**CHƯƠNG 7**  
**CHUYỂN SANG THIẾT KẾ**

Phát triển hệ thống hướng đối tượng sử dụng các yêu cầu thu thập trong quá trình phân tích để tạo ra một bản thiết kế cho hệ thống tương lai. Một thiết kế hướng đối tượng thành công dựa trên những gì đã học được trong các giai đoạn trước đó và dẫn đến việc triển khai suôn sẻ bằng cách tạo ra một kế hoạch rõ ràng, chính xác về những gì cần làm. Chương này mô tả sự chuyển đổi ban đầu từ phân tích sang thiết kế và giới thiệu ba cách tiếp cận để thiết kế hệ thống mới.

**MỤC TIÊU**

* Hiểu quá trình xác minh và xác thực các mô hình phân tích.
* Hiểu quá trình chuyển đổi từ phân tích sang thiết kế.
* Hiểu việc sử dụng phân lớp, phân đoạn và lớp.
* Có thể tạo sơ đồ gói (package diagrams).
* Quen thuộc với các phương án thiết kế: tùy chỉnh, đóng gói và thuê ngoài.
* Có thể tạo ma trận các phương án thay thế.

**GIỚI THIỆU**

Mục đích của phân tích là xác định nhu cầu của doanh nghiệp. Mục đích của thiết kế là quyết định cách xây dựng hệ thống. Hoạt động chính diễn ra trong quá trình thiết kế là phát triển tập hợp các biểu diễn từ phân tích thành các biểu diễn thiết kế.

Trong suốt quá trình thiết kế, nhóm dự án phải xem xét kỹ hệ thống mới liên quan đến môi trường hiện tại và các hệ thống đang tồn tại trong tổ chức nói chung. Các yếu tố quan trọng cần cân nhắc để xác định hệ thống sẽ hoạt động như thế nào bao gồm các yếu tố môi trường, chẳng hạn như tích hợp với các hệ thống hiện có, chuyển đổi dữ liệu từ các hệ thống cũ và tận dụng các kỹ năng sẵn có trong tổ chức. Mặc dù việc lập kế hoạch và phân tích được thực hiện để phát triển một hệ thống tiềm năng, nhưng mục tiêu của thiết kế là tạo ra một bản thiết kế cho một hệ thống có thể được triển khai.

Một phần quan trọng ban đầu của thiết kế là xem xét một số chiến lược thiết kế và quyết định sẽ sử dụng phương án nào để xây dựng hệ thống. Hệ thống có thể được xây dựng từ đầu, mua sẵn và tùy chỉnh, hoặc thuê ngoài cho người khác, và nhóm dự án cần điều tra tính khả thi của từng phương án thay thế. Quyết định này ảnh hưởng đến các nhiệm vụ sẽ được thực hiện trong quá trình thiết kế.

Đồng thời, thiết kế chi tiết các lớp và phương thức riêng lẻ được sử dụng để vạch ra các chi tiết kỹ thuật của hệ thống và cách lưu trữ chúng vẫn cần được hoàn thành. Các kỹ thuật như thẻ CRC, sơ đồ lớp, hợp đồng đặc tả, đặc tả phương thức và thiết kế cơ sở dữ liệu cung cấp các chi tiết thiết kế cuối cùng để chuẩn bị cho giai đoạn triển khai, và chúng đảm bảo rằng các lập trình viên có đủ thông tin để xây dựng hệ thống đúng cách và hiệu quả. Các chủ đề này được trình bày trong Chương 8 và 9.

Thiết kế cũng bao gồm các hoạt động như thiết kế giao diện người dùng, đầu vào hệ thống và đầu ra hệ thống, liên quan đến cách người dùng tương tác với hệ thống. Chương 10 mô tả chi tiết ba hoạt động này cùng với các kỹ thuật như tạo kịch bản (storyboarding) và tạo mẫu (prototyping), giúp nhóm dự án thiết kế một hệ thống đáp ứng nhu cầu của người dùng và mang lại sự hài lòng khi sử dụng.

Cuối cùng, các quyết định về kiến trúc vật lý liên quan đến phần cứng và phần mềm sẽ được mua để hỗ trợ hệ thống mới và cách tổ chức quá trình xử lý của hệ thống. Ví dụ, hệ thống có thể được tổ chức sao cho quá trình xử lý của nó tập trung tại một địa điểm, phân tán hoặc cả tập trung và phân tán, và mỗi giải pháp đều mang lại những lợi ích và thách thức riêng cho nhóm dự án. Bởi vì các vấn đề toàn cầu và bảo mật ảnh hưởng đến các kế hoạch triển khai được thực hiện, chúng cần được xem xét cùng với kiến trúc kỹ thuật của hệ thống. Kiến trúc vật lý, bảo mật và các vấn đề toàn cầu được mô tả trong Chương 11.

Các bước thiết kế có mối liên hệ chặt chẽ với nhau và, giống như các bước trong phân tích, các nhà phân tích thường phải quay lại thực hiện các bước khác nhau. Ví dụ, tạo mẫu trong bước thiết kế giao diện thường tiết lộ thêm thông tin cần thiết cho hệ thống. Ngoài ra, một hệ thống được thiết kế cho một tổ chức có hệ thống tập trung có thể yêu cầu đầu tư đáng kể vào phần cứng và phần mềm nếu nhóm dự án quyết định chuyển sang một hệ thống trong đó tất cả các quá trình xử lý được phân tán.

**MẸO THỰC TẾ**  
**Tránh những lỗi thiết kế kinh điển**

Trong Chương 2, chúng ta đã thảo luận về một số lỗi kinh điển và cách tránh chúng. Dưới đây là tóm tắt bốn lỗi kinh điển trong thiết kế và cách để tránh chúng.

1. **Giảm thời gian thiết kế**: Nếu thời gian bị hạn chế, dễ có xu hướng giảm thời gian dành cho các hoạt động “không hiệu quả” như thiết kế để nhóm có thể nhảy vào lập trình “hiệu quả”. Điều này dẫn đến việc bỏ sót các chi tiết quan trọng mà sau đó phải điều tra với chi phí và thời gian cao hơn nhiều (thường gấp ít nhất mười lần). **Giải pháp**: Nếu áp lực thời gian lớn, hãy sử dụng phương pháp chia khung thời gian (timeboxing) để loại bỏ các tính năng hoặc chuyển chúng sang các phiên bản tương lai.
2. **Lạm phát tính năng**: Ngay cả khi bạn thành công trong việc tránh mở rộng phạm vi, khoảng 25% yêu cầu hệ thống vẫn sẽ thay đổi. Những thay đổi, dù lớn hay nhỏ, đều có thể làm tăng đáng kể thời gian và chi phí. **Giải pháp**: Đảm bảo rằng mọi thay đổi đều là cần thiết và người dùng nhận thức được tác động về chi phí và thời gian. Cố gắng chuyển các thay đổi được đề xuất vào các phiên bản tương lai.
3. **Hội chứng đạn bạc**: Các nhà phân tích đôi khi tin vào các tuyên bố tiếp thị về một số công cụ thiết kế tuyên bố giải quyết mọi vấn đề và giảm thời gian, chi phí một cách thần kỳ. Không có công cụ hay kỹ thuật nào có thể giảm tổng thời gian hay chi phí hơn 25% (mặc dù một số có thể giảm từng bước riêng lẻ tới mức này). **Giải pháp**: Nếu một công cụ thiết kế có tuyên bố nghe quá tốt để là sự thật, hãy từ chối sử dụng.
4. **Thay đổi công cụ giữa dự án**: Đôi khi các nhà phân tích chuyển sang một công cụ khác có vẻ tốt hơn trong quá trình thiết kế với hy vọng tiết kiệm thời gian hoặc chi phí. Thường thì, bất kỳ lợi ích nào đều bị lấn át bởi nhu cầu học công cụ mới. Điều này cũng áp dụng cho các nâng cấp nhỏ của công cụ hiện tại. **Giải pháp**: Đừng thay đổi hoặc nâng cấp trừ khi có nhu cầu cấp thiết về các tính năng cụ thể trong công cụ mới, và khi đó cần tăng thời gian dự kiến để bao gồm thời gian học tập.

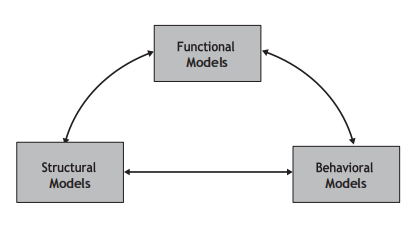
**XÁC MINH VÀ XÁC THỰC CÁC MÔ HÌNH PHÂN TÍCH**

Trước khi chuyển các biểu diễn phân tích thành biểu diễn thiết kế, chúng ta cần xác minh và xác thực bộ mô hình phân tích hiện tại để đảm bảo chúng phản ánh chính xác miền vấn đề đang được xem xét. Điều này bao gồm việc kiểm tra độ trung thực của từng mô hình; ví dụ, chúng ta phải chắc chắn rằng các sơ đồ hoạt động, mô tả trường hợp sử dụng, và sơ đồ trường hợp sử dụng đều mô tả các yêu cầu chức năng giống nhau. Nó cũng liên quan đến việc kiểm tra độ trung thực giữa các mô hình; chẳng hạn, các chuyển đổi trên máy trạng thái hành vi phải được liên kết với các thao tác trong sơ đồ lớp. Trong các Chương 4, 5, và 6, chúng ta đã tập trung vào việc xác minh và xác thực các mô hình cá nhân: chức năng, cấu trúc và hành vi. Trong chương này, chúng ta tập trung vào việc đảm bảo rằng các mô hình khác nhau nhất quán với nhau. Hình 7-1 minh họa rằng các mô hình phân tích hướng đối tượng có mối quan hệ chặt chẽ với nhau. Ví dụ, các mô hình chức năng và cấu trúc có đồng nhất không? Còn các mô hình chức năng và hành vi thì sao? Và cuối cùng, các mô hình cấu trúc và hành vi có đáng tin cậy không? Trong phần này, chúng ta mô tả một tập hợp các quy tắc hữu ích để xác minh và xác thực các điểm giao nhau của các mô hình phân tích. Tùy thuộc vào các cấu trúc cụ thể của từng mô hình, các mối quan hệ khác nhau sẽ liên quan. Quá trình đảm bảo tính nhất quán giữa chúng được gọi là cân bằng các mô hình.

