Chương 9

Lập kế hoạch và Ước tính

Mục tiêu học tập

Sau khi nghiên cứu chương này, bạn sẽ có thể

- Giải thích tầm quan trọng của việc lập kế hoạch.
- Ước tính quy mô và chi phí xây dựng một sản phẩm phần mềm.
- Đánh giá cao tầm quan trọng của việc cập nhật và theo dõi các ước tính.
- Lập kế hoạch quản lý dự án phù hợp với tiêu chuẩn IEEE

Những thách thức khi xây dựng một sản phẩm phần mềm không có giải pháp dễ dàng. Để kết hợp một sản phẩm phần mềm lớn cần có thời gian và nguồn lực. Và, giống như bất kỳ dự án xây dựng lớn nào khác, lập kế hoạch cẩn thận khi bắt đầu dự án có lẽ là yếu tố quan trọng nhất giúp phân biệt thành công và thất bại. Tuy nhiên, kế hoạch ban đầu này không có nghĩa là đủ. Lập kế hoạch, giống như kiểm thử, phải tiếp tục trong suốt quá trình phát triển và bảo trì phần mềm. Bất chấp nhu cầu lập kế hoạch liên tục, các hoạt động này đạt đến đỉnh điểm sau khi các thông số kỹ thuật đã được phác thảo nhưng trước khi các hoạt động thiết kế bắt đầu. Tại thời điểm này trong quá trình này, các ước tính chi phí và thời lượng có ý nghĩa được tính toán và lập kế hoạch chi tiết để hoàn thành dự án.

Trong chương này, chúng ta phân biệt hai loại **quy hoạch này** , quy hoạch tiến hành xuyên suốt dự án và quy hoạch cường độ cao phải được thực hiện sau khi các thông số kỹ thuật hoàn chỉnh.

9.1 Lập kế hoạch và Quy trình Phần mềm

Tốt nhất, chúng tôi muốn lập kế hoạch cho toàn bộ dự án phần mềm ngay từ đầu của quá trình, và sau đó thực hiện theo kế hoạch đó cho đến khi phần mềm mục tiêu cuối cùng đã được giao cho khách hàng. Tuy nhiên, điều này là không thể, vì chúng tôi thiếu đủ thông tin trong quá trình ban đầu

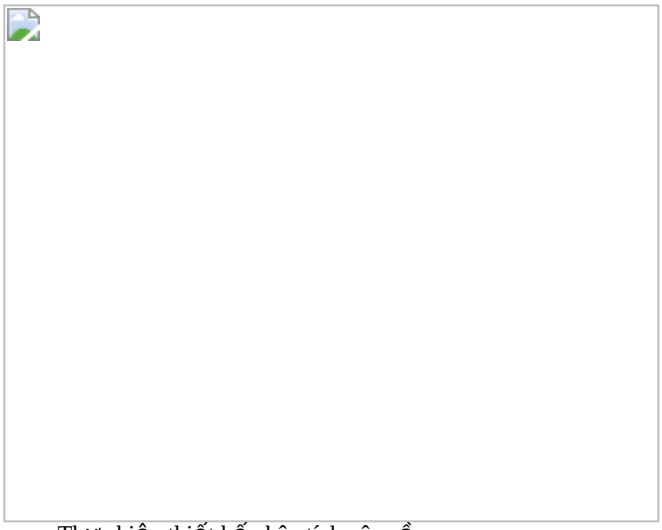
HÌNH 9.1 Mô hình ước tính phạm vi tương đối của ước tính chi phí cho mỗi quy trình làm việc trong vòng đời.

quy trình làm việc để có thể vạch ra một kế hoạch có ý nghĩa cho dự án hoàn chỉnh. Ví dụ, trong quy trình làm việc yêu cầu, bất kỳ loại lập kế hoạch nào (không chỉ dành cho bản thân quy trình yêu cầu) đều vô ích.

Có một thế giới khác biệt giữa thông tin do nhà phát triển sử dụng ở cuối quy trình yêu cầu và ở cuối quy trình phân tích, tương tự như sự khác biệt giữa bản phác thảo thô và bản thiết kế chi tiết. Khi kết thúc quy trình làm việc yêu cầu, các nhà phát triển tốt nhất phải có hiểu biết không chính thức về những gì khách hàng cần. Ngược lại, vào cuối quy trình phân tích, tại thời điểm khách hàng ký một tài liệu nêu chính xác những gì sẽ được xây dựng, các nhà phát triển sẽ đánh giá chi tiết hầu hết (nhưng thường không phải là tất cả) các khía cạnh của sản phẩm mục tiêu. Đây là điểm sớm nhất trong quá trình có thể xác định chính xác khoảng thời gian và ước tính chi phí.

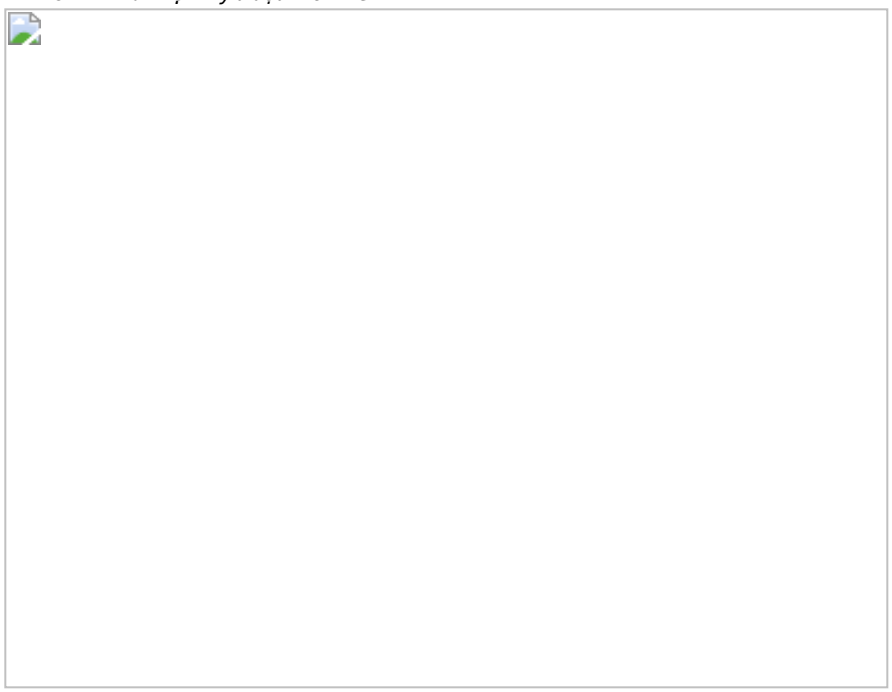
Tuy nhiên, trong một số tình huống, một tổ chức có thể được yêu cầu đưa ra các ước tính về thời lượng và chi phí trước khi các thông số kỹ thuật có thể được đưa ra. Trong trường hợp xấu nhất, khách hàng có thể nhất quyết trả giá trên cơ sở một hoặc hai giờ thảo luận sơ bộ. F hình 9.1 cho thấy cách có vấn đề này có thể được. Dựa trên một mô hình trong [Boehm và cộng sự, 2000], nó mô tả phạm vi ước tính chi phí tương đối cho các quy trình công việc khác nhau của vòng đời. Ví dụ: giả sử rằng, khi một sản phẩm vượt qua kiểm tra chấp nhận ở cuối quy trình triển khai và được giao cho khách hàng, chi phí của nó được tìm thấy là 1 triệu đô la. Nếu một ước tính chi phí đã được thực hiện giữa chừng với quy trình làm việc theo yêu cầu, thì có khả năng nó sẽ nằm ở đâu đó trong phạm vi (0,25 triệu đô la, 4 triệu đô la), như thể hiện trong Hình 9.2. Tương tự, nếu ước tính chi phí được thực hiện giữa chừng trong quá trình phân tích, phạm vi ước tính có thể sẽ giảm xuống (0,5 triệu đô la, 2 triệu đô la). Hơn nữa,

nếu ước tính chi phí đã được thực hiện vào cuối quy trình phân tích, tức là, vào thời điểm thích hợp, kết quả có thể sẽ nằm trong phạm vi tương đối rộng (0,67 triệu đô la, 1,5 triệu đô la). Tất cả bốn điểm được đánh dấu trên các đường giới hạn trên và dưới trong Hình 9.2, có thang đo logarit trên trục tung. Mô hình này được gọi là **hình nón của sự không chắc chắn** . Nó là



Thực hiện thiết kế phân tích yêu cầu

Quy trình làm việc trong đó ước tính chi phí được thực hiện



Rõ ràng từ Hình 9.1 và 9 .2 rằng ước tính chi phí không phải là một khoa học chính xác; lý do cho điều này được nêu trong Phần 9.2.

Dữ liệu dựa trên mô hình hình nón của sự không chắc chắn đã cũ, bao gồm năm đề xuất được đệ trình lên Bộ phận Hệ thống Điện tử của Lực lượng Không quân Hoa Kỳ [Devenny, 1976], và các kỹ thuật ước tính đã được cải thiện kể từ thời điểm đó. Tuy nhiên, hình dạng tổng thể của đường cong trong Hình 9.1 có thể không thay đổi quá nhiều. Do đó, thời hạn hoặc ước tính chi phí quá sớm, tức là ước tính được thực hiện trước khi khách hàng ký duyệt các thông số kỹ thuật, có thể kém chính xác hơn đáng kể so với ước tính được thực hiện khi đã tích lũy đủ dữ liệu.

Bây giờ chúng ta kiểm tra các kỹ thuật để ước tính thời lượng và chi phí. Giả định trong suốt phần còn lại của chương này là quy trình phân tích đã được hoàn thành; nghĩa là bây giờ có thể thực hiện ước tính và lập kế hoạch có ý nghĩa.

9.2 Thời lượng và chi phí ước tính

Ngân sách là một phần không thể thiếu trong bất kỳ kế hoạch quản lý dự án phần mềm nào. Trước khi bắt đầu thiết kế, khách hàng cần biết họ sẽ phải trả bao nhiêu cho sản phẩm. Nếu nhóm phát triển đánh giá thấp chi phí thực tế, tổ chức phát triển có thể mất tiền cho dự án. Mặt khác, nếu nhóm phát triển đánh giá quá cao, thì khách hàng có thể quyết định rằng, dựa trên cơ sở phân tích chi phí - lợi ích hoặc lợi tức đầu tư, việc xây dựng sản phẩm sẽ không có ích lợi gì. Ngoài ra, khách hàng có thể giao công việc cho một tổ chức phát triển khác có ước tính hợp lý hơn. Dù bằng cách nào, rõ ràng là việc ước tính chi phí chính xác là rất quan trọng.

