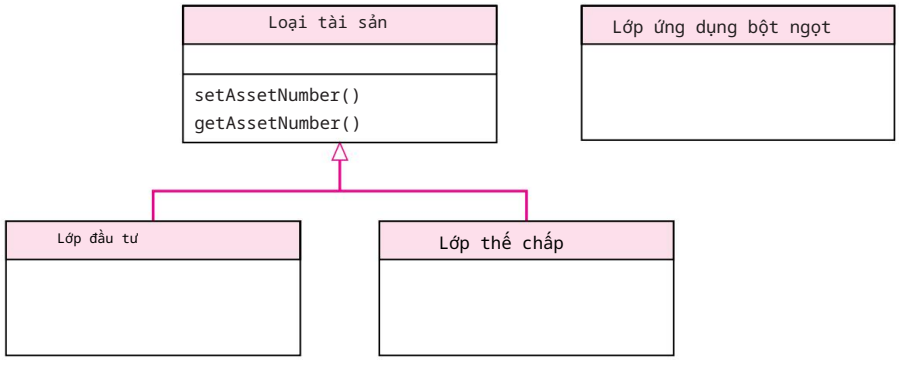
HÌNH 14.15 Một phần của sơ đồ lớp cho nghiên cứu tình huống của MSG Foundation với các phương thức setAssetNumber và getAssetNumber được gán cho Lớp tài sản.



có thể được áp dụng không chỉ cho các thể hiện của Loại Tài sản mà còn, như một hệ quả của tính kế thừa, cho các thể hiện của mọi phân lớp của Loại Tài sản ,, tức là lớp Đầu tư thể loại Thế chấp . Tương tự, phương thức getAssetNumber cũng sẽ nên được phân bổ cho lớp trên Lớp tài sản .

Việc gán các phương thức khác cho các lớp thích hợp cũng đơn giản như nhau. Kết quả thiết kế được thể hiện trong Phụ lục G.

Bước 2. Thực hiện thiết kế chi tiết Tiếp theo, thiết kế chi tiết được xây dựng bằng cách thực hiện từng phương pháp và xác định chức năng của nó. Hình 14.16 cho thấy thiết kế chi tiết (trong PDL cho Java) của một phương thức com puteEstimatedFunds của lớp EstimateFundsForWeek của nghiên cứu điển hình về Tổ chức MSG. Phương thức này gọi phương thức totalWeeklyNetPayments của lớp Mortgage thể hiện trong Hình 14.17.

Các bước của thiết kế hướng đối tượng được tóm tắt trong Cách thực hiện Hộp 14.3.

14.9 Quy trình thiết kế

Mục đích tổng thể của quy trình thiết kế là tinh chỉnh các thành phần tạo tác của quy trình phân tích cho đến khi tài liệu ở dạng có thể được thực hiện bởi các lập trình viên. Do đó, đầu vào của quy trình công việc thiết kế là các tạo phẩm của quy trình công việc phân tích (Chương 13). Trong quy trình thiết kế, các tạo phẩm này được lặp đi lặp lại và tăng dần cho đến khi chúng ở định dạng mà các lập trình viên có thể sử dụng.

Cách thực hiện thiết kế hướng đối tượng — Hộp 14.3

• Hoàn thành sơ đồ lớp. •
Thực hiện thiết kế chi tiết.

HÌNH 14.16
Thiết kế chi tiết của phương pháp tính toán Quỹ ước tính của lớp Ước tính quỹ cho tuần của nghiên cứu điển hình về Tổ chức MSG.

```
công khai khoảng trống tính toánEstimatedFunds()

Phương pháp này tính toán số tiền ước tính có sẵn trong tuần.

{

   fl oat dự kiếnLợi tức đầu tư hàng tuần;                    ( lợi nhuận đầu tư hàng tuần dự kiến )

   fl oat dự kiếnTotalWeeklyNetPayments = ( fl oat ) 0,0;

                                                              ( tổng số tiền thanh toán thế chấp dự kiến trừ
                                                              đi tổng số tiền trợ cấp hàng tuần )

   fl oat Quỹ ước tính = ( fl oat ) 0,0;                      ( tổng số tiền ước tính trong tuần )

Tạo một thể hiện của một hồ sơ đầu tư.

   Đầu tư inv = Đầu tư mới ( );

Tạo một thể hiện của hồ sơ thế chấp.

   Thế chấp thế chấp = thế chấp mới ( );

Gọi phương thức totalWeeklyReturnOnInvestment.

   kỳ vọngWeeklyInvestmentReturn = inv.totalWeeklyReturnOnInvestment ( );

Gọi phương thức dự kiếnTotalWeeklyNetPayments              ( xem Hình 14.17 )

   dự kiếnTotalWeeklyNetPayments = mort.totalWeeklyNetPayments ( );

Bây giờ hãy tính số tiền ước tính trong tuần.

   Quỹ ước tính = (Lợi tức đầu tư hàng tuần dự kiến

            (MSGApplication.getAnnualOperatingExpenses ( ) / ( fl oat ) 52,0)

          + dự kiếnTotalWeeklyNetPayments);


Lưu trữ giá trị này ở vị trí thích hợp.

   MSGApplication.setEstimatedFundsForWeek (Quỹ ước tính);

} // tính toánEstimatedFunds
```

Một khía cạnh của việc lặp lại và gia tăng này là việc xác định các phương thức và sự phân bổ của chúng cho các lớp thích hợp. Một khía cạnh khác là thực hiện thiết kế chi tiết. Hai bước này cấu thành thành phần thiết kế hướng đối tượng của quy trình thiết kế.

Ngoài việc thực hiện thiết kế hướng đối tượng, nhiều quyết định phải được thực hiện như một phần của quy trình thiết kế. Một quyết định như vậy là việc lựa chọn ngôn ngữ lập trình mà sản phẩm phần mềm sẽ được triển khai. Quá trình này được mô tả chi tiết trong Chương 15 . Một quyết định khác là sử dụng lại bao nhiêu sản phẩm phần mềm hiện có trong sản phẩm phần mềm mới sẽ được phát triển. Tái sử dụng được mô tả trong Chương 8 . Tính di động là một quyết định thiết kế quan trọng khác; chủ đề này cũng vậy, được mô tả trong Chương 8 . Ngoài ra, các sản phẩm phần mềm lớn thường được triển khai trên mạng máy tính; một quyết định thiết kế khác là phân bổ từng thành phần phần mềm cho thành phần phần cứng mà nó sẽ chạy trên đó.