**Cân bằng các mô hình chức năng và cấu trúc**

Để cân bằng các mô hình chức năng và cấu trúc, chúng ta phải đảm bảo rằng hai tập hợp mô hình này nhất quán với nhau. Điều đó có nghĩa là các sơ đồ hoạt động, mô tả trường hợp sử dụng và sơ đồ trường hợp sử dụng phải đồng nhất với các thẻ CRC và sơ đồ lớp thể hiện mô hình đang phát triển của miền vấn đề. Hình 7-2 cho thấy mối quan hệ giữa các mô hình chức năng và cấu trúc. Bằng cách xem xét hình này, chúng ta phát hiện ra bốn tập hợp liên kết giữa các mô hình. Điều này cho chúng ta điểm bắt đầu để cân bằng các mô hình chức năng và cấu trúc.

1. **Mỗi lớp trong sơ đồ lớp và mỗi thẻ CRC phải liên quan đến ít nhất một trường hợp sử dụng và ngược lại**. Ví dụ, thẻ CRC được minh họa trong Hình 7-3 và lớp liên quan của nó trong sơ đồ lớp (xem Hình 7-4) được liên kết với trường hợp sử dụng Đặt lịch hẹn cho bệnh nhân cũ (Make Old Patient Appt) được mô tả trong Hình 7-5.
2. **Mỗi hoạt động hoặc hành động trong sơ đồ hoạt động và mỗi sự kiện trong mô tả trường hợp sử dụng nên liên quan đến một hoặc nhiều trách nhiệm trên thẻ CRC và một hoặc nhiều thao tác trong một lớp trong sơ đồ lớp và ngược lại**. Ví dụ, hoạt động Lấy thông tin bệnh nhân trong sơ đồ hoạt động (xem Hình 7-6) và hai sự kiện đầu tiên trong mô tả trường hợp sử dụng (xem Hình 7-5) liên quan đến trách nhiệm đặt lịch hẹn trên thẻ CRC (xem Hình 7-3) và thao tác makeAppointment() trong lớp Bệnh nhân (Patient) trên sơ đồ lớp (xem Hình 7-4).
3. **Mỗi nút đối tượng trong sơ đồ hoạt động phải liên quan đến một thực thể của một lớp trong sơ đồ lớp (tức là một đối tượng) và một thẻ CRC hoặc một thuộc tính được chứa trong một lớp và trên một thẻ CRC**. Tuy nhiên, trong Hình 7-6, có một nút đối tượng, Appt Request Info, dường như không liên quan đến bất kỳ lớp nào trong sơ đồ lớp được mô tả trong Hình 7-4. Vì vậy, hoặc sơ đồ hoạt động hoặc sơ đồ lớp có lỗi hoặc nút đối tượng phải đại diện cho một thuộc tính. Trong trường hợp này, nó dường như không đại diện cho một thuộc tính. Chúng ta có thể thêm một lớp vào sơ đồ lớp để tạo các đối tượng tạm thời liên quan đến nút đối tượng trong sơ đồ hoạt động. Tuy nhiên, không rõ có thao tác nào sẽ liên quan đến các đối tượng tạm thời này hay không. Do đó, một giải pháp tốt hơn là xóa các nút đối tượng Appt Request Info khỏi sơ đồ hoạt động. Thực tế, nút đối tượng này chỉ đại diện cho một tập hợp các giá trị thuộc tính được đóng gói, tức là dữ liệu sẽ được sử dụng trong quy trình hệ thống đặt lịch hẹn (xem Hình 7-7).



**Hình 7-1: Các Mô Hình Phân Tích Hướng Đối Tượng**

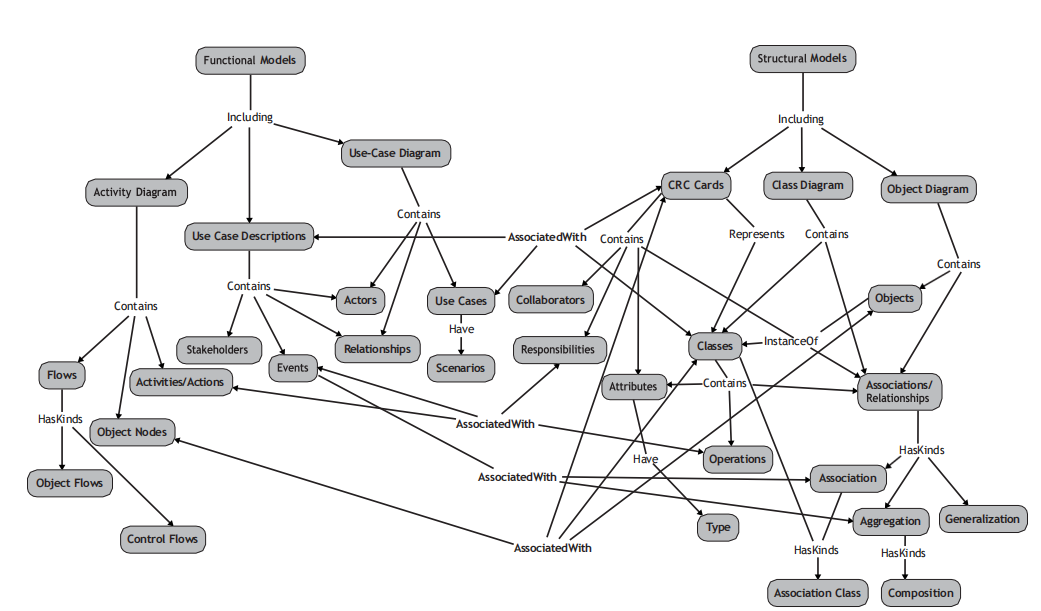
1. **Mỗi thuộc tính và mối quan hệ/liên kết/tập hợp chứa trong thẻ CRC (và được kết nối với một lớp trong sơ đồ lớp) nên liên quan đến đối tượng hoặc chủ thể của một sự kiện trong mô tả trường hợp sử dụng**. Ví dụ, trong Hình 7-5, sự kiện thứ hai nêu: Bệnh nhân cung cấp cho nhân viên lễ tân tên và địa chỉ của mình. Bằng cách xem xét thẻ CRC trong Hình 7-3 và sơ đồ lớp trong Hình 7-4, chúng ta thấy rằng lớp Bệnh nhân là một lớp con của lớp Người tham gia (Participant) và do đó kế thừa tất cả các thuộc tính, liên kết và thao tác được định nghĩa trong lớp Người tham gia, nơi các thuộc tính tên và địa chỉ được định nghĩa.

**Cân Bằng Các Mô Hình Chức Năng và Hành Vi**

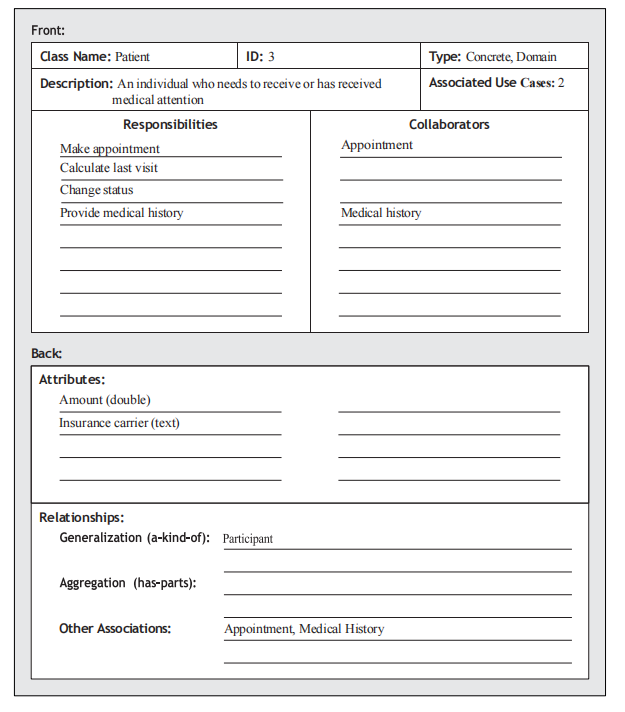
Tương tự như việc cân bằng các mô hình chức năng và cấu trúc, chúng ta cần đảm bảo tính nhất quán của hai tập hợp mô hình này. Trong trường hợp này, các biểu đồ hoạt động, mô tả trường hợp sử dụng (use-case), và biểu đồ trường hợp sử dụng phải phù hợp với các biểu đồ tuần tự, biểu đồ giao tiếp, máy trạng thái hành vi và ma trận CRUDE. Hình 7-8 thể hiện mối quan hệ giữa các mô hình chức năng và hành vi. Dựa trên các mối quan hệ này, chúng ta thấy có bốn lĩnh vực cần quan tâm.

Thứ nhất, các biểu đồ tuần tự và giao tiếp phải được liên kết với một trường hợp sử dụng trên biểu đồ trường hợp sử dụng và một mô tả trường hợp sử dụng. Ví dụ, biểu đồ tuần tự trong Hình 7-9 và biểu đồ giao tiếp trong Hình 7-10 liên quan đến các kịch bản của trường hợp sử dụng "Đặt Lịch Hẹn Cho Bệnh Nhân Cũ" xuất hiện trong mô tả trường hợp sử dụng ở Hình 7-5 và biểu đồ trường hợp sử dụng ở Hình 7-11.