Trên thực tế, có hai loại chi phí liên quan đến việc phát triển phần mềm. Đầu tiên là **chi phí nội bộ** , **chi phí** cho các nhà phát triển; thứ hai là **chi phí bên ngoài** , **mức giá** mà khách hàng sẽ phải trả. Chi phí nội bộ bao gồm tiền lương của các nhóm phát triển, người quản lý và nhân viên hỗ trợ tham gia vào dự án; chi phí của phần cứng và phần mềm để phát triển sản phẩm; và chi phí chung như tiền thuê nhà, tiền điện nước và tiền lương của quản lý cấp cao. Mặc dù giá cả thường dựa trên chi phí cộng với tỷ suất lợi nhuận, nhưng trong một số trường hợp, các yếu tố kinh tế và tâm lý là quan trọng. Ví dụ: các nhà phát triển rất cần công việc có thể chuẩn bị tính phí khách hàng. Một tình huống khác nảy sinh khi hợp đồng được trao trên cơ sở hồ sơ dự thầu. Khách hàng có thể từ chối một giá thầu thấp hơn đáng kể so với tất cả các giá thầu khác với lý do rằng chất lượng của sản phẩm kết quả có thể cũng sẽ thấp hơn đáng kể. Do đó, một nhóm phát triển có thể cố gắng đưa ra một giá thầu sẽ thấp hơn một chút, nhưng không đáng kể, thấp hơn so với giá thầu của các đối thủ cạnh tranh.

Một phần quan trọng khác của bất kỳ kế hoạch nào là ước tính thời gian của dự án. Khách hàng chắc chắn muốn biết khi nào thành phẩm sẽ được giao. Nếu tổ chức phát triển không thể giữ đúng tiến độ của mình, thì tốt nhất là tổ chức mất uy tín, các điều khoản hình phạt tồi tệ nhất sẽ được viện dẫn. Trong mọi trường hợp, các nhà quản lý chịu trách nhiệm về kế hoạch quản lý dự án phần mềm có rất nhiều việc phải giải thích. Ngược lại, nếu tổ chức phát triển đánh giá quá cao thời gian cần thiết để xây dựng sản phẩm, thì rất có thể khách hàng sẽ đi nơi khác.

Thật không may, không có nghĩa là dễ dàng để có được một **ước tính chi phí** và **ước tính thời gian** chính xác . Có quá nhiều biến liên quan để có thể xử lý chính xác chi phí hoặc thời lượng. Một khó khăn lớn là yếu tố con người. Hơn 40 năm trước, Sackman và các đồng nghiệp đã quan sát thấy sự khác biệt lên tới 28 đến 1 giữa các cặp lập trình viên [Sackman, Erikson và Grant, 1968]. Thật dễ dàng để phủ nhận kết quả của họ bằng cách nói rằng các lập trình viên có kinh nghiệm luôn làm tốt hơn những người mới bắt đầu, nhưng Sackman và các đồng nghiệp của ông đã so sánh các cặp lập trình viên phù hợp. Ví dụ, họ đã quan sát hai lập trình viên với 10 năm kinh nghiệm về các loại dự án tương tự và đo thời gian họ thực hiện các tác vụ như viết mã và gỡ lỗi. Sau đó, họ quan sát, chẳng hạn, hai người mới bắt đầu vào nghề trong cùng một khoảng thời gian ngắn và có nền tảng giáo dục tương tự nhau. So sánh hiệu suất kém nhất và tốt nhất, họ đã quan sát thấy sự khác biệt từ 6 đến 1 về kích thước sản phẩm, 8 đến 1 về thời gian thực thi sản phẩm, 9 đến 1 về thời gian phát triển, 18-1 về thời gian mã hóa và 28-1 về thời gian gỡ lỗi. Một quan sát đặc biệt đáng báo động là màn trình diễn tốt nhất và kém nhất trên một sản phẩm là của hai lập trình viên, mỗi người đã có 11 năm kinh nghiệm. Ngay cả khi các trường hợp tốt nhất và xấu nhất được loại bỏ khỏi mẫu của Sackman và cộng sự, sự khác biệt quan sát được vẫn theo thứ tự từ 5 đến 1. Trên cơ sở các kết quả này, rõ ràng, chúng tôi không thể hy vọng ước tính chi phí hoặc thời lượng phần mềm với bất kỳ mức độ chính xác (trừ khi chúng tôi có thông tin chi tiết về tất cả các kỹ năng của tất cả các nhân viên, điều này sẽ là bất thường nhất). Người ta đã lập luận rằng, trong một dự án lớn, sự khác biệt giữa các cá nhân có xu hướng bị loại bỏ, nhưng điều này có lẽ là mơ tưởng; sự hiện diện của một hoặc hai thành viên nhóm rất giỏi (hoặc rất kém) có thể gây ra sự sai lệch rõ rệt so với lịch trình và ảnh hưởng đáng kể đến ngân sách. Một yếu tố con người khác có thể ảnh hưởng đến ước tính là ở một quốc gia tự do, không có cách nào đảm bảo rằng một nhân viên quan trọng sẽ không từ chức trong quá trình dự án. Sau đó, thời gian và tiền bạc được dành để cố gắng lấp đầy vị trí trống và tích hợp người thay thế vào đội, hoặc sắp xếp lại các thành viên còn lại trong đội để bù đắp cho sự mất mát. Dù bằng cách nào, lịch trình trượt và ước tính không ổn định.

1 Phần A *Khái niệm Kỹ thuật Phần mềm*

Cơ bản của vấn đề ước tính chi phí là một vấn đề khác: Kích thước của một sản phẩm được đo lường như thế nào?

9.2.1 Các thước đo về kích thước của một sản phẩm

Số liệu phổ biến nhất cho kích thước của sản phẩm là số dòng mã. Hai đơn vị thường được sử dụng: **dòng mã (LOC)** và **hàng nghìn hướng dẫn nguồn được phân phối (KDSI)** . Nhiều vấn đề liên quan đến việc sử dụng các dòng mã [van der Poel và Schach, 1983].

- Tạo mã nguồn chỉ là một phần nhỏ trong tổng số nỗ lực phát triển phần mềm. Có vẻ hơi xa vời khi thời gian cần thiết cho các quy trình công việc yêu cầu, phân tích, thiết kế, triển khai và thử nghiệm (bao gồm các hoạt động lập kế hoạch và tài liệu) có thể chỉ được biểu thị dưới dạng một hàm của số dòng mã trong sản phẩm cuối cùng.
- Việc triển khai cùng một sản phẩm bằng hai ngôn ngữ khác nhau dẫn đến các phiên bản có số dòng mã khác nhau. Ngoài ra, với các ngôn ngữ như Lisp hoặc với nhiều 4GL phi thủ tục (Phần 15.2), khái niệm về một dòng mã không được xác định.
- Thường không rõ chính xác cách đếm các dòng mã. Chỉ nên đếm các dòng mã thực thi hay định nghĩa dữ liệu? Và những bình luận có nên được tính không? Nếu không, có một nguy cơ là các lập trình viên sẽ miễn cưỡng dành thời gian cho những gì họ cho là các nhận xét "không hiệu quả", nhưng nếu các nhận xét được tính, thì điều nguy hiểm ngược lại là các lập trình viên sẽ viết hàng loạt nhận xét để tăng năng suất rõ ràng. Ngoài ra, những gì về đếm các câu lệnh ngôn ngữ kiểm soát công việc? Một vấn đề khác là cách tính các dòng đã thay đổi hoặc các dòng đã xóa — trong quá trình cải tiến một sản phẩm để cải thiện hiệu suất của nó, đôi khi số lượng dòng mã bị giảm xuống. Việc sử dụng lại mã (Phần 8.1) cũng làm phức tạp việc đếm dòng: Nếu mã được sử dụng lại được sửa đổi, thì nó được tính như thế nào? Và, điều gì sẽ xảy ra nếu mã được kế thừa từ một lớp cha (Phần 7.8)? Nói tóm lại, số liệu rõ ràng đơn giản của các dòng mã là bất kỳ thứ gì ngoại trừ đơn giản để đếm.
- Không phải tất cả các mã được triển khai đều được chuyển đến máy khách. Không có gì lạ khi một nửa mã bao gồm các công cụ cần thiết để hỗ trợ nỗ lực phát triển.
- Giả sử rằng một nhà phát triển phần mềm sử dụng trình tạo mã, chẳng hạn như trình tạo báo cáo, trình tạo màn hình hoặc trình tạo giao diện người dùng đồ họa (GUI). Sau một vài phút hoạt động thiết kế của nhà phát triển, công cụ có thể tạo ra hàng nghìn dòng mã.

- Số lượng dòng mã trong sản phẩm cuối cùng chỉ có thể được xác định khi sản phẩm được hoàn thiện hoàn toàn. Do đó, việc ước tính chi phí dựa trên các dòng mã sẽ nguy hiểm gấp đôi. Để bắt đầu quá trình ước tính, số lượng dòng mã trong thành phẩm phải được ước tính. Sau đó, ước tính này được sử dụng để ước tính giá thành của sản phẩm. Không chỉ có sự không chắc chắn trong mọi kỹ thuật tính chi phí, mà nếu đầu vào của bản thân công cụ ước tính chi phí không chắc chắn là không chắc chắn (tức là số dòng mã trong một sản phẩm chưa được tạo ra), thì độ tin cậy của chi phí kết quả ước tính không chắc là rất cao.

Bởi vì số lượng dòng mã không đáng tin cậy, các chỉ số khác phải được xem xét. Một cách tiếp cận thay thế để ước tính kích thước của sản phẩm là sử dụng các chỉ số dựa trên

các đại lượng đo lường có thể được xác định sớm trong quá trình phần mềm. Ví dụ, van der Poel và Schach [1983] đưa ra các **số liệu FFP** xây dựng dự toán chi phí trung bình - sản phẩm quy mô xử lý dữ liệu. Ba yếu tố cấu trúc cơ bản của một sản phẩm xử lý dữ liệu là fi les, dòng chảy và quy trình của nó; tên gọi FFP là một từ viết tắt được hình thành từ các chữ cái đầu tiên của các yếu tố đó. Một *fi le* được định nghĩa là một tập hợp các hồ sơ một cách logic hay vật lý liên quan đến vĩnh viễn thường trú trong sản phẩm; giao dịch và tài khoản tạm thời bị loại trừ. Một *dòng chảy* là một giao diện dữ liệu giữa các sản phẩm và môi trường, chẳng hạn như một màn hình hoặc báo cáo. Một *quá trình* là một thao tác logic hoặc số học chức năng xác định của dữ liệu; các ví dụ bao gồm sắp xếp, xác thực hoặc cập nhật. Với số lượng fi les *Fi* , lưu lượng *Fl* và xử lý *Pr* trong một sản phẩm, kích thước *S* và chi phí *C của* nó được cho bởi

S *Fi Fl Pr* **(9.1)** *C d S* **(9.2)**

trong đó *d* là hằng số thay đổi giữa các tổ chức. Hằng số *d* là thước đo **hiệu quả** (**năng suất**) của quá trình phát triển phần mềm trong tổ chức đó. Kích thước của một sản phẩm đơn giản là tổng số lượng fi les, dòng chảy và quy trình, một đại lượng có thể được xác định sau khi thiết kế kiến trúc hoàn thành. Khi đó, chi phí tỷ lệ thuận với quy mô, hằng số tỷ lệ *d* được xác định bằng bình phương nhỏ nhất đối với dữ liệu chi phí liên quan đến các sản phẩm do tổ chức đó phát triển trước đó. Không giống như các số liệu dựa trên số lượng dòng mã, chi phí có thể được ước tính trước khi bắt đầu mã hóa.