Động lực chính đằng sau sự phát triển của Quy trình Unifi ed là trình bày một phương pháp có thể được sử dụng để phát triển các sản phẩm phần mềm quy mô lớn, điển hình là 500.000 dòng mã trở lên. Mặt khác, việc triển khai nghiên cứu tình huống của Tổ chức MSG trong Phụ lục H và tôi lần lượt là ít hơn 5000 dòng của C++ và Java. trong khác

HÌNH 14.17
Thiết kế chi tiết của phương thức TotalWeekly NetPayments của lớp Thế chấp trong nghiên cứu tình huống của Quỹ MSG.

```
tổng số tiền chuyển nhượng công khaiWeeklyNetPayments ( )
Phương pháp này tính toán tổng số tiền thanh toán ròng hàng tuần được thực hiện bởi những người thế chấp, nghĩa là tổng số tiền thế
chấp hàng tuần dự kiến trừ đi tổng số tiền trợ cấp hàng tuần dự kiến.

{
   Tệp thế chấpFile = tệp mới ("mortgage.dat");                          ( file hồ sơ thế chấp )
   fl oat kỳ vọng Tổng số tiền thế chấp hàng tuần = ( fl oat) 0,0;        ( tổng số tiền thanh toán thế chấp hàng tuần dự kiến )
   fl oat dự kiếnTotalWeeklyGrants = ( fl oat ) 0,0;                     ( tổng số tiền trợ cấp hàng tuần dự kiến )
   fl yến lãiThanh toán;                                                 ( trả lãi )
   fl oat escrowThanh toán;                                              ( thanh toán ký quỹ )
   fl oat vốnTrả nợ;                                                     ( trả vốn )
   fl oatThanh toán hàng tuần;                                           ( thanh toán thế chấp trong tuần )
   fl oat tối đaCho phépThanh toán thế chấp;                             ( số tiền tối đa mà cặp vợ chồng có thể trả )
Mở tệp thế chấp, đặt tên trong Tệp ,và đọc lần lượt từng phần tử.
{
   đã đọc (trong Tệp);
Tính tiền lãi, tiền ký quỹ và tiền hoàn trả vốn cho khoản thế chấp này.
   tiền lãiThanh toán = Số dư thế chấp * INTEREST_RATE / WEEKS_IN_YEAR ;
   Khoản thanh toán ký quỹ = (Thuế tài sản hàng năm + Phí bảo hiểm hàng năm) / WEEKS_IN_YEAR;
   vốnTrả nợ = trả gốc và trả lãi hàng tuần    trả lãi;
   thế chấpSố dư   =vốnTrả nợ;
Đầu tiên, giả sử rằng cặp vợ chồng có thể trả toàn bộ khoản thế chấp mà không cần trợ cấp.
   Thanh toán hàng tuần = Thanh toán gốc và lãi hàng tuần + Thanh toán ký quỹ;
Thêm khoản thanh toán gốc và lãi hàng tuần vào tổng số khoản thanh toán thế chấp đang diễn ra
   dự kiếnTổng số tiền thế chấp hàng tuần += Khoản thanh toán gốc và lãi hàng tuần;
Bây giờ hãy xác định số tiền mà cặp vợ chồng thực sự có thể trả.
   thanh toán thế chấp tối đa được phép = thu nhập hàng tuần hiện tại *
       MAXIMUM_PERC_OF_INCOME;
Nếu cần trợ cấp, hãy thêm số tiền trợ cấp vào tổng số trợ cấp hiện có
   nếu (Thanh toán hàng tuần > Khoản thanh toán thế chấp tối đa được phép)
   dự kiếnTổng số tiền trợ cấp hàng tuần += Khoản thanh toán hàng tuần    Khoản thanh toán thế chấp tối đa được phép;
   }
Đóng hồ sơ thế chấp. Trả về tổng số tiền thanh toán ròng dự kiến trong tuần.
   trả về (dự kiếnTotalWeeklyMortgages   expectedTotalWeeklyGrants);
} // tổng thanh toán hàng tuầnNet
```

Nói cách khác, Quy trình Unifi ed chủ yếu dành cho các sản phẩm phần mềm lớn hơn ít nhất 100 lần so với nghiên cứu tình huống của Tổ chức MSG được trình bày trong cuốn sách này. Theo đó, nhiều khía cạnh của Tiến trình Unifi ed không thể áp dụng cho nghiên cứu điển hình này. Ví dụ, một phần quan trọng của quy trình phân tích là phân vùng sản phẩm phần mềm thành các gói phân tích.
Mỗi gói bao gồm một tập hợp các lớp có liên quan, thường liên quan đến một tập hợp con nhỏ của các tác nhân, có thể được triển khai như một đơn vị duy nhất. Ví dụ: tài khoản phải trả, tài khoản phải thu và sổ cái chung là các gói phân tích điển hình. Khái niệm bên dưới các gói phân tích nói dối là việc phát triển các sản phẩm phần mềm nhỏ hơn sẽ dễ dàng hơn nhiều so với các sản phẩm phần mềm lớn hơn. Theo đó, một sản phẩm phần mềm lớn sẽ dễ phát triển hơn nếu nó có thể được phân tách thành các gói tương đối độc lập. Phân tách một sản phẩm phần mềm thành các gói là một ví dụ về chia để trị (Phần 5.3).

Ý tưởng phân tách một luồng công việc lớn thành các luồng công việc nhỏ hơn tương đối độc lập được chuyển sang luồng công việc thiết kế. Ở đây, mục tiêu là chia nhỏ quy trình triển khai sắp tới thành các phần có thể quản lý được, được gọi là các hệ thống con . Một lần nữa, sẽ không hợp lý nếu chia nghiên cứu tình huống của Tổ chức MSG thành các hệ thống con; nghiên cứu trường hợp chỉ là quá nhỏ.