Thứ hai, các tác nhân trong biểu đồ tuần tự, biểu đồ giao tiếp, và/hoặc ma trận CRUDE phải liên kết với các tác nhân trong biểu đồ trường hợp sử dụng hoặc được nhắc đến trong mô tả trường hợp sử dụng, và ngược lại. Ví dụ, tác nhân "aPatient" trong biểu đồ tuần tự ở Hình 7-9, biểu đồ giao tiếp ở Hình 7-10, và hàng và cột "Patient" trong ma trận CRUDE ở Hình 7-12 xuất hiện trong biểu đồ trường hợp sử dụng ở Hình 7-11 và mô tả trường hợp sử dụng ở Hình 7-5. Tuy nhiên, "aReceptionist" không xuất hiện trong biểu đồ trường hợp sử dụng nhưng được nhắc đến trong các sự kiện liên quan đến mô tả trường hợp sử dụng "Đặt Lịch Hẹn Cho Bệnh Nhân Cũ". Trong trường hợp này, tác nhân "aReceptionist" rõ ràng là một tác nhân nội bộ, không thể hiện trên biểu đồ trường hợp sử dụng của UML.



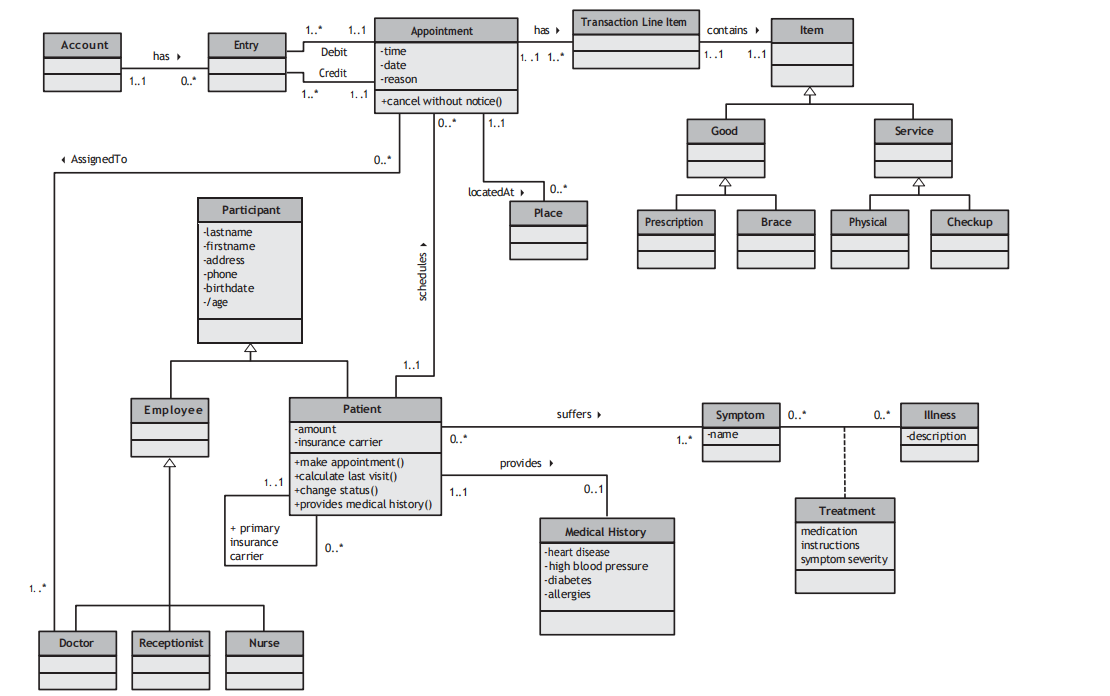
**HÌNH 7-2: Các Mối Quan Hệ Giữa Mô Hình Chức Năng và Cấu Trúc**

****

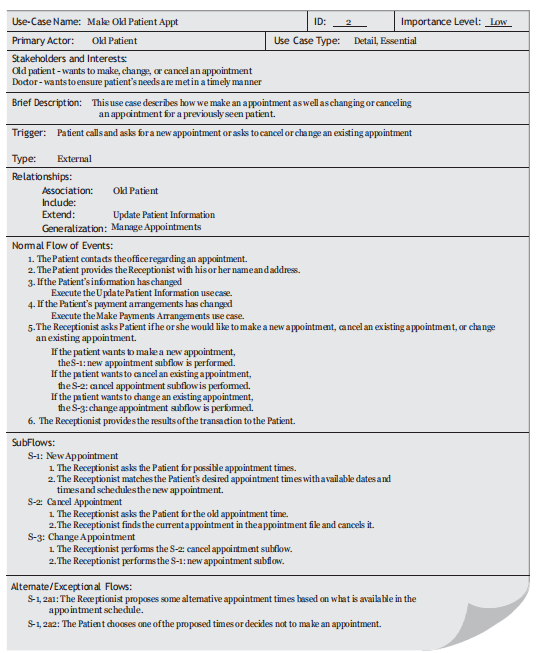
**HÌNH 7-3: Thẻ CRC Bệnh Nhân Cũ (Hình 5-25)**

Thứ ba, các thông điệp trên sơ đồ tuần tự và sơ đồ truyền thông, các chuyển đổi trên các máy trạng thái hành vi, và các mục nhập trong ma trận CRUDE phải liên quan đến các hoạt động và hành động trên sơ đồ hoạt động và các sự kiện được liệt kê trong mô tả trường hợp sử dụng, và ngược lại. Ví dụ, thông điệp **CreateAppt()** trên sơ đồ tuần tự và sơ đồ truyền thông (xem Hình 7-9 và 7-10) có liên quan đến hoạt động **CreateAppointment** (xem Hình 7-7) và luồng phụ **S-1: New Appointment** trong mô tả trường hợp sử dụng (xem Hình 7-5). Mục **C** trong ô **Receptionist Appointment** của ma trận CRUDE cũng liên quan đến các thông điệp, hoạt động và luồng phụ này.

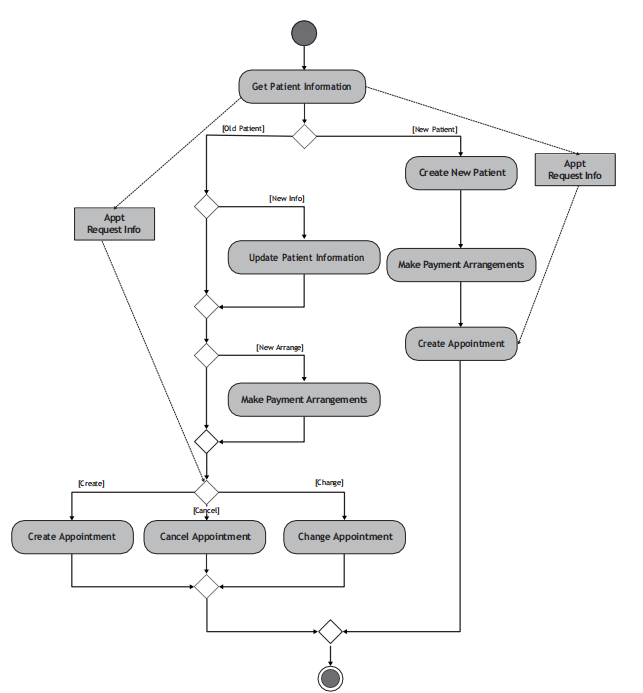
Thứ tư, tất cả các đối tượng phức tạp được biểu diễn bằng một nút đối tượng trong sơ đồ hoạt động phải có một máy trạng thái hành vi đại diện cho vòng đời của đối tượng đó, và ngược lại. Như đã đề cập trong Chương 6, các đối tượng phức tạp thường rất năng động và trải qua nhiều trạng thái khác nhau trong suốt vòng đời của chúng. Tuy nhiên, trong trường hợp này, vì chúng ta không còn bất kỳ nút đối tượng nào trong sơ đồ hoạt động (xem Hình 7-7), nên không cần thiết phải tạo một máy trạng thái hành vi dựa trên sơ đồ hoạt động.



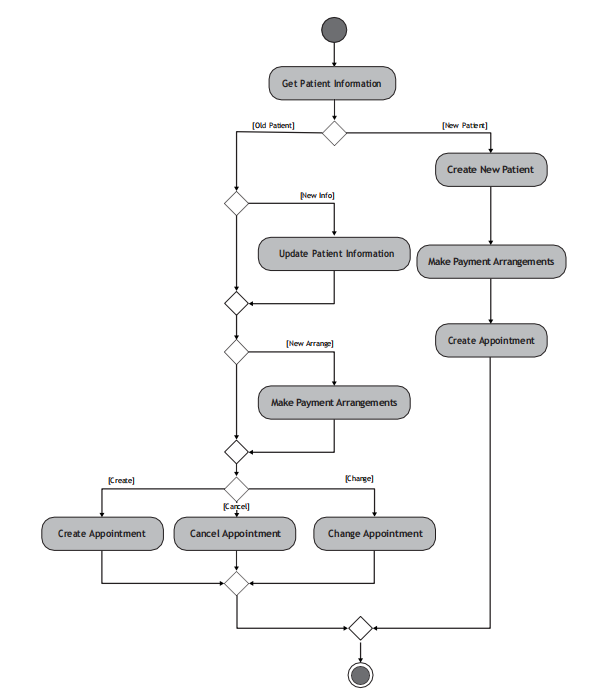
**HÌNH 7-4 Sơ đồ lớp của Vấn đề Cuộc hẹn (Hình 5-7)**