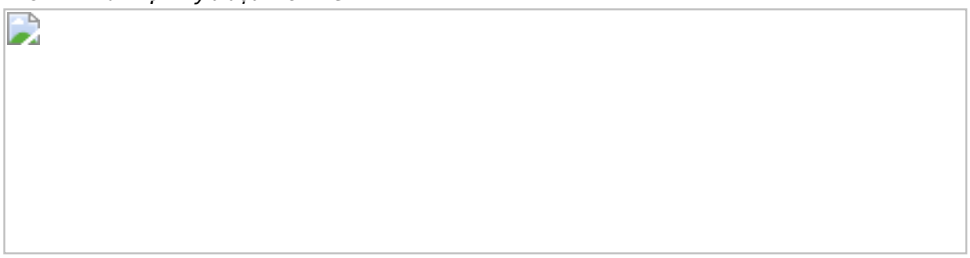
Tính hợp lệ và độ tin cậy của số liệu FFP đã được chứng minh bằng cách sử dụng một mẫu có chủ đích bao gồm một loạt các ứng dụng xử lý dữ liệu quy mô trung bình. Thật không may, số liệu không bao giờ được mở rộng để bao gồm cơ sở dữ liệu, một thành phần thiết yếu của nhiều dữ liệu - sản phẩm chế biến.

Một thước đo tương tự, nhưng được phát triển độc lập, cho kích thước của một sản phẩm được phát triển bởi Albrecht [1979] dựa trên các điểm chức năng; Số liệu của Albrecht dựa trên số lượng mục đầu vào *Inp* , mục đầu ra *Out* , yêu cầu *Inq* , master fi les *Maf* và giao diện *Inf* . Ở dạng đơn giản nhất, số **điểm chức năng** *FP* được cho bởi phương trình

FP 4 *Inp* 5 *Out* 4 *Inq* 10 *Maf* 7 *Inf* **(9.3)**

B ecause này là thước đo kích thước của sản phẩm, nó có thể được sử dụng cho các dự toán chi phí và ước tính năng suất.
Công thức (9.3) là đơn giản hóa của phép tính ba bước. Đầu tiên, các điểm chức năng chưa được điều chỉnh được tính:

1. Mỗi thành phần của sản *phẩm— Inp* , *Out* , *Inq* , *Maf* và *Inf*— phải được phân loại là đơn giản, trung bình hoặc phức tạp (xem Hình 9.3).
2. Mỗi thành phần được gán một số điểm chức năng tùy thuộc vào cấp độ của nó. Ví dụ, một đầu vào trung bình được gán bốn điểm chức năng, như được phản ánh trong phương trình (9.3), nhưng một đầu vào đơn giản chỉ được gán ba, trong khi đầu vào phức tạp được gán sáu điểm chức năng. Dữ liệu cần thiết cho bước này xuất hiện trong Hình 9.3.
3. Các điểm chức năng được gán cho mỗi thành phần sau đó được tổng hợp lại, tạo ra các **điểm chức năng chưa được điều chỉnh (*UFP*)** .



HÌNH 9.3
Bảng giá trị điểm chức năng.

HÌNH 9.4
Kỹ thuật 1. Truyền thông dữ liệu

các yếu tố cho 2. Điểm chức năng xử lý dữ liệu phân tán

3. tính toán tiêu chí hiệu suất .
4. phần cứng được sử dụng hiệu quả
5. tỷ lệ giao dịch cao
6. nhập dữ liệu trực tuyến
7. hiệu quả người dùng cuối
8. cập nhật trực tuyến
9. tính toán đơn giản
10. khả năng tái sử dụng

11. dễ dàng cài đặt
12. dễ dàng hoạt động
13.Tính năng
14. Khả năng bảo trì

Thứ hai, hệ số **phức tạp kỹ thuật (*TCF*)** được tính toán. Đây là thước đo ảnh hưởng của 14 yếu tố kỹ thuật, chẳng hạn như tỷ lệ giao dịch cao, tiêu chí hiệu suất (ví dụ: thông lượng hoặc thời gian phản hồi) và cập nhật trực tuyến; tập hợp đầy đủ các yếu tố được thể hiện trong Hình 9.4. Mỗi yếu tố trong số 14 yếu tố này được gán một giá trị từ 0 ("không có hoặc không có ảnh hưởng") đến 5 ("ảnh hưởng mạnh mẽ xuyên suốt"). 14 số kết quả được tổng hợp lại, cho ra tổng mức độ ảnh hưởng (*DI*). Các *TCF* sau đó được đưa ra bởi

TCF 0,65 0,01 *DI* **(9,4)**

Vì *DI* có thể thay đổi từ 0 đến 70, *TCF* thay đổi từ 0,65 đến 1,35. Thứ ba, *FP* , số điểm chức năng, được cung cấp bởi

FP UFP TCF **(9.5)**

Các thử nghiệm để đo lường tỷ lệ năng suất phần mềm đã cho thấy sự phù hợp hơn khi sử dụng các điểm chức năng so với việc sử dụng KDSI. Ví dụ, Jones [1987] đã tuyên bố rằng ông đã quan sát thấy các lỗi

HÌNH 9.5 So sánh giữa sản phẩm lắp ráp và Ada [Jones, 1987].
(© 1987 IEEE.)

	Phiên bản Assembler	**Phiên bản Ada**
Kích thước mã nguồn	70 KDSI	25 KDSI
Chi phí phát triển	$ 1,043,000	$ 590,000
KDSI mỗi người-tháng	0,335	0,211
Chi phí cho mỗi báo cáo nguồn	$ 14,90	$ 23,60
Điểm chức năng mỗi người-tháng	1,65	2,92
Chi phí cho mỗi điểm chức năng	$ 3,023	$ 1,170

vượt quá 800 phần trăm khi đếm KDSI, nhưng *chỉ* [nhấn mạnh thêm] 200 phần trăm trong việc đếm các điểm chức năng, một nhận xét tiết lộ nhất.

Để cho thấy sự vượt trội của các điểm chức năng so với các dòng mã, Jones [1987] trích dẫn ví dụ được chỉ ra trong Hình 9.5. Cùng một sản phẩm được mã hóa cả trong trình lắp ráp và Ada và các kết quả được so sánh. Đầu tiên, hãy xem xét KDSI mỗi người-tháng. Số liệu này cho chúng ta biết rằng mã hóa trong trình hợp dịch rõ ràng là hiệu quả hơn 60% so với mã hóa trong Ada, điều này hoàn toàn sai. Các ngôn ngữ thế hệ thứ ba như Ada đã thay thế trình hợp dịch đơn giản vì nó hiệu quả hơn nhiều khi viết mã bằng ngôn ngữ thế hệ thứ ba. Bây giờ, hãy xem xét chỉ số thứ hai, chi phí cho mỗi báo cáo nguồn. Lưu ý rằng một câu lệnh Ada trong sản phẩm này tương đương với 2,8 câu lệnh trình hợp dịch. Việc sử dụng báo cáo chi phí trên mỗi nguồn như một thước đo hiệu quả một lần nữa ngụ ý rằng việc viết mã trong trình hợp dịch sẽ hiệu quả hơn trong Ada. Tuy nhiên, khi điểm chức năng trên người / tháng được lấy làm thước đo hiệu quả lập trình, thì sự vượt trội của Ada so với trình hợp dịch được phản ánh rõ ràng.

Mặt khác, cả điểm chức năng và chỉ số FFP của phương trình (9.1) và (9.2) đều có chung một điểm yếu: Bảo trì sản phẩm thường được đo lường không chính xác. Khi một sản phẩm được bảo trì, có thể thực hiện những thay đổi lớn đối với sản phẩm mà không làm thay đổi số lượng fi les, luồng và quy trình hoặc số lượng đầu vào, đầu ra, yêu cầu, master fi les và giao diện. Các dòng mã không tốt hơn về mặt này. Trong trường hợp cực đoan, có thể thay thế mọi dòng của sản phẩm bằng một dòng hoàn toàn khác mà không làm thay đổi tổng số dòng mã.

Ít nhất 40 biến thể và phần mở rộng cho các điểm chức năng của Albrecht đã được đề xuất [Maxwell và Forselius, 2000]. Các điểm chức năng Mk II đã được Symons đưa ra [1991] để cung cấp một cách chính xác hơn để tính toán các điểm chức năng chưa được điều chỉnh (*UFP*). Phần mềm được phân tách thành một tập hợp các giao dịch thành phần, mỗi giao dịch bao gồm một đầu vào, một quá trình và một đầu ra. Giá trị của *UFP* sau đó được tính toán từ các đầu vào, quy trình và đầu ra này. Điểm chức năng Mk II được sử dụng rộng rãi trên toàn thế giới.

9.2.2 Các kỹ thuật ước tính chi phí

Bất chấp những khó khăn trong việc ước tính kích thước, điều quan trọng là các nhà phát triển phần mềm phải cố gắng hết sức có thể để có được ước tính chính xác về cả thời gian dự án và chi phí dự án, đồng thời tính đến càng nhiều càng tốt các yếu tố có thể ảnh hưởng đến ước tính của họ. Chúng bao gồm trình độ kỹ năng của nhân viên, mức độ phức tạp của dự án, quy mô của dự án (chi phí tăng theo quy mô nhưng nhiều hơn so với tuyến tính), mức độ quen thuộc của nhóm phát triển với lĩnh vực ứng dụng, phần cứng mà sản phẩm sử dụng. là

chạy và tính khả dụng của các công cụ CASE. Một yếu tố khác là hiệu ứng thời hạn. Nếu một dự án phải được hoàn thành vào một thời gian nhất định, thì nỗ lực tính bằng tháng người lớn hơn nếu không có ràng buộc nào về thời gian hoàn thành; do đó, chi phí càng lớn. Điều này cho thấy rằng thời lượng và chi phí không độc lập; thời hạn càng ngắn, nỗ lực càng lớn và do đó, chi phí càng lớn.

Từ danh sách trước, không có nghĩa là toàn diện, ước tính rõ ràng là một vấn đề khó khăn. Một số cách tiếp cận đã được sử dụng, với thành công lớn hơn hoặc ít hơn.

1. Phán đoán chuyên môn bằng phép tương tự

Trong phần **đánh giá chuyên môn bằng** kỹ thuật loại suy, một số chuyên gia được tham khảo ý kiến. Một chuyên gia đưa ra ước tính bằng cách so sánh sản phẩm mục tiêu với các sản phẩm đã hoàn thành mà chuyên gia đã tham gia tích cực và lưu ý những điểm giống và khác nhau. Ví dụ: một chuyên gia có thể so sánh sản phẩm mục tiêu với một sản phẩm tương tự được phát triển cách đây 2 năm mà dữ liệu được nhập ở chế độ hàng loạt, trong khi sản phẩm mục tiêu phải được thu thập dữ liệu trực tuyến. Vì tổ chức đã quen với loại sản phẩm sẽ được phát triển, chuyên gia giảm thời gian và nỗ lực phát triển đi 15 phần trăm. Tuy nhiên, giao diện người dùng đồ họa hơi phức tạp; điều này làm tăng thời gian và nỗ lực lên 25 phần trăm. Cuối cùng, sản phẩm mục tiêu phải được phát triển bằng một ngôn ngữ mà hầu hết các thành viên trong nhóm không quen thuộc, do đó tăng thời gian lên 15 phần trăm và nỗ lực lên 20 phần trăm. Kết hợp ba số liệu này, chuyên gia quyết định rằng sản phẩm mục tiêu sẽ tốn nhiều thời gian hơn 25% và nỗ lực hơn 30% so với sản phẩm trước đó. Vì sản phẩm trước đó mất 12 tháng để hoàn thành và yêu cầu 100 người-tháng, sản phẩm mục tiêu ước tính mất 15 tháng và tiêu thụ 130 người-tháng.