Có hai lý do tại sao các luồng công việc lớn hơn được chia thành các hệ thống con:

1. Như đã giải thích trước đây, việc triển khai một số hệ thống con nhỏ sẽ dễ dàng hơn so với một hệ thống lớn. Nghĩa là, chia nhỏ một sản phẩm phần mềm thành các hệ thống con là một ví dụ khác về chia để trị (Phần 5.3).
2. Nếu các hệ thống con được triển khai thực sự tương đối độc lập, thì chúng có thể được triển khai bởi các nhóm lập trình làm việc song song. Điều này dẫn đến toàn bộ sản phẩm phần mềm được chuyển giao sớm hơn.

Nhớ lại từ Phần 8.5.4 rằng kiến trúc của một sản phẩm phần mềm bao gồm các thành phần khác nhau và cách chúng khớp với nhau. Việc phân bổ các thành phần cho các hệ thống con là một phần chính của nhiệm vụ kiến trúc. Quyết định về kiến trúc của một sản phẩm phần mềm không hề dễ dàng và trong tất cả, trừ những sản phẩm phần mềm nhỏ nhất, được thực hiện bởi một chuyên gia, kiến trúc sư phần mềm .

Ngoài việc là một chuyên gia kỹ thuật, một kiến trúc sư cần biết cách làm cho một sản phẩm phần sự đánh đổi ·mềm phải thỏa mãn các yêu cầu chức năng, tức là các trường hợp sử dụng. Nó cũng cần đáp ứng các yêu cầu phi chức năng, bao gồm tính di động (Chương 8), độ tin cậy (Phần 6.4.2), độ bền (Phần 6.4.3), khả năng bảo trì và bảo mật. Nhưng nó cần phải làm tất cả những điều này trong giới hạn ngân sách và thời gian. Gần như không bao giờ có thể phát triển một sản phẩm phần mềm đáp ứng tất cả các yêu cầu của nó, cả về chức năng và phi chức năng, và hoàn thành dự án trong những ràng buộc về chi phí và thời gian; thỏa hiệp hầu như luôn luôn phải được thực hiện. Khách hàng phải nới lỏng một số yêu cầu, tăng ngân sách hoặc dời thời hạn giao hàng hoặc thực hiện nhiều hơn một trong những yêu cầu này. Kiến trúc sư phải hỗ trợ quá trình ra quyết định của khách hàng bằng cách vạch ra rõ ràng những sự đánh đổi.

Trong một số trường hợp, sự đánh đổi là rõ ràng. Ví dụ, kiến trúc sư có thể chỉ ra rằng một tập hợp các yêu cầu bảo mật phù hợp với tiêu chuẩn bảo mật cao mới sẽ mất thêm 3 tháng và 350.000 USD để đưa vào sản phẩm phần mềm. Nếu sản phẩm là một mạng lưới ngân hàng quốc tế, vấn đề sẽ được tranh luận – không có cách nào mà khách hàng có thể đồng ý thỏa hiệp về bảo mật theo bất kỳ cách nào. Tuy nhiên, trong các trường hợp khác, khách hàng cần đưa ra các quyết định quan trọng liên quan đến sự đánh đổi và phải dựa vào kỹ thuật.

chuyên môn của kiến trúc sư để hỗ trợ đưa ra quyết định kinh doanh đúng đắn. Ví dụ, kiến trúc sư có thể chỉ ra rằng việc trì hoãn một yêu cầu cụ thể cho đến khi sản phẩm phần mềm được chuyển giao và đang được bảo trì có thể tiết kiệm được 150.000 đô la ngay bây giờ nhưng sẽ tốn 300.000 đô la để kết hợp sau này (xem Hình 1.6). Quyết định có trì hoãn yêu cầu hay không chỉ có thể được đưa ra bởi khách hàng, nhưng họ cần chuyên môn kỹ thuật của kiến trúc sư để hỗ trợ đưa ra quyết định chính xác.

Kiến trúc của một sản phẩm phần mềm là một yếu tố sống còn trong sự thành công hay thất bại của sản phẩm được chuyển giao. Và các quyết định quan trọng liên quan đến kiến trúc phải được đưa ra trong khi thực hiện quy trình thiết kế. Nếu quy trình công việc yêu cầu được thực hiện kém, vẫn có thể có một dự án thành công, với điều kiện là phải dành thêm thời gian và tiền bạc cho quy trình công việc phân tích. Tương tự, nếu quy trình phân tích không đầy đủ, có thể phục hồi bằng cách nỗ lực thêm như một phần của quy trình thiết kế. Nhưng nếu kiến trúc dưới mức tối ưu, thì không có cách nào để phục hồi; kiến trúc phải ngay lập tức được thiết kế lại. Do đó, điều cần thiết là nhóm phát triển phải bao gồm một kiến trúc sư có chuyên môn kỹ thuật và kỹ năng con người cần thiết.

14.10 Quy trình thử nghiệm: Thiết kế

Mục tiêu của kiểm tra thiết kế là để xác minh rằng các thông số kỹ thuật đã được đưa vào thiết kế một cách chính xác và đầy đủ cũng như để đảm bảo tính đúng đắn của bản thân thiết kế. Ví dụ, thiết kế phải không có lỗi logic và tất cả các giao diện phải được xác định chính xác. Điều quan trọng là bất kỳ lỗi nào trong thiết kế đều được phát hiện trước khi bắt đầu viết mã; nếu không, chi phí sửa lỗi sẽ cao hơn đáng kể, như được phản ánh trong Hình 1.6. Lỗi thiết kế có thể được phát hiện bằng phương pháp kiểm tra thiết kế cũng như hướng dẫn thiết kế. Việc kiểm tra thiết kế được thảo luận trong phần còn lại của phần này, nhưng các nhận xét cũng được áp dụng như nhau đối với hướng dẫn thiết kế.