****

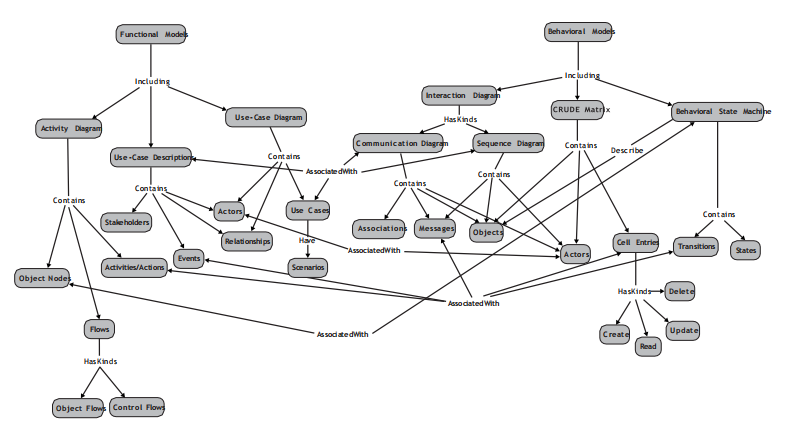
**HÌNH 7-5 Mô tả trường hợp sử dụng cho trường hợp sử dụng Đặt cuộc hẹn cho Bệnh nhân Cũ (Hình 4-13)**

****

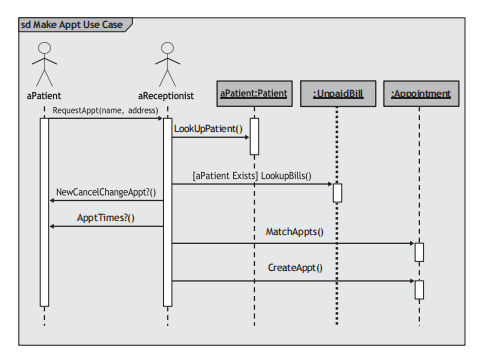
**HÌNH 7-6 Sơ đồ hoạt động cho trường hợp sử dụng Quản lý cuộc hẹn (Hình 4-8)**

****

**HÌNH 7-7 Sơ đồ hoạt động đã chỉnh sửa cho trường hợp sử dụng Quản lý cuộc hẹn**

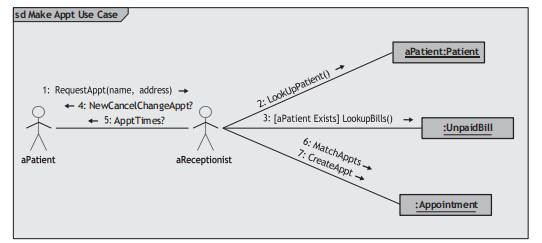
****

**HÌNH 7-8 Mối quan hệ giữa các Mô hình Chức năng và Hành vi**

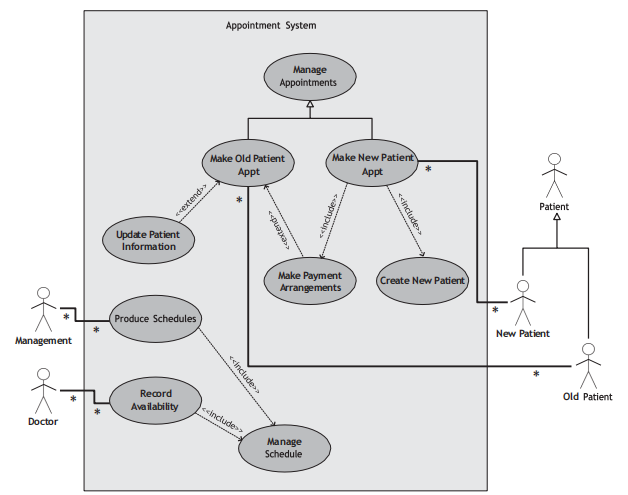
****

**HÌNH 7-9 Sơ đồ tuần tự cho một kịch bản của trường hợp sử dụng Đặt cuộc hẹn cho Bệnh nhân Cũ (Hình 6-1)**

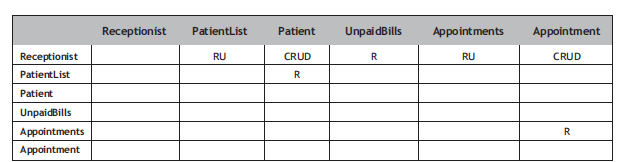
**Cân bằng Mô hình Cấu trúc và Hành vi**  
Để khám phá các mối quan hệ giữa các mô hình cấu trúc và hành vi, chúng ta sử dụng bản đồ khái niệm trong **Hình 7-13**. Trong trường hợp này, có năm khu vực mà chúng ta phải đảm bảo tính nhất quán giữa các mô hình.



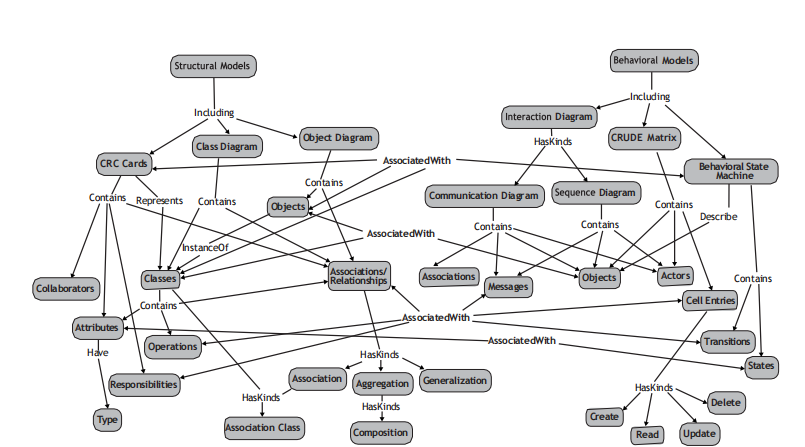
**HÌNH 7-10 Biểu đồ Giao tiếp cho Kịch bản của Ca Sử dụng Đặt Lịch cho Bệnh nhân Cũ (Hình 6-10)**

****

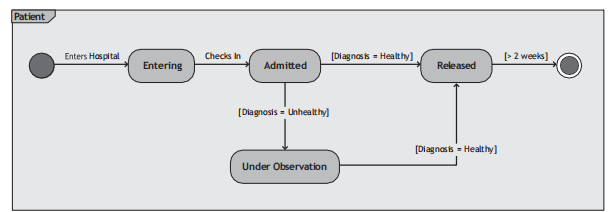
**HÌNH 7-11** **Biểu đồ Ca Sử dụng đã Chỉnh sửa cho Hệ thống Đặt Lịch (Hình 4-21)**

****

**HÌNH 7-12 Ma trận CRUDE cho Ca Sử dụng Đặt Lịch cho Bệnh nhân Cũ (Hình 6-23)**

****

**HÌNH 7-13** **Mối quan hệ giữa các Mô hình Cấu trúc và Hành vi**

****

**HÌNH 7-14** **Máy Trạng thái Hành vi cho Bệnh nhân tại Bệnh viện (Hình 6-16)**

**Thứ nhất**, các đối tượng xuất hiện trong ma trận CRUDE phải được liên kết với các lớp được đại diện bởi thẻ CRC và xuất hiện trên sơ đồ lớp, và ngược lại. Ví dụ, lớp **Patient** trong ma trận CRUDE ở **Hình 7-12** được liên kết với thẻ CRC trong **Hình 7-3** và lớp **Patient** trong sơ đồ lớp ở **Hình 7-4**.

**Thứ hai**, vì máy trạng thái hành vi thể hiện vòng đời của các đối tượng phức tạp, chúng phải được liên kết với các phiên bản (đối tượng) của các lớp trên sơ đồ lớp và với một thẻ CRC đại diện cho lớp của phiên bản đó. Ví dụ, máy trạng thái hành vi mô tả một phiên bản của lớp **Patient** trong **Hình 7-14** ngụ ý rằng một lớp **Patient** tồn tại trên sơ đồ lớp liên quan (xem **Hình 7-4**) và rằng một thẻ CRC tồn tại cho lớp liên quan (xem **Hình 7-3**).

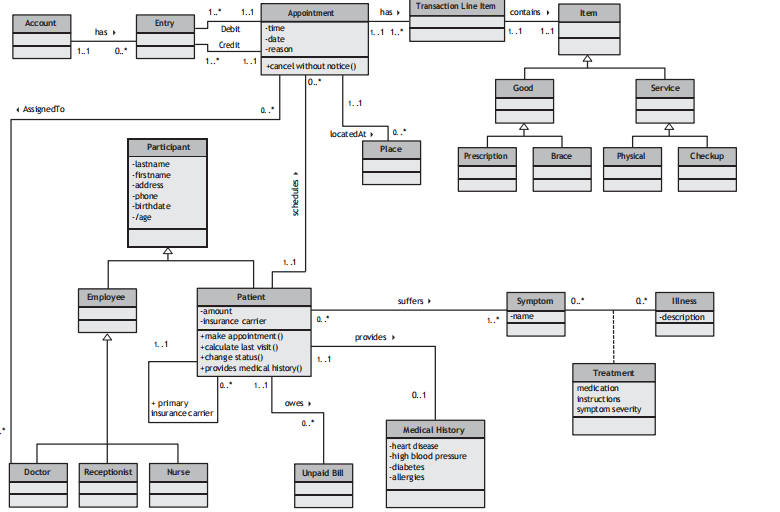
**Thứ ba**, các biểu đồ giao tiếp và biểu đồ tuần tự chứa các đối tượng phải là một phiên bản của một lớp được đại diện bởi thẻ CRC và được đặt trên sơ đồ lớp. Ví dụ, **Hình 7-9** và **Hình 7-10** có một đối tượng **anAppt** là một phiên bản của lớp **Appointment**. Do đó, lớp **Appointment** phải tồn tại trong sơ đồ lớp (xem **Hình 7-4**), và một thẻ CRC nên tồn tại để mô tả nó. Tuy nhiên, có một đối tượng trên biểu đồ giao tiếp và biểu đồ tuần tự liên kết với một lớp không tồn tại trên sơ đồ lớp: **UnpaidBill**. Tại thời điểm này, nhà phân tích phải quyết định hoặc sửa đổi sơ đồ lớp bằng cách thêm các lớp này hoặc xem xét lại các biểu đồ giao tiếp và biểu đồ tuần tự. Trong trường hợp này, tốt hơn là thêm lớp vào sơ đồ lớp (xem **Hình 7-15**).