T wo các chuyên gia khác trong tổ chức so sánh hai sản phẩm giống nhau. Một người kết luận rằng sản phẩm mục tiêu sẽ mất 13,5 tháng và 140 tháng. Người còn lại đưa ra các số liệu của 16 tháng và 95 người-tháng. Làm thế nào để có thể dung hòa các dự đoán của ba chuyên gia này? Một kỹ thuật là kỹ thuật **Delphi** : Nó cho phép các chuyên gia đi đến thống nhất mà không cần họp nhóm, điều này có thể gây ra tác dụng phụ không mong muốn là một thành viên thuyết phục làm lung lay nhóm. Trong kỹ thuật này, các chuyên gia làm việc độc lập. Mỗi ước tính tạo ra một ước tính và cơ sở lý luận cho ước tính đó. Sau đó, những ước tính và hợp lý này được phân phối cho tất cả các chuyên gia, những người hiện đưa ra ước tính thứ hai. Quá trình ước tính và phân phối này tiếp tục cho đến khi các chuyên gia có thể đồng ý trong phạm vi dung sai được chấp nhận. Không có cuộc họp nhóm nào diễn ra trong quá trình lặp lại.

Định giá bất động sản thường được thực hiện trên cơ sở đánh giá của chuyên gia bằng phép loại suy. Một thẩm định viên đến định giá bằng cách so sánh một ngôi nhà với những ngôi nhà tương tự đã được bán gần đây. Giả sử rằng căn nhà A được định giá, căn nhà B bên cạnh vừa được bán với giá $ 205,000, và căn nhà C trên đường tiếp theo được bán cách đây 3 tháng với giá $ 218,000. Người thẩm định có thể suy luận như sau: Nhà A có nhiều phòng tắm hơn nhà B, và sân rộng hơn 5000 feet vuông. Nhà C có diện tích tương đương với nhà A, nhưng mái của nó kém. Mặt khác, nhà C có một bể sục. Sau khi suy nghĩ kỹ lưỡng, nhà thẩm định có thể đưa ra con số $ 215,000 cho căn nhà A.

Trong trường hợp sản phẩm phần mềm, đánh giá của chuyên gia bằng cách loại suy kém chính xác hơn so với định giá bất động sản. Hãy nhớ lại rằng chuyên gia phần mềm đầu tiên của chúng tôi đã tuyên bố rằng việc sử dụng một ngôn ngữ không quen thuộc sẽ tăng thời gian lên 15 phần trăm và nỗ lực lên 20 phần trăm. Trừ khi chuyên gia có

một số dữ liệu đã được xác thực mà từ đó có thể xác định được ảnh hưởng của mỗi chênh lệch (khả năng rất khó xảy ra), các sai sót gây ra bởi những gì chỉ có thể được mô tả là phỏng đoán sẽ dẫn đến ước tính chi phí không chính xác một cách vô vọng. Ngoài ra, trừ khi các chuyên gia may mắn được thu hồi toàn bộ (hoặc đã lưu giữ hồ sơ chi tiết), việc thu hồi các sản phẩm đã hoàn thành của họ có thể không đủ chính xác để làm mất hiệu lực dự đoán của họ. Cuối cùng, các chuyên gia là con người và do đó, có những thành kiến có thể ảnh hưởng đến dự đoán của họ. Đồng thời, kết quả đánh giá của một nhóm chuyên gia cần phản ánh kinh nghiệm của tập thể họ; nếu điều này đủ rộng, thì kết quả cũng có thể chính xác.

2. Phương pháp tiếp cận từ dưới lên

Một cách để cố gắng giảm thiểu các lỗi do đánh giá tổng thể một sản phẩm là chia sản phẩm thành các thành phần nhỏ hơn. Ước tính về thời lượng và chi phí được thực hiện cho từng thành phần riêng biệt và được kết hợp để cung cấp một con số tổng thể. Phương pháp **từ dưới lên** này có ưu điểm là ước tính chi phí cho một số thành phần nhỏ hơn thường nhanh hơn và chính xác hơn so với một thành phần lớn. Ngoài ra, quá trình ước tính có thể sẽ chi tiết hơn so với một sản phẩm lớn, nguyên khối. Điểm yếu của phương pháp này là một sản phẩm nhiều hơn tổng các thành phần của nó.

Với mô hình hướng đối tượng, sự độc lập của các lớp khác nhau giúp cho cách tiếp cận từ dưới lên. Tuy nhiên, sự tương tác giữa các đối tượng khác nhau trong sản phẩm làm phức tạp quá trình ước tính.

3. Mô hình ước tính chi phí theo thuật toán

Tôi n phương pháp này, một thước đo, chẳng hạn như các điểm chức năng hoặc các số liệu FFP, được sử dụng như đầu vào cho một mô hình để xác định giá thành sản phẩm. Công cụ ước tính tính toán giá trị của số liệu; ước tính thời lượng và chi phí sau đó có thể được tính toán bằng cách sử dụng mô hình. Nhìn bề ngoài, **mô hình ước tính chi phí theo thuật toán** vượt trội hơn so với ý kiến của chuyên gia, vì một chuyên gia về con người, như đã chỉ ra trước đây, có thể bị sai lệch và có thể bỏ qua một số khía cạnh của cả sản phẩm hoàn thành và sản phẩm mục tiêu. Ngược lại, một mô hình ước tính chi phí theo thuật toán là không thiên vị; mọi sản phẩm đều được xử lý theo cùng một cách. Điều nguy hiểm với một mô hình như vậy là các ước tính của nó chỉ tốt như các giả định cơ bản. Ví dụ, cơ bản của mô hình điểm chức năng là giả định rằng mọi khía cạnh của sản phẩm được thể hiện trong năm đại lượng ở phía bên phải của phương trình (9.3) và 14 yếu tố kỹ thuật. Một vấn đề nữa là thường cần một lượng đáng kể đánh giá chủ quan để quyết định những giá trị nào cần gán cho các tham số của mô hình. Ví dụ, thường thì không rõ một hệ số kỹ thuật cụ thể của mô hình điểm chức năng nên được xếp hạng 3 hay 4.

Nhiều mô hình ước tính chi phí theo thuật toán đã được đề xuất. Một số dựa trên các lý thuyết toán học về cách phần mềm được phát triển. Các mô hình khác dựa trên thống kê; số lượng lớn các dự án được nghiên cứu và xác định các quy tắc thực nghiệm từ dữ liệu. Mô hình kết hợp kết hợp các phương trình toán học, mô hình thống kê và đánh giá của chuyên gia . Mô hình lai quan trọng nhất là COCOMO của Boehm, được mô tả chi tiết trong Phần 9.2.3. (Xem Chỉ trong trường hợp bạn muốn biết Hộp 9.1 để thảo luận về từ viết tắt COCOMO.)

Chỉ trong trường hợp bạn muốn biết Hộp 9.1

COCOMO là một từ viết tắt được hình thành từ hai chữ cái đầu tiên của mỗi từ trong COnstructive COst MOdel. Bất kỳ mối liên hệ nào với Kokomo, Indiana, hoàn toàn là ngẫu nhiên.

Các *MO* trong COCOMO là viết tắt của "mô hình", vì vậy cụm từ *mô hình COCOMO* không nên được sử dụng. Cụm từ đó thuộc cùng một loại với "máy ATM" và "số PIN", cả hai đều được Sở Thông tin Dự phòng đưa ra.

9.2.3 COCOMO trung gian

COCOMO thực sự là một chuỗi ba mô hình, từ mô hình ước tính vĩ mô xử lý sản phẩm tổng thể đến mô hình ước tính vi mô xử lý sản phẩm một cách chi tiết. Trong phần này, mô tả được đưa ra về COCOMO trung gian, có mức độ phức tạp và chi tiết ở mức trung bình. COCOMO được mô tả chi tiết trong [Boehm, 1981]; tổng quan được trình bày trong [Boehm, 1984].

Thời gian phát triển máy tính sử dụng COCOMO trung gian được thực hiện trong hai giai đoạn. Đầu tiên, một ước tính sơ bộ về nỗ lực phát triển được cung cấp. Hai tham số phải được ước tính: độ dài của sản phẩm trong KDSI và chế độ phát triển của sản phẩm, một thước đo mức độ khó nội tại của việc phát triển sản phẩm đó. Có ba chế độ: *hữu cơ* (nhỏ và đơn giản), *semidetached* (kích thước trung bình) và *nhúng* (phức tạp).

Từ hai tham số này, có thể tính được **nỗ lực danh nghĩa** . Ví dụ, nếu dự án được đánh giá là đơn giản về cơ bản (hữu cơ), thì nỗ lực danh nghĩa (tính theo tháng) được đưa ra bởi phương trình

$$\text{Nỗ lực danh nghĩa } 3{,}2\ (\text{KDSI})^{1{,}05} \text{ người-tháng} \quad \textbf{(9,6)}$$

Các hằng số 3.2 và 1.05 là các giá trị phù hợp nhất với dữ liệu về các sản phẩm chế độ hữu cơ được Boehm sử dụng để phát triển COCOMO trung gian.

F hoặc ví dụ, nếu sản phẩm được xây dựng là hữu cơ và được ước tính là 12.000 báo cáo nguồn được phân phối (12 KDSI), thì nỗ lực danh nghĩa là

$$3{,}2\ (12)^{1{,}05} \approx 43 \text{ người-tháng}$$

(nhưng hãy đọc Chỉ trong trường hợp bạn muốn biết Hộp 9.2 để biết nhận xét về giá trị này).

Tiếp theo, giá trị danh nghĩa này phải được nhân với 15 **hệ số nhân nỗ lực phát triển phần mềm** . Các số nhân này và giá trị của chúng được cho trong Hình 9.6. Mỗi hệ số có thể có tối đa sáu giá trị. Ví dụ: hệ số phức tạp của sản phẩm được gán các giá trị 0,70, 0,85, 1,00, 1,15, 1,30 hoặc 1,65, tùy theo việc các nhà phát triển đánh giá độ phức tạp của dự án là rất thấp, thấp, danh nghĩa (trung bình), cao, rất cao hay cực cao. Như có thể thấy từ Hình 9.6 , tất cả 15 số nhân đều nhận giá trị 1,00 khi tham số tương ứng là danh nghĩa.

Boehm cung cấp các hướng dẫn để giúp nhà phát triển xác định xem thông số có thực sự nên được xếp hạng danh nghĩa hay xếp hạng thấp hơn hay cao hơn. Ví dụ, hãy xem xét lại hệ số phức tạp của mô-đun. Nếu các hoạt động điều khiển của mô-đun về cơ bản bao gồm một chuỗi các cấu trúc của lập trình có cấu trúc (chẳng hạn như **if-then-else** , **do-while** , **case**), thì độ phức tạp được đánh giá là *rất thấp* . Nếu các toán tử này được lồng vào nhau, thì xếp hạng sẽ *thấp* . Thêm bảng quyết định và điều khiển liên mô-đun sẽ tăng xếp hạng lên *danh nghĩa* . Nếu các toán tử được lồng vào nhau cao, với các vị từ ghép, hàng đợi và ngăn xếp, thì xếp hạng sẽ *cao.* Sự hiện diện của mã hóa quay lại và đệ quy và

Chỉ trong trường hợp bạn muốn biết Hộp 9.2

Một phản ứng đối với giá trị của nỗ lực danh nghĩa có thể là, "Nếu cần nỗ lực 43 tháng người để tạo ra 12.000 hướng dẫn nguồn được phân phối, thì trung bình mỗi lập trình viên tạo ra ít hơn 300 dòng mã mỗi tháng - tôi đã triển khai nhiều hơn hơn thế trong một đêm! "

Một sản phẩm 300 dòng thường chỉ là: 300 dòng mã. Ngược lại, một sản phẩm 12.000 dòng có thể bảo trì phải trải qua tất cả các quy trình làm việc của vòng đời. Nói cách khác, tổng công sức của 43 tháng người được chia cho nhiều hoạt động, bao gồm cả viết mã.