Khi sản phẩm được định hướng giao dịch (Phần 14.4), việc kiểm tra thiết kế sẽ phản ánh điều này [Beizer, 1990]. Việc kiểm tra bao gồm tất cả các loại giao dịch có thể nên được lên lịch. Người đánh giá nên liên hệ từng giao dịch trong thiết kế với các thông số kỹ thuật, chỉ ra cách giao dịch phát sinh từ tài liệu thông số kỹ thuật. Ví dụ: nếu ứng dụng là một máy rút tiền tự động, một giao dịch tương ứng với từng thao tác mà khách hàng có thể thực hiện, chẳng hạn như gửi tiền vào hoặc rút tiền từ tài khoản thẻ tín dụng. Trong các trường hợp khác, sự tương ứng giữa các thông số kỹ thuật và giao dịch không nhất thiết phải là một đối một. Ví dụ: trong hệ thống điều khiển đèn giao thông, nếu một ô tô lái qua bảng cảm biến khiến hệ thống quyết định thay đổi một đèn cụ thể từ đỏ sang xanh trong 15 giây, thì các xung tiếp theo từ bảng cảm biến đó có thể bị bỏ qua. Ngược lại, để tăng tốc độ giao thông, một xung lực duy nhất có thể khiến toàn bộ dãy đèn chuyển từ màu đỏ sang màu xanh lá cây.

Hạn chế đánh giá để kiểm tra theo hướng giao dịch không phát hiện trường hợp các nhà thiết kế đã bỏ qua các trường hợp giao dịch theo yêu cầu của các thông số kỹ thuật. Lấy một ví dụ cực đoan, các thông số kỹ thuật của bộ điều khiển đèn giao thông có thể quy định rằng trong khoảng thời gian từ 11:00 đêm đến 6:00 sáng , tất cả các đèn sẽ nhấp nháy màu vàng tro ở một hướng và đỏ ở hướng khác. Nếu các nhà thiết kế bỏ qua quy định này, thì các giao dịch do đồng hồ tạo ra lúc 11:00 tối và 6:00 sáng sẽ không được đưa vào thiết kế; và nếu các giao dịch này bị bỏ qua, chúng không thể được kiểm tra trong quá trình kiểm tra thiết kế dựa trên

giao dịch. Do đó, việc lên lịch kiểm tra thiết kế chỉ nhằm thúc đẩy hành động chuyển đổi là không phù hợp; kiểm tra theo định hướng đặc điểm kỹ thuật cũng rất cần thiết để đảm bảo rằng không có tuyên bố nào trong tài liệu đặc điểm kỹ thuật bị bỏ sót hoặc hiểu sai.

nghiên cứu điển hình

14.11 Quy trình thử nghiệm: Quỹ MSG

nghiên cứu điển hình

Bây giờ thì thiết kế rõ ràng đã hoàn tất, tất cả các khía cạnh thiết kế của nghiên cứu điển hình về Quỹ MSG phải được kiểm tra bằng phương tiện kiểm tra thiết kế (Phần 6.2.3). Đặc biệt, mỗi hiện vật thiết kế phải được kiểm tra. Ngay cả khi không tìm thấy lỗi nào, có thể thiết kế sẽ thay đổi một lần nữa, có lẽ là triệt để, khi nghiên cứu tình huống của Tổ chức MSG được triển khai.

14.12 Kỹ thuật chính thức cho thiết kế chi tiết

Một kỹ thuật thiết kế chi tiết đã được trình bày. Trong Phần 5.1, mô tả về tinh chỉnh từng bước đã được đưa ra. Sau đó, nó được áp dụng cho thiết kế chi tiết bằng cách sử dụng lưu đồ.
Ngoài việc tinh chỉnh từng bước, các kỹ thuật hình thức có thể được sử dụng để tạo lợi thế trong thiết kế chi tiết. Chương 6 gợi ý rằng việc triển khai một sản phẩm hoàn chỉnh và sau đó chứng minh nó đúng có thể phản tác dụng. Tuy nhiên, phát triển song song bằng chứng và thiết kế chi tiết cũng như kiểm tra mã cẩn thận lại là một vấn đề hoàn toàn khác. Các kỹ thuật chính thức được áp dụng cho thiết kế chi tiết có thể hỗ trợ rất nhiều theo ba cách:

1. Công nghệ tiên tiến nhất trong việc chứng minh tính đúng đắn là mặc dù nhìn chung không thể áp dụng cho toàn bộ sản phẩm nhưng có thể áp dụng cho các phần có kích thước mô-đun của sản phẩm.
2. Phát triển bằng chứng cùng với thiết kế chi tiết sẽ dẫn đến một thiết kế có ít hơn lỗi hơn nếu bằng chứng tính đúng đắn không được sử dụng.
3. Nếu cùng một lập trình viên chịu trách nhiệm cho cả thiết kế chi tiết và triển khai, thì lập trình viên đó sẽ cảm thấy tự tin rằng thiết kế chi tiết là chính xác. Thái độ tích cực này đối với thiết kế sẽ dẫn đến ít lỗi hơn trong mã.

14.13 Kỹ thuật thiết kế thời gian thực

Như đã giải thích trong Phần 6.4.4, phần mềm thời gian thực được đặc trưng bởi các ràng buộc về thời gian cứng, tức là các ràng buộc về thời gian có tính chất như vậy, nếu một ràng buộc không được đáp ứng, thông tin sẽ bị mất. Đặc biệt, mỗi đầu vào phải được xử lý trước khi đầu vào tiếp theo đến. Một ví dụ về hệ thống như vậy là lò phản ứng hạt nhân do máy tính điều khiển. Các đầu vào như nhiệt độ của lõi và mực nước trong buồng lò phản ứng liên tục được gửi đến máy tính để đọc giá trị của từng đầu vào và thực hiện các thao tác cần thiết.

xử lý trước khi đầu vào tiếp theo đến. Một ví dụ khác là đơn vị chăm sóc đặc biệt được điều khiển bằng máy tính. Có hai loại dữ liệu bệnh nhân: thông tin thường quy như nhịp tim, nhiệt độ và huyết áp của từng bệnh nhân và thông tin cấp cứu, khi hệ thống suy luận rằng tình trạng của bệnh nhân đã trở nên nguy kịch. Khi những trường hợp khẩn cấp như vậy xảy ra, phần mềm phải xử lý cả đầu vào thông thường và đầu vào liên quan đến trường hợp khẩn cấp từ một hoặc nhiều bệnh nhân.