**Thứ tư**, các thông điệp chứa trên biểu đồ tuần tự và biểu đồ giao tiếp, các chuyển tiếp trên máy trạng thái hành vi, và các mục ô trong ma trận CRUDE phải được liên kết với các trách nhiệm và mối liên hệ trên các thẻ CRC và các hoạt động trong các lớp và mối liên hệ liên kết với các lớp trên sơ đồ lớp. Ví dụ, thông điệp **CreateAppt()** trên biểu đồ tuần tự và biểu đồ giao tiếp (xem **Hình 7-9** và **Hình 7-10**) liên quan đến hoạt động **makeAppointment** của lớp **Patient** và mối liên hệ **schedules** giữa các lớp **Patient** và **Appointment** trên sơ đồ lớp (xem **Hình 7-15**).

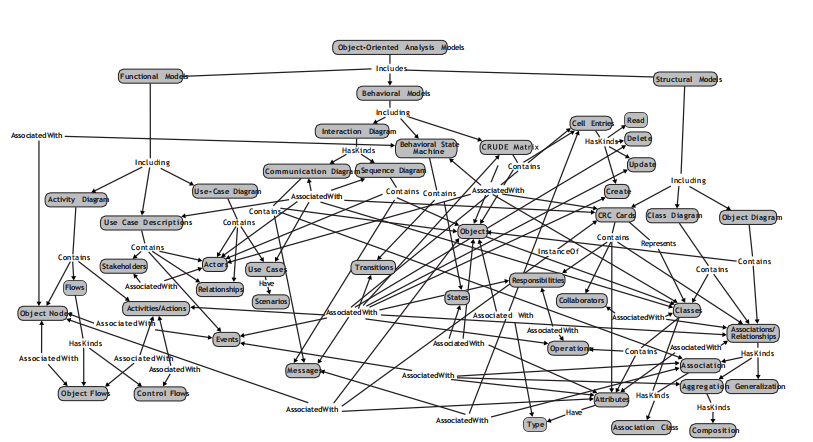
**Thứ năm**, các trạng thái trong một máy trạng thái hành vi phải được liên kết với các giá trị khác nhau của một thuộc tính hoặc tập hợp các thuộc tính mô tả một đối tượng. Ví dụ, máy trạng thái hành vi cho đối tượng bệnh nhân tại bệnh viện ngụ ý rằng nên có một thuộc tính, có thể là **trạng thái hiện tại**, cần được bao gồm trong định nghĩa của lớp.

**Tóm tắt**

**Hình 7-16** mô tả một bản đồ khái niệm thể hiện toàn bộ bức tranh về mối quan hệ giữa các biểu đồ được đề cập trong phần này. Rõ ràng từ sự phức tạp của hình này rằng cân bằng tất cả các mô hình chức năng, cấu trúc và hành vi là một nhiệm vụ rất tốn thời gian, tỉ mỉ và khó khăn. Tuy nhiên, nếu không chú ý đến mức độ này cho các mô hình đang phát triển thể hiện hệ thống, các mô hình sẽ không cung cấp một nền tảng vững chắc để thiết kế và xây dựng hệ thống.

****

**HÌNH 7-15 Sơ đồ lớp Hệ thống Lịch hẹn đã được sửa chữa**

****

**HÌNH 7-16 Mối quan hệ giữa các Mô hình Phân tích Đối tượng**

**PHÁT TRIỂN CÁC MÔ HÌNH PHÂN TÍCH THÀNH CÁC MÔ HÌNH THIẾT KẾ**

Bây giờ chúng ta đã xác minh và xác thực thành công các mô hình phân tích, chúng ta cần bắt đầu phát triển chúng thành các mô hình thiết kế phù hợp. Mục đích của các mô hình phân tích là để đại diện cho vấn đề trong lĩnh vực kinh doanh như một tập hợp các đối tượng cộng tác. Nói cách khác, các hoạt động phân tích đã xác định các yêu cầu chức năng. Để đạt được điều này, các hoạt động phân tích đã bỏ qua các yêu cầu phi chức năng như hiệu suất và các vấn đề môi trường hệ thống (ví dụ: xử lý phân tán hay tập trung, các vấn đề giao diện người dùng và cơ sở dữ liệu). Ngược lại, mục đích chính của các mô hình thiết kế là tăng khả năng cung cấp thành công một hệ thống thực hiện các yêu cầu chức năng theo cách tiết kiệm và dễ bảo trì. Do đó, trong thiết kế hệ thống, chúng ta phải giải quyết cả các yêu cầu chức năng và phi chức năng.

Từ góc độ hướng đối tượng, các mô hình thiết kế hệ thống đơn giản là làm tinh chỉnh các mô hình phân tích hệ thống bằng cách thêm các chi tiết môi trường hệ thống (hoặc miền giải pháp) vào chúng và tinh chỉnh thông tin miền vấn đề đã có trong các mô hình phân tích. Khi phát triển mô hình phân tích thành mô hình thiết kế, bạn nên xem xét cẩn thận các trường hợp sử dụng và tập hợp các lớp hiện tại (các thao tác và thuộc tính của chúng cùng với các mối quan hệ giữa chúng). Tất cả các lớp có cần thiết không? Có lớp nào thiếu không? Các lớp đã được định nghĩa đầy đủ chưa? Có thuộc tính hoặc phương thức nào bị thiếu không? Các lớp có thuộc tính và phương thức không cần thiết không? Cách thể hiện hiện tại của hệ thống đang phát triển có tối ưu không? Rõ ràng, nếu chúng ta đã xác minh và xác thực các mô hình phân tích, thì đã có một phần của quá trình này diễn ra. Tuy nhiên, phát triển hệ thống hướng đối tượng là cả một quá trình gia tăng và lặp lại. Do đó, chúng ta cần xem xét lại các mô hình phân tích một lần nữa. Tuy nhiên, lần này chúng ta bắt đầu nhìn vào các mô hình của miền vấn đề thông qua một cái nhìn thiết kế. Trong bước này, chúng ta thực hiện các sửa đổi đối với các mô hình miền vấn đề nhằm nâng cao hiệu quả và hiệu suất của hệ thống đang phát triển.

Trong các phần tiếp theo, chúng tôi sẽ giới thiệu việc phân tích, phân vùng và hợp tác, và các lớp như một cách để phát triển các mô hình phân tích định hướng miền vấn đề thành các mô hình thiết kế định hướng miền giải pháp tối ưu. Từ góc độ Enhanced Unified Process (xem Hình 1-16), chúng ta đang chuyển từ quy trình phân tích sang quy trình thiết kế, và chúng ta đang tiến sâu hơn vào giai đoạn Elaboration và một phần vào giai đoạn Construction.

**Phân tích**

Phân tích là quá trình tách một module thành một module độc lập. Module mới có thể là một lớp mới hoặc một phương thức mới. Ví dụ, khi xem xét một tập hợp các lớp, có thể phát hiện rằng chúng có một tập hợp các thuộc tính và phương thức tương tự. Do đó, có thể hợp lý để tách các điểm tương đồng này thành một lớp riêng biệt. Tùy thuộc vào việc lớp mới có nên ở trong mối quan hệ lớp cha với các lớp hiện có hay không, lớp mới có thể được liên kết với các lớp hiện có thông qua một mối quan hệ tổng quát (một loại) hoặc có thể thông qua một mối quan hệ tổng hợp (có các phần). Sử dụng ví dụ về hệ thống lịch hẹn, nếu lớp Nhân viên chưa được xác định, chúng ta có thể xác định nó ở giai đoạn này bằng cách phân tích các phương thức và thuộc tính tương tự từ các lớp Y tá, Lễ tân và Bác sĩ. Trong trường hợp này, chúng ta sẽ liên kết lớp mới (Nhân viên) với các lớp hiện có bằng mối quan hệ tổng quát (một loại). Rõ ràng, bằng cách mở rộng, chúng ta cũng có thể đã tạo ra lớp Tham gia nếu nó chưa được xác định trước đó.

Trừu tượng và tinh chỉnh là hai quá trình liên quan chặt chẽ đến phân tích. Trừu tượng liên quan đến việc tạo ra một ý tưởng cấp cao hơn từ một tập hợp các ý tưởng. Việc xác định lớp Nhân viên là một ví dụ về việc trừu tượng từ một tập hợp các lớp thấp hơn lên một lớp cao hơn. Trong một số trường hợp, quá trình trừu tượng xác định các lớp trừu tượng, trong khi trong những tình huống khác, nó xác định các lớp cụ thể bổ sung. Quy trình tinh chỉnh là quá trình ngược lại với quá trình trừu tượng. Trong ví dụ về hệ thống lịch hẹn, chúng ta có thể xác định các lớp con bổ sung của lớp Nhân viên, chẳng hạn như Thư ký và Kế toán. Tất nhiên, chúng ta chỉ thêm các lớp mới nếu có đủ sự khác biệt giữa chúng. Nếu không, lớp tổng quát hơn, Nhân viên, sẽ đủ.