HÌNH 9.6 Hệ số nhân nỗ lực phát triển phần mềm COCOMO trung gian [Boehm, 1984]. (© 1984 IEEE)

Trình điều khiển chi phí	Xếp hạng					
	Rất thấp	Thấp	Trên danh nghĩa	Cao	Rất cao	Cực cao
Thuộc tính sản phẩm						
Độ tin cậy cần thiết của phần mềm	0,75	0,88	1,00	1,15	1,40	
Kích thước cơ sở dữ liệu		0,94	1,00	1,08	1.16	
Độ phức tạp của sản phẩm	0,70	0,85	1,00	1,15	1,30	1,65
Thuộc tính máy tính						
Giới hạn thời gian thực hiện			1,00	1.11	1,30	1,66
Hạn chế lưu trữ chính			1,00	1,06	1,21	1.56

Sự biến động của máy ảo *		0,87	1,00	1,15	1,30
Thời gian quay vòng máy tính		0,87	1,00	1,07	1,15
Thuộc tính nhân sự					
Năng lực của nhà phân tích	1,46	1.19	1,00	0,86	0,71
Ứng dụng trải nghiệm	1,29	1.13	1,00	0,91	0,82
Khả năng lập trình viên	1,42	1.17	1,00	0,86	0,70
Trải nghiệm máy ảo *	1,21	1.10	1,00	0,90	
Kinh nghiệm ngôn ngữ lập trình	1,14	1,07	1,00	0,95	
Thuộc tính dự án					
Sử dụng các phương pháp lập trình hiện đại	1,24	1.10	1,00	0,91	0,82
Sử dụng các công cụ phần mềm	1,24	1.10	1,00	0,91	0,83
Lịch trình phát triển bắt buộc	1,23	1,08	1,00	1,04	1.10

* Đối với một sản phẩm phần mềm nhất định, máy ảo bên dưới là một tổ hợp phần cứng và phần mềm (hệ điều hành, hệ quản trị cơ sở dữ liệu) mà nó sử dụng để hoàn thành nhiệm vụ của nó.

xử lý ngắt ưu tiên cố định đẩy xếp hạng lên *rất cao* . Cuối cùng, lập lịch nhiều tài nguyên với các mức độ ưu tiên thay đổi động và kiểm soát mức vi mã đảm bảo rằng xếp hạng sẽ *rất cao*. Các xếp hạng này áp dụng cho các hoạt động kiểm soát. Một mô-đun cũng phải được đánh giá từ quan điểm của các hoạt động tính toán, hoạt động phụ thuộc vào thiết bị và hoạt động quản lý dữ liệu. Để biết chi tiết về các tiêu chí tính toán từng hệ số trong số 15 số nhân, hãy tham khảo [Boehm, 1981].

Để xem điều này hoạt động như thế nào, Boehm [1984] đưa ra ví dụ về phần mềm xử lý truyền thông dựa trên bộ vi xử lý cho một mạng chuyển tiền điện tử mới có độ tin cậy cao, với các yêu cầu về hiệu suất, lịch phát triển và giao diện. Sản phẩm này phù hợp với mô tả của chế độ nhúng và ước tính có độ dài 10.000 hướng dẫn nguồn được phân phối (10 KDSI), vì vậy nỗ lực phát triển danh nghĩa được đưa ra bởi

$$\text{Nỗ lực danh nghĩa } 2{,}8\ (\text{KDSI})^{1{,}20} \quad \textbf{(9,7)}$$

HÌNH 9.7
Trung gian COCOMO xếp hạng hệ số nỗ lực cho phần mềm truyền thông vi xử lý [Boehm, 1984]. (© 1984 IEEE)

Hệ số xếp hạng tình huống trình điều khiển chi phí	Cố gắng
Độ tin cậy yêu cầu của phần mềm Hậu quả nghiêm trọng về tài chính	Cao 1. 15 lỗi phần mềm
Kích thước cơ sở dữ liệu 20.000 byte	Thấp 0,94
Độ phức tạp của sản phẩm Xử lý truyền thông	Rất cao 1,30
Ràng buộc thời gian thực thi Sẽ sử dụng 70% thời gian khả dụng	Cao 1,11
Hạn chế lưu trữ chính 45K của 64K cửa hàng (70%)	Cao 1,06
Tính biến động của máy ảo Dựa trên danh nghĩa thương mại phần cứng vi xử lý	1.00
Thời gian quay vòng của máy tính Thời gian quay vòng trung bình 2 giờ	Danh nghĩa 1,00
Năng lực của nhà phân tích Các nhà phân tích cấp cao giỏi	Cao 0,86
Ứng dụng có kinh nghiệm 3 năm	Danh nghĩa 1,00
Khả năng lập trình viên Tốt lập trình viên cao cấp	Cao 0,86
Trải nghiệm máy ảo 6 tháng	Thấp 1,10
Kinh nghiệm ngôn ngữ lập trình 12 tháng	Danh nghĩa 1,00
Sử dụng chương trình hiện đại Hầu hết các kỹ thuật được sử dụng trên	Cao 0,91 thực hành 1 năm
Sử dụng các công cụ phần mềm ở máy tính mini cơ bản mức công cụ	Thấp 1,10
Lịch trình phát triển cần thiết 9 tháng	Danh nghĩa 1,00

(Một lần nữa , các hằng số 2,8 và 1,20 là những giá trị phù hợp nhất với fi t ted các dữ liệu trên các sản phẩm nhúng.) Bởi vì dự án được ước tính là 10 KDSI dài, nỗ lực trên danh nghĩa là

$$2{,}8\ (10)^{1{,}20} \quad 44 \text{ người-tháng}$$

Các nỗ lực phát triển ước tính thu được bằng cách nhân các nỗ lực trên danh nghĩa của 15 phần mềm nhân nỗ lực phát triển. Xếp hạng của các số nhân này và giá trị của chúng được cho trong Hình 9.7. Sử dụng các giá trị này, tích của các số nhân được tìm thấy là
1,35, do đó, nỗ lực ước tính cho dự án là

$$1{,}35\ 44\ 59 \text{ người-tháng}$$

Con số này sau đó được sử dụng trong các công thức bổ sung để xác định chi phí đô la, lịch trình phát triển, phân phối giai đoạn và hoạt động, chi phí máy tính, chi phí bảo trì hàng năm và các hạng mục liên quan khác; để biết thêm chi tiết, xem [Boehm, 1981]. COCOMO trung gian là một mô hình ước tính chi phí theo thuật toán hoàn chỉnh, cung cấp cho người dùng hầu như mọi hỗ trợ có thể hình dung được trong việc lập kế hoạch dự án.
COCOMO trung gian đã được xác nhận đối với một mẫu rộng gồm 63 dự án bao gồm nhiều lĩnh vực ứng dụng khác nhau. Kết quả của việc áp dụng COCOMO trung gian cho mẫu này là các giá trị thực tế nằm trong khoảng 20% giá trị dự đoán khoảng 68% thời gian. Những nỗ lực để cải thiện độ chính xác này không có ý nghĩa gì vì trong hầu hết các tổ chức, dữ liệu đầu vào cho COCOMO trung gian thường chỉ chính xác trong khoảng 20%. Tuy nhiên, độ chính xác thu được bởi các nhà ước tính có kinh nghiệm đã đặt COCOMO trung gian vào vị trí tiên tiến của nghiên cứu ước tính chi phí trong suốt những năm 1980; không có kỹ thuật nào khác chính xác nhất quán.

Vấn đề lớn với COCOMO trung gian là đầu vào quan trọng nhất của nó là số dòng mã trong sản phẩm mục tiêu. Nếu ước tính này không chính xác, thì mọi dự đoán của mô hình có thể không chính xác. Do có khả năng các dự đoán của COCOMO trung gian hoặc bất kỳ kỹ thuật ước lượng nào khác có thể không chính xác, ban quản lý phải theo dõi tất cả các dự đoán trong suốt quá trình phát triển phần mềm.

9.2.4 COCOMO II

C OCOMO được đưa ra vào năm 1981. Vào thời điểm đó, mô hình vòng đời duy nhất được sử dụng là mô hình thác nước. Hầu hết phần mềm được chạy trên máy tính lớn. Về cơ bản, các công nghệ như máy khách-máy chủ và hướng đối tượng chưa được biết đến. Theo đó, COCOMO đã không kết hợp bất kỳ yếu tố nào trong số này. Tuy nhiên, khi các công nghệ mới hơn bắt đầu trở thành thực hành kỹ thuật phần mềm được chấp nhận, COCOMO bắt đầu trở nên kém chính xác hơn.

COCOMO II [Boehm và cộng sự, 2000] là một bản sửa đổi lớn của COCOMO năm 1981.
COCOMO II có thể xử lý nhiều loại kỹ thuật kỹ thuật phần mềm hiện đại, bao gồm hướng đối tượng, các mô hình vòng đời khác nhau được mô tả trong Chương 2, tạo mẫu nhanh (Phần 11.13), ngôn ngữ thế hệ thứ tư (Phần 15.2), tái sử dụng (Phần 8.1) , và phần mềm COTS (Phần 1.11). COCOMO II vừa linh hoạt vừa tinh vi. Thật không may, để đạt được mục tiêu này, COCOMO II phức tạp hơn đáng kể so với COCOMO ban đầu. Do đó, độc giả muốn sử dụng COCOMO II nên nghiên cứu chi tiết [Boehm et al., 2000]; ở đây chỉ đưa ra tổng quan về sự khác biệt chính giữa COCOMO II và COCOMO trung gian.