Một đặc điểm của nhiều hệ thống thời gian thực là chúng được thực hiện trên phần cứng phân tán. Ví dụ, phần mềm điều khiển máy bay chiến đấu có thể được triển khai trên năm máy tính: một máy tính điều khiển điều hướng, một máy tính khác dùng cho hệ thống vũ khí, máy tính thứ ba dùng cho các biện pháp đối phó điện tử, máy tính thứ tư dùng để điều khiển phần cứng của máy bay như cánh tà và động cơ, và người thứ năm đề xuất các chiến thuật trong chiến đấu. Vì phần cứng không hoàn toàn đáng tin cậy nên có thể có thêm các máy tính dự phòng tự động thay thế một thiết bị bị hỏng hóc. Không chỉ thiết kế của một hệ thống như vậy có ý nghĩa quan trọng về truyền thông, mà các vấn đề về thời gian, ngoài những vấn đề thuộc loại vừa mô tả, còn phát sinh do tính chất phân tán của hệ thống. Ví dụ, trong các điều kiện chiến đấu, máy tính chiến thuật có thể gợi ý phi công nên leo lên, trong khi máy tính vũ khí khuyến nghị phi công lặn xuống để một loại vũ khí cụ thể có thể được phóng trong điều kiện tối ưu. Tuy nhiên, phi công con người quyết định di chuyển cần lái sang phải, do đó gửi tín hiệu đến máy tính phần cứng máy bay để thực hiện các điều chỉnh cần thiết sao cho máy bay nghiêng theo hướng đã chỉ định. Tất cả những thông tin này phải được quản lý cẩn thận sao cho chuyển động thực tế của máy bay được ưu tiên hơn mọi thao tác được đề xuất. Hơn nữa, chuyển động thực tế phải được chuyển tiếp đến các máy tính chiến thuật và vũ khí để các đề xuất mới có thể được hình thành dưới ánh sáng của các điều kiện thực tế, thay vì được đề xuất.

Một khó khăn nữa với các hệ thống thời gian thực là vấn đề đồng bộ hóa. Giả sử rằng một hệ thống thời gian thực sẽ được triển khai trên phần cứng phân tán. Các tình huống như bế tắc (hoặc bế tắc chết người) có thể phát sinh khi hai hoạt động, mỗi hoạt động có quyền sử dụng độc quyền một mục dữ liệu và mỗi hoạt động yêu cầu sử dụng độc quyền mục dữ liệu của hoạt động kia. Tất nhiên, bế tắc không chỉ xảy ra trong các hệ thống thời gian thực, được thực hiện trên phần cứng phân tán. Nhưng nó đặc biệt rắc rối trong các hệ thống thời gian thực, nơi không có sự kiểm soát đối với thứ tự hoặc thời gian của các đầu vào và tình hình có thể phức tạp do tính chất phân tán của phần cứng. Ngoài bế tắc, các vấn đề đồng bộ hóa khác có thể xảy ra, bao gồm các điều kiện chủng tộc; để biết chi tiết, người đọc có thể tham khảo [Silberschatz, Galvin, và Gagne, 2002] hoặc các sách giáo khoa về hệ điều hành khác.

Từ những ví dụ này, rõ ràng là khó khăn chính đối với việc thiết kế các hệ thống thời gian thực là đảm bảo rằng các ràng buộc về thời gian được đáp ứng bởi thiết kế. Nghĩa là, kỹ thuật thiết kế sẽ cung cấp một cơ chế để kiểm tra xem, khi được triển khai, thiết kế có thể đọc và xử lý dữ liệu đến ở tốc độ yêu cầu hay không. Hơn nữa, có thể chỉ ra rằng các vấn đề đồng bộ hóa trong thiết kế cũng đã được giải quyết một cách chính xác.

Kể từ khi bắt đầu kỷ nguyên máy tính, những tiến bộ trong công nghệ phần cứng đã tước bỏ hầu hết mọi khía cạnh những tiến bộ trong công nghệ phần mềm. Do đó, mặc dù phần cứng tồn tại để xử lý mọi khía cạnh của hệ thống thời gian thực được mô tả trước đó, nhưng công nghệ thiết kế phần mềm đã bị tụt lại phía sau đáng kể. Trong một số lĩnh vực của công nghệ phần mềm thời gian thực, tiến bộ lớn đã được thực hiện. Ví dụ, nhiều kỹ thuật phân tích của Chương 12 và 13 có thể được sử dụng để xác định các hệ thống thời gian thực. Thật không may, thiết kế phần mềm vẫn chưa đạt đến mức tinh vi như vậy. Những bước tiến lớn thực sự đang được thực hiện, nhưng trạng thái của nghệ thuật vẫn chưa thể so sánh với những gì đã đạt được liên quan đến phân tích

kỹ xảo. Bởi vì hầu hết mọi kỹ thuật thiết kế cho các hệ thống thời gian thực đều tốt hơn là không có kỹ thuật nào, nên một số kỹ thuật thiết kế thời gian thực được sử dụng trong thực tế. Tuy nhiên, vẫn còn một chặng đường dài trước khi có thể thiết kế các hệ thống thời gian thực như những hệ thống được mô tả trước đó và chắc chắn rằng, trước khi hệ thống được triển khai, mọi ràng buộc thời gian thực sẽ được đáp ứng và các vấn đề đồng bộ hóa sẽ không thể xảy ra. nảy sinh.