**Phân vùng và Hợp tác**

Dựa trên tất cả các quá trình phân tích, tinh chỉnh và trừu tượng có thể diễn ra với hệ thống đang phát triển, kích thước khổng lồ của đại diện hệ thống có thể làm quá tải người dùng và nhà phát triển. Tại thời điểm này trong quá trình phát triển hệ thống, có thể hợp lý để chia nhỏ đại diện thành một tập hợp các phân vùng. Một phân vùng là tương đương với một hệ thống con trong hướng đối tượng, nơi một hệ thống con là sự phân rã của một hệ thống lớn hơn thành các hệ thống thành phần của nó (ví dụ: một hệ thống thông tin kế toán có thể được phân rã về mặt chức năng thành một hệ thống phải trả, một hệ thống phải thu, một hệ thống tiền lương, v.v.). Từ góc độ hướng đối tượng, các phân vùng dựa trên mẫu hoạt động (các thông điệp được gửi) giữa các đối tượng trong một hệ thống hướng đối tượng. Chúng tôi sẽ mô tả một phương pháp đơn giản để mô hình hóa các phân vùng và hợp tác trong phần sau của chương này: các gói và sơ đồ gói.

Một nơi tốt để tìm kiếm các phân vùng tiềm năng là các hợp tác được mô hình hóa trong các sơ đồ giao tiếp của UML (xem Chương 6). Nếu bạn nhớ, một cách hữu ích để xác định các hợp tác là tạo một sơ đồ giao tiếp cho mỗi trường hợp sử dụng. Tuy nhiên, vì một lớp riêng lẻ có thể hỗ trợ nhiều trường hợp sử dụng, nên một lớp riêng lẻ có thể tham gia vào nhiều hợp tác dựa trên trường hợp sử dụng. Trong các trường hợp mà các lớp hỗ trợ nhiều trường hợp sử dụng, các hợp tác nên được hợp nhất. Sơ đồ lớp nên được xem xét để xem các lớp khác nhau có liên quan đến nhau như thế nào. Ví dụ, nếu các thuộc tính của một lớp có các loại đối tượng phức tạp, chẳng hạn như Người, Địa chỉ hoặc Phòng ban, và các loại đối tượng này không được mô hình hóa như các mối quan hệ trong sơ đồ lớp, chúng ta cần nhận ra những mối quan hệ ngầm này. Việc tạo ra một sơ đồ kết hợp sơ đồ lớp với các sơ đồ giao tiếp có thể rất hữu ích để cho thấy mức độ mà các lớp gắn kết với nhau. Mức độ gắn kết giữa các lớp càng cao, thì khả năng các lớp nên được nhóm lại trong một hợp tác hoặc phân vùng càng lớn. Bằng cách xem xét một ma trận CRUDE, chúng ta có thể sử dụng phân tích CRUDE (xem Chương 6) để xác định các lớp tiềm năng mà trên đó chúng ta có thể hợp nhất các hợp tác.

Một trong những kỹ thuật dễ dàng nhất để xác định các lớp có thể được nhóm lại để hình thành một hợp tác là thông qua việc sử dụng phân tích cụm hoặc phân tích kích thước đa chiều. Những kỹ thuật thống kê này cho phép nhóm có thể khách quan nhóm các lớp lại với nhau dựa trên mối liên hệ của chúng với nhau. Mối liên hệ có thể dựa trên các mối quan hệ ngữ nghĩa, các loại thông điệp khác nhau được gửi giữa chúng (ví dụ: tạo, đọc, cập nhật, xóa hoặc thực hiện), hoặc một sự kết hợp có trọng số của cả hai. Có rất nhiều biện pháp tương đồng khác nhau và nhiều thuật toán khác nhau mà các cụm có thể dựa vào, vì vậy cần phải cẩn thận khi sử dụng những kỹ thuật này. Hãy luôn đảm bảo rằng các hợp tác được xác định bằng các kỹ thuật này **làm cho nó có ý nghĩa từ góc độ miền vấn đề.** Chỉ vì một thuật toán toán học gợi ý rằng các lớp thuộc về nhau thì không có nghĩa là như vậy. Tuy nhiên, đây là một cách tiếp cận tốt để tạo ra một tập hợp hợp tác ban đầu.

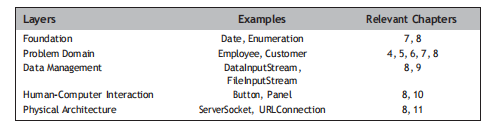
Tùy thuộc vào độ phức tạp của hợp tác đã hợp nhất, có thể hữu ích khi phân rã hợp tác thành nhiều phân vùng. Trong trường hợp này, ngoài việc có các hợp tác giữa các đối tượng, còn có thể có các hợp tác giữa các phân vùng. Quy tắc chung là càng nhiều thông điệp được gửi giữa các đối tượng, thì khả năng cao các đối tượng thuộc về cùng một phân vùng. Ngược lại, càng ít thông điệp được gửi, khả năng hai đối tượng không thuộc về nhau càng cao.

Một cách tiếp cận hữu ích khác để xác định các phân vùng tiềm năng là mô hình hóa mỗi hợp tác giữa các đối tượng theo dạng khách hàng, máy chủ và hợp đồng. Một khách hàng là một thể hiện của một lớp gửi một thông điệp đến một thể hiện của lớp khác để thực hiện một phương thức; một máy chủ là thể hiện của một lớp nhận thông điệp; và một hợp đồng là tài liệu xác định các tương tác giữa các đối tượng khách hàng và máy chủ (xem Chương 5 và 8). Cách tiếp cận này cho phép nhà phát triển xây dựng các phân vùng tiềm năng bằng cách xem xét các hợp đồng đã được xác định giữa các đối tượng. Trong trường hợp này, càng nhiều hợp đồng giữa các đối tượng, khả năng cao rằng các đối tượng thuộc về cùng một phân vùng. Càng ít hợp đồng, thì khả năng hai lớp thuộc về cùng một phân vùng càng thấp.

Hãy nhớ rằng, mục đích chính của việc xác định các hợp tác và phân vùng là để xác định các lớp nào nên được nhóm lại trong thiết kế.

**Các lớp**  
Cho đến thời điểm này trong quá trình phát triển hệ thống của chúng ta, chúng ta chỉ tập trung vào miền vấn đề; chúng ta hoàn toàn bỏ qua môi trường hệ thống (quản lý dữ liệu, giao diện người dùng và kiến trúc vật lý). Để thành công trong việc phát triển mô hình phân tích của hệ thống thành mô hình thiết kế của hệ thống, chúng ta phải thêm thông tin về môi trường hệ thống. Một cách hữu ích để làm điều này, mà không làm quá tải nhà phát triển, là sử dụng các lớp. Một lớp đại diện cho một phần tử của kiến trúc phần mềm của hệ thống đang phát triển. Chúng ta chỉ tập trung vào một lớp trong kiến trúc phần mềm đang phát triển: lớp miền vấn đề. Sẽ có một lớp cho mỗi yếu tố khác nhau của môi trường hệ thống (ví dụ: quản lý dữ liệu, giao diện người dùng, kiến trúc vật lý). Giống như các phân vùng và hợp tác, các lớp cũng có thể được mô tả bằng cách sử dụng các gói và sơ đồ gói (xem phần tiếp theo của chương này).

Ý tưởng phân tách các yếu tố khác nhau của kiến trúc thành các lớp riêng biệt có thể được truy nguyên đến kiến trúc MVC của Smalltalk. Khi Smalltalk được tạo ra lần đầu tiên, các tác giả quyết định tách biệt logic ứng dụng khỏi logic của giao diện người dùng. Theo cách này, có thể dễ dàng phát triển các giao diện người dùng khác nhau hoạt động với cùng một ứng dụng. Để thực hiện điều này, họ đã tạo ra kiến trúc Model–View–Controller (MVC), trong đó các Model thực hiện logic ứng dụng (miền vấn đề) và các View và Controller thực hiện logic cho giao diện người dùng. Các View xử lý đầu ra, và các Controller xử lý đầu vào. Vì giao diện người dùng đồ họa đầu tiên được phát triển trong ngôn ngữ Smalltalk, kiến trúc MVC đã phục vụ như một nền tảng cho hầu hết tất cả các giao diện người dùng đồ họa được phát triển ngày nay (bao gồm các giao diện Mac, các gia đình Windows và các môi trường GUI dựa trên Unix khác nhau).



**HÌNH 7-17 Các lớp và Các lớp mẫu**

Dựa trên kiến trúc MVC sáng tạo của Smalltalk, nhiều lớp phần mềm khác nhau đã được đề xuất. Chúng tôi gợi ý các lớp sau đây để xây dựng kiến trúc phần mềm: nền tảng, miền vấn đề, quản lý dữ liệu, tương tác giữa người và máy tính, và kiến trúc vật lý (xem Hình 7-17). Mỗi lớp giới hạn các loại lớp có thể tồn tại trên đó (ví dụ, chỉ các lớp giao diện người dùng có thể tồn tại trên lớp tương tác giữa người và máy tính).

**Nền tảng**

Lớp nền tảng, theo nhiều cách, là một lớp rất không thú vị. Nó chứa các lớp cần thiết cho bất kỳ ứng dụng hướng đối tượng nào tồn tại. Chúng bao gồm các lớp đại diện cho các kiểu dữ liệu cơ bản (ví dụ: số nguyên, số thực, ký tự, chuỗi), các lớp đại diện cho các cấu trúc dữ liệu cơ bản, đôi khi được gọi là các lớp chứa (ví dụ: danh sách, cây, đồ thị, tập hợp, ngăn xếp, hàng đợi), và các lớp đại diện cho các trừu tượng hữu ích, đôi khi được gọi là các lớp tiện ích (ví dụ: ngày, giờ, tiền). Những lớp này hiếm khi, nếu không muốn nói là không bao giờ, được sửa đổi bởi nhà phát triển. Chúng chỉ đơn giản được sử dụng. Ngày nay, các lớp có trên lớp này thường được bao gồm trong các môi trường phát triển hướng đối tượng.