Đầu tiên, COCOMO trung gian bao gồm một mô hình tổng thể dựa trên các dòng mã (KDSI). Mặt khác, COCOMO II bao gồm ba mô hình khác nhau. Các **mô hình thành phần ứng dụng** , dựa trên điểm đối tượng (tương tự như điểm chức năng), được áp dụng ở các quy trình công việc sớm nhất, khi kiến thức tối thiểu có sẵn liên quan đến sản phẩm được xây dựng. Sau đó, khi có nhiều kiến thức hơn, **mô hình thiết kế ban đầu** được sử dụng; mô hình này dựa trên các điểm chức năng. Cuối cùng, khi các nhà phát triển có thông tin tối đa, **mô hình kiến trúc bài đăng** được sử dụng. Mô hình này sử dụng các điểm chức năng hoặc các dòng mã (KDSI). Đầu ra từ COCOMO trung gian là ước tính chi phí và quy mô; đầu ra từ mỗi mô hình trong số ba mô hình của COCOMO II là một loạt các ước tính về chi phí và kích thước. Theo đó, nếu ước tính có khả năng nhất của nỗ lực là E , thì mô hình thành phần ứng dụng trả về phạm vi (0,50 E , 2,0 E) và mô hình kiến trúc bài trả về phạm vi (0,80 E , 1,25 E). Điều này phản ánh mức độ chính xác ngày càng tăng của các mô hình COCOMO II. Điểm khác biệt thứ hai nằm ở mô hình nỗ lực cơ bản của COCOMO:

$$\text{Nỗ lực} = a\,(\text{kích thước})^{b} \quad \textbf{(9,8)}$$

trong đó a và b là hằng số. Trong COCOMO trung gian, số mũ b nhận ba giá trị khác nhau, tùy thuộc vào chế độ của sản phẩm được xây dựng là hữu cơ (b = 1,05), bán kỳ (b = 1,12) hay nhúng (b = 1,20). Trong COCOMO II, giá trị của b thay đổi trong khoảng từ 1,01 đến 1,26, tùy thuộc vào nhiều tham số của mô hình. Chúng bao gồm sự quen thuộc với các sản phẩm thuộc loại đó, mức độ thành thục của quy trình (Phần 3.13), mức độ giải quyết rủi ro (Phần 2.7) và mức độ hợp tác nhóm (Phần 4.1).
Sự khác biệt thứ ba là giả định liên quan đến việc tái sử dụng. COCOMO trung gian giả định rằng số tiền tiết kiệm được do tái sử dụng tỷ lệ thuận với số lượng tái sử dụng. COCOMO II tính đến rằng những thay đổi nhỏ đối với phần mềm được sử dụng lại sẽ phát sinh chi phí lớn không tương xứng (vì mã đã được hiểu chi tiết cho dù chỉ một thay đổi nhỏ và chi phí thử nghiệm một mô-đun sửa đổi là tương đối lớn).

Thứ tư, hiện có 17 trình điều khiển chi phí nhân, thay vì 15 trong COCOMO trung gian. Bảy trong số các yếu tố chi phí là yếu tố mới, chẳng hạn như khả năng tái sử dụng bắt buộc trong các sản phẩm trong tương lai, luân chuyển nhân sự hàng năm và liệu sản phẩm có đang được phát triển tại nhiều địa điểm hay không. C OCOMO II đã được hiệu chỉnh bằng cách sử dụng 83 dự án từ nhiều lĩnh vực khác nhau. Mô hình vẫn còn quá mới để có nhiều kết quả về độ chính xác của nó và đặc biệt là mức độ cải tiến của nó so với người tiền nhiệm của nó, bản gốc (1981)
COCOMO.

9.2.5 Thời lượng theo dõi và ước tính chi phí

Trong khi sản phẩm đang được phát triển, nỗ lực phát triển thực tế phải liên tục được so sánh với các dự đoán. Ví dụ: giả sử rằng số liệu ước tính được các nhà phát triển phần mềm sử dụng dự đoán rằng **thời gian** của quy trình phân tích sẽ kéo dài 3 tháng và yêu cầu 7 tháng người lao động. Tuy nhiên, 4 tháng đã trôi qua và 10 tháng công sức đã được tiêu tốn, nhưng các thông số kỹ thuật vẫn chưa hoàn thiện. Những sai lệch kiểu này có thể coi là một cảnh báo sớm rằng có điều gì đó không ổn và cần phải thực hiện hành động sửa chữa. Vấn đề có thể là do kích thước của sản phẩm bị đánh giá thấp nghiêm trọng hoặc nhóm phát triển không đủ năng lực như người ta vẫn tưởng. Bất kể lý do là gì, sẽ có thời gian và chi phí vượt quá nghiêm trọng, và ban giám đốc phải có hành động thích hợp để giảm thiểu ảnh hưởng.

Việc theo dõi cẩn thận các dự đoán phải được thực hiện trong suốt quá trình phát triển, bất kể các kỹ thuật mà các dự đoán đã được thực hiện. Sai lệch có thể do các chỉ số dự đoán kém, phát triển phần mềm không hiệu quả, kết hợp cả hai hoặc một số lý do khác. Điều quan trọng là phát hiện sớm những sai lệch và có hành động sửa chữa ngay lập tức. Ngoài ra, điều cần thiết là phải liên tục cập nhật các dự đoán dưới ánh sáng của thông tin bổ sung khi nó có sẵn.

Vì các số liệu để ước tính thời lượng và chi phí đã được thảo luận, các thành phần của kế hoạch quản lý dự án phần mềm được mô tả.

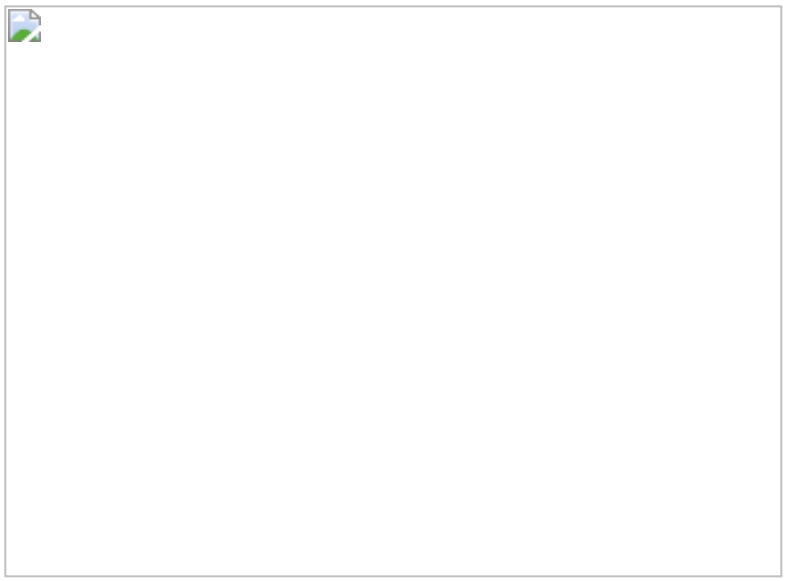
9.3 Các thành phần của Kế hoạch Quản lý Dự án Phần mềm

Một kế hoạch quản lý dự án phần mềm có ba thành phần chính: công việc phải thực hiện, nguồn lực để thực hiện nó và tiền để trả cho tất cả. Trong phần này, ba thành phần của kế hoạch sẽ được thảo luận. Thuật ngữ này được lấy từ [IEEE 1058, 1998], được thảo luận chi tiết hơn trong Phần 9.4.

Phát triển phần mềm yêu cầu *tài nguyên* . Các **nguồn lực** chính cần có là những người sẽ phát triển phần mềm, phần cứng chạy phần mềm và phần mềm hỗ trợ như hệ điều hành, trình soạn thảo văn bản và phần mềm điều khiển phiên bản (Phần 5.9). Các nguồn lực như nhân sự thay đổi theo thời gian. Norden [1958] đã chỉ ra rằng đối với các dự án lớn, **phân phối Rayleigh** là một xấp xỉ tốt của cách mà mức tiêu thụ tài nguyên, R_c, thay đổi theo thời gian, t , nghĩa là,

$$R_c = 2k^2 t e^{-t^2/2k^2} \quad (9.9)$$

Tham số k là hằng số, thời gian tiêu thụ ở mức cao nhất và e = 2.71828 . . ., cơ số của logarit Naperian (tự nhiên). Một đường cong Rayleigh điển hình được thể hiện trong Hình 9.8.



Tiêu thụ tài nguyên bắt đầu nhỏ, tăng nhanh đến đỉnh điểm, sau đó giảm với tốc độ chậm hơn. Putnam [1978] đã nghiên cứu khả năng ứng dụng các kết quả của Norden vào việc phát triển phần mềm và nhận thấy rằng nhân sự và các nguồn tài nguyên khác đã được mô hình hóa với độ chính xác ở một mức độ nào đó bằng phân phối Rayleigh.

Do đó, tôi không đủ trong một kế hoạch phần mềm chỉ đơn thuần nói rằng yêu cầu ba lập trình viên cao cấp với ít nhất 5 năm kinh nghiệm. Những gì cần thiết là một cái gì đó như sau:

> Cần ba lập trình viên cao cấp với ít nhất 5 năm kinh nghiệm trong lập trình thời gian thực, hai người để bắt đầu 3 tháng sau khi dự án bắt đầu, người thứ ba bắt đầu 6 tháng sau đó. Hai sẽ bị loại bỏ dần khi bắt đầu thử nghiệm sản phẩm, thứ ba khi bắt đầu bảo trì giao hàng sau.

Thực tế là nhu cầu nguồn lực phụ thuộc vào thời gian không chỉ áp dụng cho nhân sự mà còn áp dụng cho thời gian sử dụng máy tính, phần mềm hỗ trợ, phần cứng máy tính, tiện nghi văn phòng và thậm chí cả việc đi lại. Do đó, kế hoạch quản lý dự án phần mềm là một hàm của thời gian.

Công việc phải làm được chia thành hai loại. Đầu tiên là công việc tiếp tục trong suốt dự án và không liên quan đến bất kỳ quy trình phát triển phần mềm cụ thể nào. Công việc như vậy được gọi là một **chức năng của dự án** . Ví dụ như quản lý dự án và kiểm soát chất lượng. Thứ hai là công việc liên quan đến quy trình làm việc cụ thể trong quá trình phát triển sản phẩm; công việc

đó được gọi là một *hoạt động* hoặc một *nhiệm vụ* . Một **hoạt động** là một đơn vị công việc chính có ngày bắt đầu và ngày kết thúc chính xác; tiêu tốn tài nguyên, chẳng hạn như thời gian sử dụng máy tính hoặc số ngày của con người; và kết quả là **các sản phẩm công việc** , chẳng hạn như ngân sách, tài liệu thiết kế, lịch trình, mã nguồn hoặc hướng dẫn sử dụng. Ngược lại, một hoạt động bao gồm một tập hợp các nhiệm vụ, một **nhiệm vụ** là đơn vị công việc nhỏ nhất chịu trách nhiệm giải trình của cấp quản lý. Do đó, có ba loại công việc trong một kế hoạch quản lý dự án phần mềm: các chức năng dự án được thực hiện trong suốt dự án, các hoạt động (các đơn vị công việc chính) và các nhiệm vụ (các đơn vị công việc nhỏ).

Một khía cạnh quan trọng của kế hoạch liên quan đến việc hoàn thành các sản phẩm công việc. Ngày mà một sản phẩm công việc được coi là hoàn thành được gọi là một **mốc quan trọng** . Để xác định liệu một sản phẩm làm việc có thực sự đạt đến một mốc quan trọng hay không, trước tiên nó phải vượt qua một loạt các **bài đánh giá được** thực hiện bởi các thành viên trong nhóm, quản lý hoặc khách hàng. Một cột mốc tiêu biểu là ngày thiết kế được hoàn thành và vượt qua quá trình xem xét. Một khi sản phẩm công việc đã được xem xét và thống nhất, nó sẽ trở thành **đường cơ sở** và chỉ có thể được thay đổi thông qua các thủ tục chính thức, như được mô tả trong Phần 5.10.2.

Tôi thực tế, có nhiều thứ hơn đối với một sản phẩm làm việc chứ không chỉ đơn thuần là bản thân sản phẩm. Một **gói công việc** xác định không chỉ sản phẩm công việc mà còn các yêu cầu về nhân sự, thời gian, nguồn lực, tên của cá nhân chịu trách nhiệm và tiêu chí chấp nhận cho sản phẩm công việc. Tất nhiên, **tiền** là một thành phần quan trọng của kế hoạch. Một ngân sách chi tiết phải được lập ra và phân bổ tiền, như một chức năng của thời gian, cho các chức năng và hoạt động của dự án.