Các kỹ thuật thiết kế thời gian thực cũ hơn là phần mở rộng của các kỹ thuật phi thời gian thực đối với miền thời gian thực. Ví dụ: phát triển có cấu trúc cho các hệ thống thời gian thực (SDRTS) [Ward và Mellor, 1985] về cơ bản là phần mở rộng của phân tích hệ thống có cấu trúc (Phần 12.3), phân tích luồng dữ liệu (Phần 14.3) và phân tích giao dịch (Phần 14.4) cho phần mềm thời gian thực. Kỹ thuật phát triển bao gồm một thành phần cho thiết kế thời gian thực. Các kỹ thuật mới hơn được mô tả trong [Liu, 2000] và [Gomaa, 2000].

Như đã nêu trước đây, thật không may là trình độ nghệ thuật của thiết kế thời gian thực không tiên tiến như người ta mong muốn. Tuy nhiên, các nỗ lực đang được tiến hành để cải thiện tình hình.

14.14 Công cụ CASE để thiết kế

Như đã nêu trong Phần 14.10, một khía cạnh quan trọng của thiết kế là kiểm tra xem các tạo tác thiết kế có kết hợp chính xác tất cả các khía cạnh của phân tích hay không. Do đó, điều cần thiết là một công cụ CASE có thể được sử dụng cho cả các tạo phẩm phân tích và các tạo phẩm thiết kế, cái gọi là công cụ front-end hoặc upperCASE (trái ngược với công cụ back-end hoặc LowerCASE, hỗ trợ triển khai các tạo phẩm).

Một số công cụ chữ hoa có trên thị trường. Một số công việc phổ biến hơn bao gồm Nhà phân tích/Nhà thiết kế, Phần mềm thông qua Hình ảnh và Kiến trúc sư Hệ thống. Các công cụ UpperCASE thường được xây dựng xung quanh một từ điển dữ liệu. Công cụ CASE có thể kiểm tra xem mọi trường của mọi bản ghi trong từ điển có được đề cập ở đâu đó trong thiết kế hay mọi mục trong thiết kế đều được phản ánh trong sơ đồ luồng dữ liệu. Ngoài ra, nhiều công cụ viết hoa kết hợp một trình kiểm tra tính nhất quán sử dụng từ điển dữ liệu để xác định rằng mọi mục trong thiết kế đã được khai báo trong thông số kỹ thuật và ngược lại rằng mọi mục trong thông số kỹ thuật đều xuất hiện trong thiết kế.

Hơn nữa, nhiều công cụ viết hoa kết hợp trình tạo màn hình và báo cáo. Nghĩa là, khách hàng có thể chỉ định mục nào sẽ xuất hiện trong báo cáo hoặc trên màn hình nhập liệu cũng như vị trí và cách thức xuất hiện của từng mục. Do chi tiết đầy đủ về mọi mục đều có trong từ điển dữ liệu nên công cụ CASE có thể dễ dàng tạo mã để in báo cáo hoặc hiển thị màn hình nhập liệu theo ý muốn của khách hàng. Một số sản phẩm chữ hoa cũng kết hợp các công cụ quản lý tỷ lệ để ước tính và lập kế hoạch.

Đối với thiết kế hướng đối tượng, IBM Rational Rose và Phần mềm thông qua Hình ảnh cùng nhau cung cấp hỗ trợ cho quy trình công việc này trong bối cảnh của vòng đời hướng đối tượng hoàn chỉnh. Các công cụ CASE mã nguồn mở thuộc loại này bao gồm ArgoUML.

14.15 Số liệu cho thiết kế

Một loạt các số liệu có thể được sử dụng để mô tả các khía cạnh của thiết kế. Ví dụ: số lượng tạo phẩm mã (mô-đun hoặc lớp) là thước đo thô về kích thước của sản phẩm mục tiêu.
Sự gắn kết và khớp nối là thước đo chất lượng của thiết kế, cũng như thống kê lỗi.
Như với tất cả các loại kiểm tra khác, điều quan trọng là phải lưu giữ hồ sơ về số lượng và loại

các lỗi thiết kế được phát hiện trong quá trình kiểm tra thiết kế. Thông tin này được sử dụng trong quá trình kiểm tra mã sản phẩm và kiểm tra thiết kế của các sản phẩm tiếp theo.

Độ phức tạp chu trình M của một thiết kế chi tiết là số quyết định nhị phân (vị từ) cộng với 1 [McCabe, 1976] hoặc, tương đương, số nhánh trong mã tạo tác. Có ý kiến cho rằng độ phức tạp chu trình là thước đo chất lượng thiết kế; giá trị của M càng thấp càng tốt. Một điểm mạnh của số liệu này là nó rất dễ tính toán. Tuy nhiên, nó có một vấn đề cố hữu. Độ phức tạp theo chu kỳ ,hoàn toàn là thước đo của độ phức tạp điều khiển, độ phức tạp của dữ liệu bị bỏ qua. Nghĩa là, M không đo lường độ phức tạp của một tạo phẩm mã được điều khiển bởi dữ liệu, chẳng hạn như bởi các giá trị trong một bảng. Ví dụ: giả sử một nhà thiết kế không biết về hàm thư viện C++ toascii và thiết kế một tạo phẩm mã từ đầu để đọc một ký tự do người dùng nhập vào và trả về mã ASCII tương ứng (một số nguyên từ 0 đến 127). Một cách để thiết kế điều này là sử dụng nhánh 128 chiều được triển khai bằng câu lệnh chuyển đổi . Cách thứ hai là có một mảng chứa 128 ký tự theo thứ tự mã ASCII và sử dụng một vòng lặp để so sánh ký tự do người dùng nhập với từng phần tử của mảng ký tự; vòng lặp được thoát khi thu được kết quả khớp. Giá trị hiện tại của biến vòng lặp khi đó là mã ASCII tương ứng. Hai thiết kế tương đương về chức năng nhưng có độ phức tạp theo chu kỳ lần lượt là 128 và 1.