**Miền Vấn đề**

Lớp miền vấn đề là những gì chúng tôi đã tập trung sự chú ý cho đến bây giờ. Ở giai đoạn này trong quá trình phát triển hệ thống của chúng tôi, chúng tôi cần chi tiết hơn các lớp để có thể triển khai chúng một cách hiệu quả và hiệu suất. Nhiều vấn đề cần được giải quyết khi thiết kế các lớp, bất kể chúng xuất hiện trên lớp nào. Ví dụ, có những vấn đề liên quan đến phân tách, tính gắn kết và tính liên kết, tính đồng nhất, đóng gói, sử dụng đúng đắn tính kế thừa và đa hình, ràng buộc, xác định hợp đồng, và thiết kế phương thức chi tiết. Những vấn đề này được thảo luận trong Chương 8.

**Quản lý Dữ liệu**

Lớp quản lý dữ liệu giải quyết các vấn đề liên quan đến sự tồn tại của các đối tượng trong hệ thống. Các lớp xuất hiện trong lớp này xử lý cách mà các đối tượng có thể được lưu trữ và truy xuất. Các lớp chứa trong lớp này được gọi là các lớp Truy cập và Thao tác Dữ liệu (DAM). Các lớp DAM cho phép các lớp miền vấn đề độc lập với cách lưu trữ được sử dụng, từ đó tăng tính di động của hệ thống đang phát triển. Một số vấn đề liên quan đến lớp này bao gồm lựa chọn định dạng lưu trữ và tối ưu hóa. Có rất nhiều tùy chọn khác nhau để chọn cách lưu trữ các đối tượng. Chúng bao gồm các tệp tuần tự, tệp truy cập ngẫu nhiên, cơ sở dữ liệu quan hệ, cơ sở dữ liệu đối tượng/quan hệ, cơ sở dữ liệu đối tượng, và các kho dữ liệu NoSQL. Mỗi tùy chọn này đã được tối ưu hóa để cung cấp giải pháp cho các vấn đề truy cập và lưu trữ khác nhau. Ngày nay, từ góc độ thực tiễn, không có giải pháp đơn lẻ nào phục vụ tốt nhất cho tất cả các ứng dụng. Giải pháp đúng có thể là một sự kết hợp của các tùy chọn lưu trữ khác nhau. Một mô tả đầy đủ về tất cả các vấn đề liên quan đến lớp quản lý dữ liệu là vượt quá phạm vi của cuốn sách này. Tuy nhiên, chúng tôi sẽ trình bày các nguyên tắc cơ bản trong Chương 9.

**Tương tác giữa Người và Máy tính**

Lớp tương tác giữa người và máy tính chứa các lớp liên quan đến ý tưởng View và Controller từ Smalltalk. Mục đích chính của lớp này là giữ cho việc triển khai giao diện người dùng cụ thể tách biệt với các lớp miền vấn đề. Điều này tăng tính di động của hệ thống đang phát triển. Các lớp điển hình được tìm thấy trên lớp này bao gồm các lớp có thể được sử dụng để đại diện cho các nút bấm, cửa sổ, trường văn bản, thanh cuộn, hộp kiểm, danh sách thả xuống, và nhiều lớp khác đại diện cho các yếu tố giao diện người dùng.

Khi thiết kế giao diện người dùng cho một ứng dụng, nhiều vấn đề phải được giải quyết: Tính nhất quán giữa các giao diện người dùng khác nhau quan trọng đến mức nào? Thế còn các mức độ trải nghiệm người dùng khác nhau thì sao? Người dùng được mong đợi có thể điều hướng qua hệ thống như thế nào? Hệ thống trợ giúp và hướng dẫn trực tuyến thì sao? Những loại yếu tố đầu vào nào nên được bao gồm? Những loại yếu tố đầu ra nào nên được bao gồm? Các câu hỏi khác cần được giải quyết liên quan đến nền tảng mà phần mềm sẽ được triển khai. Ví dụ, ứng dụng có chạy trên máy tính độc lập không, hay nó sẽ được phân phối, hoặc ứng dụng có chạy trên thiết bị di động không? Nếu nó được mong đợi chạy trên các thiết bị di động, thì nền tảng nào: máy tính xách tay, máy tính bảng, hay điện thoại? Nó sẽ được triển khai bằng công nghệ web, có thể chạy trên nhiều thiết bị, hay sẽ được tạo bằng các ứng dụng dựa trên Android của Google, iOS của Apple, hoặc Windows của Microsoft? Tùy thuộc vào câu trả lời cho các câu hỏi này, các loại giao diện người dùng khác nhau là khả thi.

Với sự ra đời của các nền tảng mạng xã hội, chẳng hạn như Facebook, Twitter, blog, YouTube, và LinkedIn, những ảnh hưởng đối với giao diện người dùng có thể rất phức tạp. Tùy thuộc vào ứng dụng, các nền tảng mạng xã hội khác nhau có thể thích hợp cho các khía cạnh khác nhau của ứng dụng. Hơn nữa, mỗi nền tảng mạng xã hội khác nhau đều cho phép (hoặc ngăn cản) việc xem xét các loại giao diện người dùng khác nhau. Cuối cùng, với đối tượng tiềm năng của ứng dụng bạn là toàn cầu, nhiều vấn đề văn hóa sẽ phát sinh trong việc thiết kế và phát triển các giao diện người dùng nhạy cảm về văn hóa (chẳng hạn như yêu cầu đa ngôn ngữ). Rõ ràng, một mô tả hoàn chỉnh về tất cả các vấn đề liên quan đến tương tác giữa người và máy tính vượt quá phạm vi của cuốn sách này. Tuy nhiên, từ góc độ của người dùng, giao diện người dùng chính là hệ thống. Chúng tôi sẽ trình bày các vấn đề cơ bản trong thiết kế giao diện người dùng trong Chương 10.

**Kiến trúc Vật lý**

Lớp kiến trúc vật lý giải quyết cách mà phần mềm sẽ thực thi trên các máy tính và mạng cụ thể. Lớp này bao gồm các lớp liên quan đến việc giao tiếp giữa phần mềm và hệ điều hành của máy tính cũng như mạng. Ví dụ, các lớp giải quyết cách tương tác với các cổng khác nhau trên một máy tính cụ thể được bao gồm trong lớp này.

Khác với lớp nền, có nhiều vấn đề thiết kế cần được giải quyết trước khi chọn tập hợp các lớp phù hợp cho lớp này. Những vấn đề thiết kế này bao gồm lựa chọn kiến trúc máy tính hoặc mạng (chẳng hạn như các kiến trúc máy khách-máy chủ khác nhau), thiết kế thực tế của một mạng, thông số kỹ thuật phần cứng và phần mềm máy chủ, và các vấn đề bảo mật. Những vấn đề khác cần được giải quyết với thiết kế của lớp này bao gồm cấu hình phần cứng và phần mềm máy tính (lựa chọn hệ điều hành, chẳng hạn như Linux, Mac OSX và Windows; loại và tốc độ bộ xử lý; dung lượng bộ nhớ; lưu trữ dữ liệu; và công nghệ đầu vào/đầu ra), tiêu chuẩn hóa, ảo hóa, điện toán lưới, điện toán phân tán, và dịch vụ Web.

Điều này dẫn chúng ta đến một trong những vấn đề nổi cộm trong lĩnh vực này. Bạn sẽ làm gì với đám mây? Đám mây về cơ bản là một hình thức điện toán phân tán. Trong trường hợp này, đám mây cho phép bạn coi nền tảng, cơ sở hạ tầng, phần mềm, và thậm chí các quy trình kinh doanh như các dịch vụ từ xa có thể được quản lý bởi một công ty khác. Nhiều cách, đám mây cho phép phần lớn CNTT được thuê ngoài (xem thảo luận về thuê ngoài sau trong chương này). Ngoài ra, như đã đề cập ở lớp tương tác giữa con người và máy tính, vấn đề điện toán di động cũng rất quan trọng đối với lớp này. Cụ thể, các thiết bị khác nhau, chẳng hạn như điện thoại và máy tính bảng, có liên quan và cách chúng sẽ giao tiếp với nhau, chẳng hạn như qua mạng di động hoặc WiFi, cũng quan trọng.

Cuối cùng, với lượng năng lượng mà CNTT yêu cầu ngày nay, toàn bộ chủ đề về CNTT xanh cũng cần được đề cập. Các chủ đề cần được giải quyết liên quan đến CNTT xanh là vị trí của trung tâm dữ liệu, làm mát trung tâm dữ liệu, nguồn năng lượng thay thế, giảm thiểu tiêu thụ, ý tưởng về văn phòng không giấy, tuân thủ Energy Star, và tác động tiềm tàng của ảo hóa, đám mây, và điện toán di động. Giống như các lớp quản lý dữ liệu và tương tác giữa con người và máy tính, một mô tả hoàn chỉnh về tất cả các vấn đề liên quan đến kiến trúc vật lý vượt quá phạm vi của cuốn sách này. Tuy nhiên, chúng tôi trình bày những vấn đề cơ bản trong Chương 11.