Vấn đề làm thế nào để lập một kế hoạch sản xuất phần mềm sẽ được giải quyết tiếp theo.

9.4 Khung kế hoạch quản lý dự án phần mềm

Có nhiều cách để lập một kế hoạch quản lý dự án. Một trong những tiêu chuẩn tốt nhất là IEEE Standard 1058 [1998]. Các thành phần của kế hoạch được thể hiện trong Hình 9.9.

- Tiêu chuẩn được đưa ra bởi đại diện của nhiều tổ chức lớn liên quan đến phát triển phần mềm. Đầu vào đến từ cả các ngành công nghiệp và các trường đại học, và các thành viên của nhóm làm việc và nhóm đánh giá có nhiều năm kinh nghiệm trong việc lập kế hoạch quản lý dự án. Tiêu chuẩn kết hợp kinh nghiệm này.
- Kế hoạch quản lý dự án IEEE được thiết kế để sử dụng với tất cả các loại sản phẩm phần mềm. Nó không áp đặt một mô hình vòng đời cụ thể hoặc quy định một phương pháp luận cụ thể. Về cơ bản, kế hoạch là một khuôn khổ, nội dung của chúng được điều chỉnh bởi mỗi tổ chức cho một lĩnh vực, nhóm phát triển hoặc kỹ thuật cụ thể.
- Khung kế hoạch quản lý dự án IEEE hỗ trợ cải tiến quy trình. Ví dụ, nhiều phần của khuôn khổ phản ánh các lĩnh vực quy trình chính của CMM (Phần 3.13) như quản lý cấu hình và số liệu.
- Khung kế hoạch quản lý dự án IEEE là lý tưởng cho Quy trình Unifi ed. Ví dụ, một phần của kế hoạch được dành cho việc kiểm soát các yêu cầu và một phần khác để quản lý rủi ro, cả hai khía cạnh trung tâm của Quy trình Unifi ed.

Mặt khác, mặc dù tuyên bố trong Tiêu chuẩn IEEE 1058 [1998] rằng kế hoạch quản lý dự án IEEE có thể áp dụng cho các dự án phần mềm ở mọi quy mô, một số phần không liên quan đến phần mềm quy mô nhỏ. Ví dụ, phần 7.7 của khung kế hoạch có tiêu đề là “Kế hoạch quản lý nhà thầu phụ”, nhưng nó là tất cả nhưng chưa từng thấy đối với các nhà thầu phụ được sử dụng trong các dự án quy mô nhỏ.

Theo đó, bây giờ chúng tôi trình bày khung kế hoạch theo hai cách khác nhau. Đầu tiên, khung đầy đủ được mô tả trong Phần 9.5. Thứ hai, một phiên bản viết tắt của khuôn khổ được sử dụng trong Phụ lục F cho kế hoạch quản lý cho một dự án quy mô nhỏ, nghiên cứu điển hình của Tổ chức MSG.

1 Tổng quan
- 1.1 Tóm tắt dự án
  - 1.1.1 Mục đích, phạm vi và mục tiêu
  - 1.1.2 Các giả định và ràng buộc
  - 1.1.3 Các sản phẩm của dự án
  - 1.1.4 Lịch trình và tóm tắt ngân sách
- 1.2 Sự phát triển của kế hoạch quản lý dự án

2 Tài liệu tham khảo

3 Định nghĩa và từ viết tắt

4 Tổ chức dự án
- 4.1 Giao diện bên ngoài
- 4.2 Cấu trúc bên trong
- 4.3 Vai trò và trách nhiệm

5 Kế hoạch quy trình quản lý
- 5.1 Kế hoạch khởi nghiệp
  - 5.1.1 Kế hoạch ước tính
  - 5.1.2 Kế hoạch nhân sự

HÌNH 9.9 Dự án IEEE khung kế hoạch quản lý.

9.5 Kế hoạch quản lý dự án phần mềm IEEE

Bản thân khung **kế hoạch quản lý dự án phần mềm** (SPMP) của IEEE hiện nay đã được mô tả chi tiết. Các số và tiêu đề trong văn bản tương ứng với các mục trong Hình 9.9. Các thuật ngữ khác nhau được sử dụng đã được định nghĩa trong Phần 9.3.

1 Tổng quan.

1.1 Tóm tắt dự án.

1.1.1 Mục đích, phạm vi và mục tiêu. Một mô tả ngắn gọn được đưa ra về mục đích và phạm vi của sản phẩm phần mềm sẽ được phân phối, cũng như các mục tiêu của dự án. Nhu cầu kinh doanh được bao gồm trong tiểu mục này.

1.1.2 Các giả định và ràng buộc. Bất kỳ giả định nào làm cơ sở cho dự án đều được nêu ở đây, cùng với các ràng buộc, chẳng hạn như ngày giao hàng, ngân sách, tài nguyên và hiện vật sẽ được tái sử dụng.

1.1.3 Các sản phẩm của dự án. Tất cả các mặt hàng sẽ được giao cho khách hàng được liệt kê ở đây, cùng với ngày giao hàng.

1.1.4 Lịch trình và tóm tắt ngân sách. Lịch trình tổng thể được trình bày ở đây, cùng với ngân sách tổng thể.

1.2 Sự phát triển của kế hoạch quản lý dự án. Không có kế hoạch nào có thể được đúc bằng bê tông. Kế hoạch quản lý dự án, giống như bất kỳ kế hoạch nào khác, yêu cầu cập nhật liên tục theo kinh nghiệm và thay đổi trong cả tổ chức khách hàng và tổ chức phát triển phần mềm. Trong phần này, các thủ tục và cơ chế chính thức để thay đổi kế hoạch được mô tả, bao gồm cơ chế đặt bản thân kế hoạch quản lý dự án dưới sự kiểm soát cấu hình.

2 Tài liệu tham khảo. Tất cả các tài liệu tham khảo trong kế hoạch quản lý dự án được liệt kê ở đây.

3 Định nghĩa và từ viết tắt. Thông tin này đảm bảo rằng kế hoạch quản lý dự án sẽ được mọi người hiểu theo cách giống nhau.

4 Tổ chức dự án.

4.1 Các giao diện bên ngoài. Không có dự án nào được xây dựng trong môi trường chân không. Các thành viên dự án phải tương tác với tổ chức khách hàng và các thành viên khác trong tổ chức của chính họ. Ngoài ra, các nhà thầu phụ có thể tham gia vào một dự án lớn. Ranh giới hành chính và quản lý giữa dự án và các đơn vị khác phải được xác định.

4.2 Cấu trúc bên trong. Trong phần này, cấu trúc của chính tổ chức phát triển được mô tả. Ví dụ, nhiều tổ chức phát triển phần mềm được chia thành hai loại nhóm: nhóm phát triển hoạt động trên một dự án duy nhất và nhóm hỗ trợ cung cấp các chức năng hỗ trợ, chẳng hạn như quản lý cấu hình và đảm bảo chất lượng , trên

cơ sở toàn tổ chức. Ranh giới hành chính và quản lý giữa nhóm dự án và nhóm hỗ trợ cũng phải được xác định rõ ràng.

4.3 Vai trò và trách nhiệm. Đối với từng chức năng của dự án, chẳng hạn như đảm bảo chất lượng và đối với từng hoạt động, chẳng hạn như thử nghiệm sản phẩm, cá nhân chịu trách nhiệm phải được xác định.

5 Kế hoạch quy trình quản lý.

5.1 Kế hoạch khởi nghiệp.

5.1.1 Kế hoạch ước tính. Các kỹ thuật được sử dụng để ước tính thời gian và chi phí của dự án được liệt kê ở đây, cũng như cách các ước tính này được theo dõi và, nếu cần, được sửa đổi trong khi dự án đang thực hiện.

5.1.2 Kế hoạch nhân sự. Số lượng và loại nhân sự cần thiết được liệt kê, cùng với thời hạn mà họ cần thiết.

5.1.3 Kế hoạch mua lại nguồn lực. Cách thức có được các tài nguyên cần thiết, bao gồm phần cứng, phần mềm, hợp đồng dịch vụ và dịch vụ quản trị, được đưa ra ở đây.

5.1.4 Kế hoạch đào tạo nhân viên dự án. Tất cả các khóa đào tạo cần thiết để hoàn thành dự án thành công được liệt kê trong tiểu mục này.

5.2 Kế hoạch làm việc.

5.2.1 Hoạt động công việc. Trong tiểu mục này, các hoạt động công việc được chỉ rõ, xuống cấp công việc nếu thích hợp.

5.2.2 Phân bổ lịch trình. Nói chung, các gói công việc phụ thuộc lẫn nhau và phụ thuộc nhiều hơn vào các sự kiện bên ngoài. Ví dụ: quy trình thực hiện tuân theo quy trình thiết kế và thử nghiệm sản phẩm trước. Trong tiểu mục này, các phụ thuộc có liên quan được chỉ định.

5.2.3 Phân bổ nguồn lực. Các nguồn lực khác nhau được liệt kê trước đây được phân bổ cho các chức năng, hoạt động và nhiệm vụ thích hợp của dự án.

5.2.4 Phân bổ ngân sách. Trong tiểu mục này, ngân sách tổng thể được chia nhỏ ở các cấp chức năng, hoạt động và nhiệm vụ của dự án.

5.3 Kế hoạch kiểm soát.

5.3.1 Kế hoạch kiểm soát các yêu cầu. Như được mô tả trong Phần B của cuốn sách này, trong khi một sản phẩm phần mềm đang được phát triển, các yêu cầu thường xuyên thay đổi. Các cơ chế được sử dụng để giám sát và kiểm soát các thay đổi đối với các yêu cầu được đưa ra trong phần này.

5.3.2 Kế hoạch kiểm soát lịch trình. Trong phần phụ này, các cơ chế để đo lường prog-
ress được liệt kê, cùng với mô tả về các hành động sẽ được thực hiện nếu tiến độ thực tế chậm hơn tiến độ dự kiến.

5.3.3 Kế hoạch kiểm soát ngân sách. Điều quan trọng là chi tiêu không được vượt quá số tiền dự trù. Các cơ chế kiểm soát để giám sát khi chi phí thực tế vượt quá chi phí ngân sách, cũng như các hành động cần thực hiện nếu điều này xảy ra, được mô tả trong tiểu mục này.

5.3.4 Kế hoạch kiểm soát chất lượng. Các cách thức đo lường và kiểm soát chất lượng được mô tả trong tiểu mục này.

5.3.5 Kế hoạch báo cáo. Để giám sát các yêu cầu, tiến độ, ngân sách và chất lượng, cần phải có các cơ chế báo cáo. Các cơ chế này được mô tả trong tiểu mục này.

5.3.6 Kế hoạch thu thập số liệu. Như đã giải thích trong Phần 5.5, không thể quản lý quá trình phát triển mà không đo lường các chỉ số liên quan. Các chỉ số sẽ được thu thập được liệt kê trong tiểu mục này.

5.4 Kế hoạch quản lý rủi ro. Rủi ro phải được xác định, ưu tiên, giảm thiểu và theo dõi. Tất cả các khía cạnh của quản lý rủi ro được mô tả trong phần này.