Khi mô hình cổ điển được sử dụng, một lớp đo lường liên quan cho giai đoạn thiết kế dựa trên việc biểu diễn thiết kế kiến trúc dưới dạng biểu đồ có hướng với các mô-đun được biểu thị bởi các nút và luồng giữa các mô-đun (các lệnh gọi thủ tục và chức năng) được biểu thị bằng các cung. Fan -in của một mô-đun có thể được định nghĩa là số lượng luồng vào mô-đun cộng với số lượng cấu trúc dữ liệu toàn cầu được truy cập bởi mô-đun. Tương tự, fan -out là số lượng luồng ra khỏi mô-đun cộng với số lượng cấu trúc dữ liệu toàn cầu được mô-đun cập nhật. Khi đó, thước đo độ phức tạp của mô-đun được đưa ra bằng chiều dài × (quạt vào × quạt ra)
2 [Henry và Kafura, 1981], trong đó chiều dài là thước đo kích thước của mô-đun (Phần 9.2.1). Do các định nghĩa về phân phối vào và phân xuất ra kết hợp dữ liệu toàn cầu nên số liệu này có thành phần phụ thuộc vào dữ liệu. Tuy nhiên, các thí nghiệm đã chỉ ra rằng thước đo này không phải là thước đo độ phức tạp tốt hơn so với thước đo đơn giản hơn, chẳng hạn như độ phức tạp chu trình [Kitchenham, Pickard và Linkman, 1990; Shepperd, 1990].

Vấn đề về độ đo thiết kế thậm chí còn phức tạp hơn khi mô hình hướng đối tượng được sử dụng. Ví dụ, độ phức tạp chu trình của một lớp thường thấp, bởi vì nhiều lớp thường bao gồm một số lượng lớn các phương thức nhỏ, đơn giản. Hơn nữa, như đã chỉ ra trước đó, độ phức tạp theo chu trình bỏ qua độ phức tạp của dữ liệu. Bởi vì dữ liệu và hoạt động là các đối tác bình đẳng trong mô hình hướng đối tượng, nên độ phức tạp theo chu kỳ bỏ qua một thành phần chính có thể góp phần tạo nên độ phức tạp của một đối tượng. Do đó, các số liệu cho các lớp kết hợp độ phức tạp của chu trình thường ít được sử dụng.

Ví dụ, một số thước đo thiết kế hướng đối tượng đã được đưa ra trong [Chidamber và Kemerer, 1994]. Những số liệu này và các số liệu khác đã được đặt câu hỏi trên cả cơ sở lý thuyết và thực nghiệm [Binkley và Schach, 1996; 1997; 1998].

14.16 Những thách thức của quy trình thiết kế

Như đã chỉ ra trong Phần 12.16 và 13.22, điều quan trọng là không làm quá nhiều trong quy trình phân tích; nghĩa là, nhóm phân tích không được bắt đầu sớm các phần của quy trình thiết kế. Trong quy trình thiết kế, nhóm thiết kế có thể mắc sai lầm theo hai cách: làm quá nhiều và làm quá ít.

Xem xét thiết kế chi tiết PDL (mã giả) của Hình 14.7 . Sự cám dỗ mạnh mẽ đối với một nhà thiết kế thích lập trình để viết thiết kế chi tiết bằng C ++ hoặc Java, thay vì PDL. Nghĩa là, thay vì phác thảo thiết kế chi tiết bằng mã giả, người thiết kế có thể hoàn toàn viết mã cho lớp. Điều này mất nhiều thời gian để viết hơn là chỉ phác thảo lớp và lâu hơn để sửa x nếu một lỗi được phát hiện trong thiết kế (xem Hình 1.6). Giống như nhóm phân tích, các thành viên của nhóm thiết kế phải kiên quyết chống lại sự thôi thúc phải làm nhiều hơn những gì được yêu cầu.

Đồng thời, nhóm thiết kế phải cẩn thận để không làm quá ít. Xem xét thiết kế chi tiết dạng bảng của Hình 14.6. Nếu nhóm thiết kế đang vội, họ có thể quyết định thu nhỏ thiết kế chi tiết thành hộp tường thuật. Nhóm thậm chí có thể quyết định rằng những người lập trình nên tự mình thiết kế chi tiết. Một trong những quyết định này sẽ là một sai lầm. Lý do chính cho thiết kế chi tiết là để đảm bảo rằng tất cả các giao diện đều chính xác.
Bản thân hộp tường thuật là không đủ cho mục đích này; không có thiết kế chi tiết rõ ràng thậm chí còn ít hữu ích hơn. Do đó, một thách thức của quy trình thiết kế là các nhà thiết kế chỉ thực hiện đúng khối lượng công việc.

Ngoài ra, có một thách thức không thể quan trọng hơn nhiều. Trong “No Silver Bullet” (xem Just in Case You Wanted to Know Box 3.4), Brooks [1986] chỉ trích sự thiếu sót của cái mà ông gọi là những nhà thiết kế vĩ đại, nghĩa là một thiết kế của Brooks, sự thành công của một dự án phần mềm phụ thuộc rất nhiều vào việc nhóm thiết kế có được lãnh đạo bởi một nhà thiết kế giỏi hay không. Thiết kế tốt có thể được dạy; thiết kế tuyệt vời chỉ được tạo ra bởi những nhà thiết kế tuyệt vời và chúng “rất hiếm”.

Sau đó, thách thức là phát triển các nhà thiết kế tuyệt vời. Họ nên được xác định càng sớm càng tốt (các nhà thiết kế giỏi nhất không nhất thiết phải là người có kinh nghiệm nhất), chỉ định một người cố vấn, cung cấp một nền giáo dục chính thức cũng như học nghề cho các nhà thiết kế giỏi và được phép tương tác với các nhà thiết kế khác. Một con đường sự nghiệp cụ thể nên có sẵn cho những nhà thiết kế này và phần thưởng họ nhận được phải tương xứng với sự đóng góp mà chỉ một nhà thiết kế vĩ đại mới có thể thực hiện cho một dự án phát triển phần mềm.

