tự nhiên hoặc nhân tạo sang một ngôn ngữ khác biểu diễn của ngôn ngữ đó và đối với các ngôn ngữ lập trình cũng có thể thực thi mã kết quả. Trong kỹ thuật phần mềm, trình biên dịch dịch một ngôn ngữ lập trình nhân tạo thành mã máy. Các hệ thống xử lý ngôn ngữ khác có thể dịch mô tả dữ liệu XML thành các lệnh để truy vấn cơ sở dữ liệu hoặc nhằm thay thế biểu diễn XML. Các hệ thống xử lý ngôn ngữ tự nhiên có thể dịch một ngôn ngữ khác. Ví dụ, tiếng Pháp dịch sang tiếng Na Uy.

Một cấu trúc khả thi cho một hệ thống xử lý ngôn ngữ cho một ngôn ngữ lập trình được minh họa trong hình 6.18. Các hướng dẫn ngôn ngữ nguồn được xác định dùng nhằm xác định chương trình sẽ được thực thi và một trình dịch chuyển đổi chúng thành các hướng dẫn cho bản hướng dẫn sử dụng. Các hướng dẫn này sau đó được định nghĩa bởi một thành phần khác có chức năng tìm, thực thi và thực hiện chúng bằng cách sử dụng dữ liệu từ đầu vào. Đầu ra của quá trình là kết quả của việc diễn giải các hướng dẫn trên dữ liệu đầu vào.

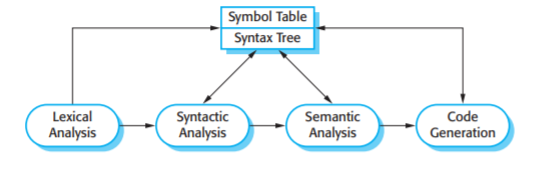


*Hình 6.18: Cấu trúc của một hệ thống xử lý ngôn ngữ*

Tất nhiên, đối với nhiều trình biên dịch, trình thông dịch là một đơn vị phần cứng xử lý các hướng dẫn và máy ảo là một bộ xử lý thực. Tuy nhiên, đối với các ngôn ngữ được nhập động, chẳng hạn như Python, trình thông dịch có thể là một phần mềm thành phần.

Trình biên dịch ngôn ngữ lập trình là một phần của chương trình tổng quát bao gồm các thành phần:

1. Một máy phân tích từ vựng, lấy mã thông báo ngôn ngữ đầu vào và chuyển đổi chúng thành hình thức bên trong.
2. Một bảng ký hiệu, chứa thông tin về tên của các thực thể (biến, tên lớp, tên đối tượng,…) được sử dụng trong văn bản đang được dịch.
3. Một bộ phân tích cú pháp, kiểm tra cú pháp của ngôn ngữ được dịch. Nó sử dụng một ngữ pháp xác định ngôn ngữ và xây dựng một cây cú pháp.
4. Cây cú pháp, là một cấu trúc nội tại bên trong đại diện cho chương trình đang biên dịch.
5. Một trình phân tích ngữ nghĩa sử dụng thông tin từ cây cú pháp và bảng ký hiệu để kiểm tra tính đúng đắn về ngữ nghĩa của văn bản ngôn ngữ đầu vào.
6. Một trình tạo mã dựa trên cây cú pháp và tạo mã máy ảo.



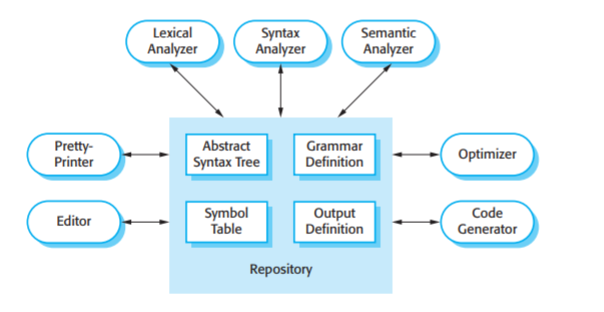
*Hình 6.19: Hệ thống đường ống và bộ lọc cho trình biên dịch.*

Các thành phần khác cũng có thể được bao gồm lại để phân tích và chuyển đổi cây cú pháp để cải thiện độ hiệu quảvà loại bỏ thành phần dư thừa khỏi máy được tạo mã. Trong các loại hệ thống xử lý ngôn ngữ khác, chẳng hạn như trình dịch ngôn ngữ tự nhiên, sẽ có các thành phần bổ sung như từ điển và mã mẫu là văn bản đầu vào được dịch sang ngôn ngữ khác.

Có những mẫu kiến trúc thay thế có thể được sử dụng trong hệ thống xử lý ngôn ngữ (Garlan và Shaw, 1993). Các trình biên dịch có thể được triển khai bằng cách sử dụng hỗn hợp của một kho lưu trữ và một mô hình ống và bộ lọc. Trong kiến trúc trình biên dịch, bảng ảo là một kho lưu trữ các dữ liệu được chia sẻ. Các giai đoạn của từ vựng, cú pháp và phân tích ngữ nghĩa được tổ chức tuần tự như trong hình 6.19 và thông qua một bảng ký hiệu dùng chung.

Mô hình biên dịch ngôn ngữ dạng ống và bộ lọc này hiệu quả trong môi trường hàng loạt nơi các chương trình được biên dịch và thực thi mà không có sự tương tác của người dùng. Ví dụ, trong bản dịch tài liệu XML này sang tài liệu XML khác. Nó kém hiệu quả hơn khi một trình biên dịch được tích hợp với các công cụ xử lý ngôn ngữ khác như hệ thống soạn thảo có cấu trúc, trình gỡ lỗi tương tác hoặc trình duyệt chương trình. Trong nội dung này, những thay đổi từ một thành phần cần được phản ánh ngay lập tức trong các thành phần. Do đó, tốt hơn là tổ chức hệ thống xung quanh một kho lưu trữ, như được thể hiện trong Hình 6.20.

Hình này minh họa cách hệ thống xử lý ngôn ngữ có thể là một phần của bộ công cụ hỗ trợ lập trình. Trong ví dụ này, bảng ký hiệu và cây cú pháp hoạt động như một kho thông tin trung tâm. Công cụ hoặc các mảnh công cụ giao tiếp thông qua nó. Thông tin khác đôi khi được nhúng trong các công cụ, chẳng hạn như định nghĩa ngữ pháp và định nghĩa của định dạng đầu ra cho chương trình, đã được đưa ra khỏi các công cụ và đưa vào kho lưu trữ. Do đó, một trình soạn thảo hướng theo cú pháp có thể kiểm tra rằng cú pháp của một chương trình chính xác như nó đang được nhập và một người đánh giá khá có thể tạo danh sách chương trình ở định dạng dễ đọc.



*Hình 6.20: Kiến trúc lưu trữ cho một hệ thống xử lý ngôn ngữ.*