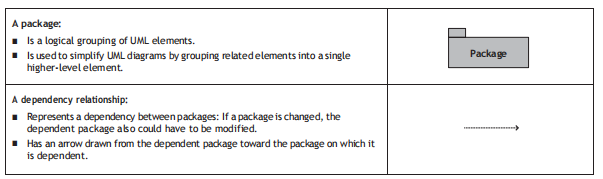
**GÓI VÀ SƠ ĐỒ GÓI**

Trong UML, các mối quan hệ hợp tác, phân vùng và lớp có thể được đại diện bởi một cấu trúc cấp cao hơn: một gói. Trên thực tế, một gói phục vụ cùng một mục đích như một thư mục trên máy tính của bạn. Khi các gói được sử dụng trong các ngôn ngữ lập trình như Java, các gói thực sự được triển khai dưới dạng thư mục. Một gói là một khái niệm chung có thể áp dụng cho bất kỳ yếu tố nào trong các mô hình UML. Trong Chương 4, chúng tôi đã giới thiệu ý tưởng về các gói như một cách để nhóm các trường hợp sử dụng lại với nhau để làm cho các sơ đồ trường hợp sử dụng dễ đọc hơn và giữ cho các mô hình ở mức độ phức tạp hợp lý. Trong các Chương 5 và 6, chúng tôi đã làm điều tương tự cho sơ đồ lớp và sơ đồ giao tiếp, tương ứng.

Trong phần này, chúng tôi mô tả một sơ đồ gói: một sơ đồ chỉ được cấu thành từ các gói. Một sơ đồ gói về cơ bản là một sơ đồ lớp chỉ hiển thị các gói.

Biểu tượng cho một gói giống như một thư mục có tab (xem Hình 7-18). Tùy thuộc vào nơi một gói được sử dụng, các gói có thể tham gia vào các loại quan hệ khác nhau. Ví dụ, trong một sơ đồ lớp, các gói đại diện cho các nhóm lớp. Do đó, các mối quan hệ tổng hợp và liên kết là có thể.

Trong một sơ đồ gói, điều hữu ích là thể hiện một mối quan hệ mới, mối quan hệ phụ thuộc. Một mối quan hệ phụ thuộc được thể hiện bằng một mũi tên gạch ngang (xem Hình 7-18). Một mối quan hệ phụ thuộc đại diện cho thực tế rằng một phụ thuộc thay đổi tồn tại giữa hai gói. Đó là, có thể rằng một sự thay đổi trong một gói có thể gây ra sự thay đổi cần thiết trong một gói khác. Hình 7-19 mô tả các phụ thuộc giữa các lớp khác nhau (lớp nền, lớp vấn đề, quản lý dữ liệu, tương tác giữa con người và máy tính, và kiến trúc vật lý). Ví dụ, nếu có một thay đổi xảy ra trong lớp vấn đề, nó rất có thể sẽ gây ra sự thay đổi xảy ra trong lớp tương tác giữa con người và máy tính, kiến trúc vật lý, và lớp quản lý dữ liệu. Lưu ý rằng những lớp này chỉ ra lớp vấn đề và do đó phụ thuộc vào nó. Tuy nhiên, điều ngược lại không đúng. Cũng lưu ý rằng tất cả các lớp đều phụ thuộc vào lớp nền. Điều này là do nội dung của lớp nền là các lớp cơ bản mà từ đó tất cả các lớp khác sẽ được xây dựng. Do đó, bất kỳ thay đổi nào được thực hiện đối với lớp này có thể có tác động đến tất cả các lớp khác.

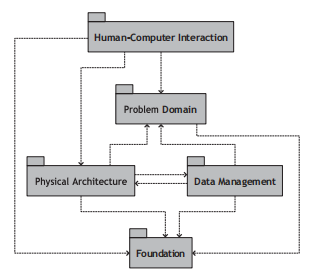


**HÌNH 7-18 Cú pháp cho Sơ đồ Gói**

Tại cấp lớp, có nhiều nguyên nhân có thể gây ra sự phụ thuộc giữa các lớp. Ví dụ, nếu giao thức cho một phương thức được thay đổi, thì điều này sẽ gây ra sự thay đổi giao diện cho tất cả các đối tượng của lớp này. Do đó, tất cả các lớp có đối tượng gửi tin nhắn đến các phiên bản của lớp đã được sửa đổi có thể cần phải được điều chỉnh. Việc ghi lại các mối quan hệ phụ thuộc giữa các lớp và gói giúp tổ chức duy trì hệ thống thông tin hướng đối tượng.

Các mối quan hệ hợp tác, phân vùng và lớp được mô hình hóa dưới dạng các gói trong UML. Các hợp tác thường được chia thành một tập hợp các phân vùng, thường được đặt trên một lớp. Các phân vùng có thể được cấu thành từ các phân vùng khác. Ngoài ra, cũng có thể có các lớp trong các phân vùng, mà được chứa trong một phân vùng khác, được đặt trên một lớp. Tất cả các nhóm này đều được đại diện bằng các gói trong UML. Hãy nhớ rằng một gói chỉ là một cấu trúc nhóm tổng quát được sử dụng để đơn giản hóa các mô hình UML thông qua việc sử dụng cấu trúc hợp thành.

Một sơ đồ gói đơn giản, dựa trên ví dụ về hệ thống đặt lịch từ các chương trước, được hiển thị trong Hình 7-20. Sơ đồ này chỉ thể hiện một phần rất nhỏ của toàn bộ hệ thống. Trong trường hợp này, chúng ta thấy rằng các lớp Patient UI, Patient-DAM và Patient Table phụ thuộc vào lớp Patient. Hơn nữa, lớp Patient-DAM phụ thuộc vào lớp Patient Table. Điều tương tự cũng có thể thấy với các lớp liên quan đến các cuộc hẹn thực tế. Bằng cách tách biệt các lớp miền vấn đề (chẳng hạn như các lớp Patient và Appt) khỏi các lớp lưu trữ đối tượng thực tế (chẳng hạn như các lớp Patient Table và Appt Table) thông qua việc sử dụng các lớp quản lý dữ liệu trung gian (các lớp Patient-DAM và Appt-DAM), chúng ta tách biệt các lớp miền vấn đề khỏi phương tiện lưu trữ thực tế. Điều này làm đơn giản hóa việc bảo trì và tăng cường khả năng tái sử dụng của các lớp miền vấn đề. Tất nhiên, trong một mô tả hoàn chỉnh của một hệ thống thực tế, sẽ có nhiều mối quan hệ phụ thuộc hơn.

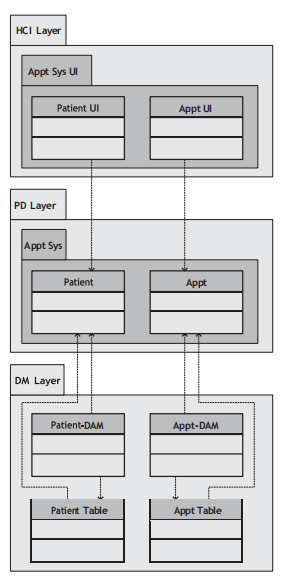


**HÌNH 7-19 Sơ đồ Gói về Các Mối Quan Hệ Phụ Thuộc giữa Các Lớp**

**Hướng dẫn Tạo Sơ đồ Gói**

Cũng như các sơ đồ UML được mô tả trong các chương trước, chúng tôi cung cấp một tập hợp các hướng dẫn mà chúng tôi đã điều chỉnh từ Ambler để tạo sơ đồ gói. Trong trường hợp này, chúng tôi đưa ra sáu hướng dẫn:

* Sử dụng sơ đồ gói để tổ chức logic các thiết kế. Cụ thể, sử dụng các gói để nhóm các lớp lại với nhau khi có mối quan hệ kế thừa, tổng hợp hoặc cấu thành giữa chúng hoặc khi các lớp hình thành một sự hợp tác.
* Trong một số trường hợp, có thể tồn tại các mối quan hệ kế thừa, tổng hợp hoặc liên kết giữa các gói. Trong những trường hợp đó, vì mục đích dễ đọc, hãy cố gắng hỗ trợ các mối quan hệ kế thừa theo chiều dọc, với gói chứa lớp cha được đặt trên gói chứa lớp con. Sử dụng vị trí nằm ngang để hỗ trợ các mối quan hệ tổng hợp và liên kết, với các gói được đặt cạnh nhau.
* Khi một mối quan hệ phụ thuộc tồn tại trên một sơ đồ, nó ngụ ý rằng có ít nhất một mối quan hệ ngữ nghĩa giữa các phần tử của hai gói. Hướng của sự phụ thuộc thường từ lớp con đến lớp cha, từ toàn bộ đến phần, và với các hợp đồng, từ khách hàng đến máy chủ. Nói cách khác, một lớp con phụ thuộc vào sự tồn tại của một lớp cha, một toàn bộ phụ thuộc vào sự tồn tại của các phần của nó, và một khách hàng không thể gửi tin nhắn đến một máy chủ không tồn tại.
* Khi sử dụng các gói để nhóm các trường hợp sử dụng lại với nhau, hãy chắc chắn bao gồm các diễn viên và các mối quan hệ mà họ có với các trường hợp sử dụng được nhóm trong gói. Điều này sẽ cho phép người sử dụng sơ đồ hiểu rõ hơn về ngữ cảnh của sơ đồ.
* Đặt tên cho mỗi gói một cách đơn giản nhưng mô tả để cung cấp cho người sử dụng sơ đồ gói đủ thông tin để hiểu gói đó chứa đựng điều gì. Nếu không, người sử dụng sẽ phải tìm hiểu sâu hơn hoặc mở gói ra để hiểu mục đích của gói.
* Đảm bảo rằng các gói có tính liên kết cao. Để một gói có tính liên kết, các lớp chứa trong gói đó, theo một cách nào đó, phải thuộc về nhau. Một quy tắc đơn giản nhưng không hoàn hảo để làm theo khi nhóm các lớp lại trong một gói là rằng, càng nhiều lớp phụ thuộc vào nhau, thì càng có khả năng chúng thuộc về cùng một gói.

****

**HÌNH 7-20 Sơ đồ Gói Một Phần của Hệ Thống Đặt Lịch**