chương
Ôn tập

Quy trình thiết kế được giới thiệu trong Phần 14.1. Có ba cách tiếp cận cơ bản để thiết kế: thiết kế hướng hoạt động (Phần 14.2), thiết kế hướng dữ liệu (Phần 14.5) và thiết kế hướng đối tượng (Phần 14.6). Hai trường hợp của thiết kế hướng hoạt động được mô tả, phân tích luồng dữ liệu (Phần 14.3) và phân tích giao dịch (Phần 14.4). Thiết kế hướng đối tượng được áp dụng cho bài toán thang máy nghiên cứu tình huống trong Phần 14.7 và cho bài nghiên cứu tình huống của Tổ chức MSG trong Phần 14.8. Quy trình thiết kế được trình bày trong Phần 14.9. Các khía cạnh thiết kế của quy trình thử nghiệm được mô tả trong Phần 14.10 và được áp dụng cho nghiên cứu tình huống của Tổ chức MSG trong Phần 14.11. Các kỹ thuật chính thức cho thiết kế chi tiết được thảo luận trong Phần 14.12. Thiết kế hệ thống thời gian thực được mô tả trong Phần 14.13. Các công cụ và số liệu CASE cho quy trình thiết kế được trình bày lần lượt trong Phần 14.14 và 14.15. Chương này kết thúc với một cuộc thảo luận về những thách thức của quy trình thiết kế (Phần 14.16).

Tổng quan về tình huống nghiên cứu của Tổ chức MSG cho Chương 14 được thể hiện trong Hình 14.18, và cho vấn đề về thang máy trong Hình 14.19 .

HÌNH 14.18
Tổng quan về Quỹ bột ngọt nghiên cứu tình huống cho Chương 14.

Thiết kế hướng đối tượng	Mục 14.8
Sơ đồ lớp tổng thể	Hình 14.13
Một phần của sơ đồ lớp tổng thể với các định dạng thuộc tính được thêm vào	Hình 14.14
Thiết kế chi tiết	Phụ lục G

HÌNH 14.19 Tổng quan về nghiên cứu tình huống bài toán thang máy cho Chương 14.

Thiết kế hướng đối tượng	Mục 14.7
Sơ đồ lớp chi tiết	Hình 14.11

Vì Hơn nữa Đọc

Phân tích luồng dữ liệu và phân tích giao dịch được mô tả trong các cuốn sách như [Gane và Sarsen, 1979] và [Yourdon và Constantine, 1979].

Số tháng 3-tháng 4 năm 2005 của IEEE Software chứa một số bài viết về thiết kế. Thiết kế để phục hồi, nghĩa là thiết kế phần mềm để phát hiện, phản ứng và phục hồi từ các điều kiện ngoại lệ, được mô tả trong [Wirfs-Brock, 2006].

Briand, Bunse và Daly [2001] thảo luận về khả năng bảo trì của các thiết kế hướng đối tượng. Một so sánh của cả kỹ thuật thiết kế hướng đối tượng và thiết kế cổ điển xuất hiện trong [Fichman và Kemerer, 1992]. Việc thiết kế lại hệ thống kiểm soát không lưu được mô tả trong [Jackson và Chapin, 2000].
Kỹ thuật thiết kế cho hiệu năng cao, hệ thống đáng tin cậy được đưa ra trong [Stolper, 1999]. Một cách tiếp cận xác suất để ước tính xu hướng thay đổi của một thiết kế hướng đối tượng xuất hiện trong [Tsantalis, Chatzigeorgiou, và Stephanides, 2005]. Một cuộc thảo luận về việc liệu thiết kế hướng đối tượng có trực quan xuất hiện trong [Hadar và Leron, 2008].

Các kỹ thuật thiết kế hình thức được mô tả trong [Hoare, 1987]. Vai trò quan trọng của kiến trúc sư được mô tả trong [McBride, 2007]. Tương tự như lập trình cặp, thiết kế cặp và hiệu quả của nó được mô tả trong [Lui, Chan, và Nosek, 2008].

Liên quan đến các đánh giá trong quá trình thiết kế, tài liệu gốc về kiểm tra thiết kế là [Fagan, 1976]; thông tin chi tiết có thể được lấy từ giấy đó. Những tiến bộ sau này trong các kỹ thuật đánh giá được mô tả trong [Fagan, 1986]. Đánh giá về kiến trúc được thảo luận trong [Maranzano et al., 2005].

Đối với thiết kế thời gian thực, các kỹ thuật cụ thể được tìm thấy trong [Liu, 2000] và [Gomaa, 2000]. So sánh bốn kỹ thuật thiết kế thời gian thực được tìm thấy trong [Kelly và Sherif, 1992]. Cách tiếp cận dựa trên tài liệu để thiết kế các hệ thống thời gian thực phức tạp được mô tả trong [Luqi, Zhang, Berzins và Qiao, 2004]. Thiết kế của các hệ thống đồng thời được mô tả trong [Magee và Kramer, 1999].

Các thước đo cho thiết kế được mô tả trong [Henry và Kafura, 1981] và [Zage và Zage, 1993]. Các thước đo cho thiết kế hướng đối tượng được thảo luận trong [Chidamber và Kemerer, 1994] và trong [Binkley và Schach, 1996]. Một mô hình cho chất lượng hướng đối tượng được trình bày trong [Bansiya và Davis, 2002].

Kỷ yếu của các Hội thảo Quốc tế về Đặc tả và Thiết kế Phần mềm là một nguồn thông tin toàn diện về các kỹ thuật thiết kế.

Thuật ngữ chính thiết kế kiểu dữ liệu trừu tượng 476 bộ truy cập 482 kiến trúc sư 486 thiết kế kiến trúc 466 sơ đồ lớp 476 độ phức tạp chu trình 491 phân tích luồng dữ liệu (DFA) 467 thiết kế hướng dữ liệu 465 quy trình thiết kế 483 thiết kế chi tiết 466 quạt vào 491 quạt ra 491 thiết kế chung 466 thiết kế cấp cao 466 chiều dài 491 thiết kế logic 466 thiết kế cấp thấp 466 thiết kế mô-đun 466 bộ biến đổi 482 thiết kế hướng đối tượng (OOD) 476 thiết kế hướng hoạt động 465 gói 486 thiết kế vật lý 466