5.5 Kế hoạch kết thúc dự án. Các hành động cần thực hiện sau khi dự án hoàn thành, bao gồm phân công lại nhân viên và lưu trữ hiện vật, được trình bày ở đây.

6 Kế hoạch quy trình kỹ thuật.

6.1 Mô hình quy trình. Trong phần này, mô tả chi tiết được đưa ra về mô hình vòng đời sẽ được sử dụng.

6.2 Phương pháp, công cụ và kỹ thuật. Các phương pháp luận phát triển và ngôn ngữ lập trình sẽ được sử dụng được mô tả ở đây.

6.3 Kế hoạch cơ sở hạ tầng. Các khía cạnh kỹ thuật của phần cứng và phần mềm được mô tả chi tiết trong phần này. Các hạng mục cần được đề cập bao gồm hệ thống máy tính (phần cứng, hệ điều hành, mạng và phần mềm) được sử dụng để phát triển sản phẩm phần mềm, cũng như các hệ thống máy tính mục tiêu mà sản phẩm phần mềm sẽ được chạy và các công cụ CASE sẽ được sử dụng .

6.4 Kế hoạch nghiệm thu sản phẩm. Để đảm bảo rằng sản phẩm phần mềm đã hoàn thành vượt qua kiểm tra chấp nhận của nó, các tiêu chí chấp nhận phải được đưa ra, khách hàng phải đồng ý với các tiêu chí bằng văn bản và sau đó các nhà phát triển phải đảm bảo rằng các tiêu chí này thực sự được đáp ứng. Cách thức mà ba giai đoạn này của quá trình chấp nhận sẽ được thực hiện được mô tả trong phần này.

7 Các kế hoạch quy trình hỗ trợ.

7.1 Kế hoạch quản lý cấu hình. Trong phần này, mô tả chi tiết được đưa ra về các phương tiện mà tất cả các hiện vật được đưa vào quản lý cấu hình .

7.2 Kế hoạch kiểm tra. Kiểm thử, giống như tất cả các khía cạnh khác của phát triển phần mềm, cần lập kế hoạch cẩn thận.

7.3 Kế hoạch tài liệu. Phần mô tả tài liệu các loại, có được giao cho khách hàng khi kết thúc dự án hay không, được bao gồm trong phần này.

7.4 Kế hoạch đảm bảo chất lượng. Phần này bao gồm tất cả các khía cạnh của đảm bảo chất lượng, bao gồm kiểm tra, tiêu chuẩn và đánh giá.

7.5 Kế hoạch đánh giá và đánh giá. Chi tiết về cách thức tiến hành đánh giá được trình bày trong phần này.

7.6 Kế hoạch giải quyết vấn đề. Trong quá trình phát triển một sản phẩm phần mềm, tất cả các vấn đề đều có thể phát sinh. Ví dụ, việc đánh giá thiết kế có thể làm sáng tỏ một lỗi nghiêm trọng trong quy trình phân tích đòi hỏi những thay đổi lớn đối với hầu hết tất cả các hiện vật đã được hoàn thành. Trong phần này, cách các vấn đề như vậy được xử lý được mô tả.

7.7 Kế hoạch quản lý nhà thầu phụ. Phần này có thể áp dụng khi các nhà thầu phụ cung cấp các sản phẩm công việc nhất định. Sau đó, cách tiếp cận để lựa chọn và quản lý nhà thầu phụ sẽ xuất hiện ở đây.

7.8 Kế hoạch cải tiến quy trình. Các chiến lược cải tiến quy trình được bao gồm trong phần này.

8 Các kế hoạch bổ sung. Đối với một số dự án nhất định, các thành phần bổ sung có thể cần xuất hiện trong kế hoạch. Về khuôn khổ IEEE, chúng xuất hiện ở cuối kế hoạch. Các thành phần bổ sung có thể bao gồm kế hoạch bảo mật, kế hoạch an toàn, kế hoạch chuyển đổi dữ liệu, kế hoạch cài đặt và kế hoạch bảo trì sau phân phối của dự án phần mềm.

9.6 Lập kế hoạch Kiểm tra

Một thành phần của SPMP thường bị bỏ qua là **lập kế hoạch kiểm tra** . Giống như mọi hoạt động khác của phát triển phần mềm, kiểm thử phải được lập kế hoạch. SPMP phải bao gồm các nguồn lực để kiểm tra và lịch trình chi tiết phải chỉ ra rõ ràng việc kiểm tra sẽ được thực hiện trong mỗi quy trình làm việc.

Nếu không có một kế hoạch thử nghiệm, một dự án có thể trở nên tồi tệ theo một số cách. Ví dụ, trong quá trình thử nghiệm sản phẩm (Phần 3.7.4), nhóm SQA phải kiểm tra xem mọi khía cạnh của tài liệu đặc điểm kỹ thuật, khi được khách hàng ký tên, đã được triển khai trong sản phẩm đã hoàn thành chưa. Một cách tốt để hỗ trợ nhóm SQA trong nhiệm vụ này là yêu cầu sự phát triển phải được theo dõi (Phần 3.7). Nghĩa là, phải có thể kết nối mỗi câu lệnh trong tài liệu đặc tả với một phần của thiết kế và mỗi phần của thiết kế phải được phản ánh rõ ràng trong mã. Một kỹ thuật để đạt được điều này là đánh số từng câu lệnh trong tài liệu đặc tả và đảm bảo rằng những con số này được phản ánh trong cả thiết kế và mã kết quả. Tuy nhiên, nếu kế hoạch kiểm tra không chỉ rõ rằng điều này phải được thực hiện, thì rất khó có khả năng các tạo tác phân tích, thiết kế và mã sẽ được dán nhãn thích hợp. Do đó, khi việc kiểm tra sản phẩm cuối cùng được thực hiện, nhóm SQA sẽ cực kỳ khó xác định rằng sản phẩm đó là sản phẩm thực hiện đầy đủ các thông số kỹ thuật. Trên thực tế, việc truy xuất nguồn gốc nên bắt đầu từ các yêu cầu; mỗi câu lệnh trong phần tạo tác yêu cầu (hoặc mỗi phần của nguyên mẫu nhanh) phải được kết nối với một phần của phần tạo tác phân tích.

Một khía cạnh mạnh mẽ của việc kiểm tra là danh sách chi tiết các lỗi được phát hiện trong quá trình kiểm tra. Giả sử rằng một nhóm đang kiểm tra các thông số kỹ thuật của một sản phẩm. Như đã giải thích trong Phần 6.2.3, danh sách các lỗi được sử dụng theo hai cách. Đầu tiên, số liệu thống kê lỗi từ cuộc kiểm tra này phải được so sánh với số liệu thống kê lỗi trung bình tích lũy từ các cuộc kiểm tra thông số kỹ thuật trước đó. Sự sai lệch so với các định mức trước đây cho thấy các vấn đề trong dự án. Thứ hai, thống kê lỗi từ việc kiểm tra thông số kỹ thuật hiện tại phải được chuyển sang kiểm tra thiết kế và mã của sản phẩm. Rốt cuộc, nếu có một số lượng lớn các lỗi thuộc một loại cụ thể, có thể không phải tất cả chúng đều được phát hiện trong quá trình kiểm tra các thông số kỹ thuật và việc kiểm tra thiết kế và mã cung cấp thêm cơ hội để xác định bất kỳ lỗi nào còn lại của lỗi này. kiểu. Tuy nhiên, trừ khi kế hoạch thử nghiệm quy định rằng các chi tiết của tất cả các lỗi phải được ghi lại cẩn thận, nếu không nhiệm vụ này sẽ không được thực hiện.

Một cách quan trọng để kiểm tra mô-đun mã được gọi là kiểm thử hộp đen (Phần 15.11), trong đó mã được thực thi với các trường hợp kiểm thử dựa trên các thông số kỹ thuật. Các thành viên của nhóm SQA đọc qua các thông số kỹ thuật và vẽ các trường hợp thử nghiệm để kiểm tra xem mã có tuân theo tài liệu đặc điểm kỹ thuật hay không. Thời điểm tốt nhất để tạo ra các trường hợp thử nghiệm hộp đen là vào cuối quy trình phân tích , khi các chi tiết của tài liệu đặc tả vẫn còn mới trong tâm trí của các thành viên của nhóm SQA đã kiểm tra chúng. Tuy nhiên, trừ khi kế hoạch kiểm tra quy định rõ ràng rằng các trường hợp thử nghiệm hộp đen sẽ được chọn vào thời điểm này, trong tất cả các xác suất, chỉ có một số trường hợp thử nghiệm hộp đen sẽ được ném cùng nhau sau đó. Có nghĩa là, một số ít trường hợp thử nghiệm sẽ được lắp ráp nhanh chóng chỉ khi áp lực bắt đầu tăng từ nhóm lập trình để nhóm SQA phê duyệt các mô-đun của họ để chúng có thể được tích hợp vào toàn bộ sản phẩm. Kết quả là, chất lượng của sản phẩm nói chung bị ảnh hưởng.

Do đó, mọi kế hoạch thử nghiệm phải chỉ rõ thử nghiệm nào sẽ được thực hiện, khi nào nó được thực hiện và cách thức thực hiện. Một kế hoạch kiểm tra như vậy là một phần thiết yếu của phần 7.2 của SPMP. Nếu không có nó, chất lượng của sản phẩm tổng thể chắc chắn sẽ bị ảnh hưởng.

9.7 Lập kế hoạch các dự án hướng đối tượng

Giả sử mô hình cổ điển được sử dụng. Từ quan điểm khái niệm, sản phẩm kết quả thường là một đơn vị lớn, mặc dù nó bao gồm các mô-đun riêng biệt. Ngược lại, việc sử dụng mô hình hướng đối tượng dẫn đến một sản phẩm bao gồm một số

các thành phần độc lập nhỏ hơn, cụ thể là các lớp. Điều này làm cho việc lập kế hoạch trở nên dễ dàng hơn đáng kể, trong đó ước tính chi phí và thời lượng có thể được tính toán dễ dàng và chính xác hơn cho các đơn vị nhỏ hơn. Tất nhiên, các ước tính phải tính đến việc một sản phẩm không chỉ là tổng các bộ phận của nó. Các thành phần riêng biệt không hoàn toàn độc lập; chúng có thể gọi lẫn nhau, và những tác động này không được bỏ qua.

Các kỹ thuật ước tính chi phí và thời lượng được mô tả trong chương này có áp dụng cho mô hình hướng đối tượng không? COCOMO II (Phần 9.2.4) được thiết kế để xử lý công nghệ phần mềm hiện đại, bao gồm hướng đối tượng, nhưng còn các số liệu trước đó như điểm chức năng (Phần 9.2.1) và COCOMO trung gian (Phần 9.2.3) thì sao? Trong trường hợp COCOMO trung gian, cần có những thay đổi nhỏ đối với một số nhân chi phí [Pittman, 1993]. Ngoài ra, các công cụ ước lượng của mô hình cổ điển dường như hoạt động khá tốt trên các dự án hướng đối tượng - với điều kiện là không sử dụng lại. Tái sử dụng đi vào mô hình hướng đối tượng theo hai cách: tái sử dụng các thành phần hiện có trong quá trình phát triển và sản xuất có chủ ý (trong dự án hiện tại) các thành phần sẽ được tái sử dụng trong các sản phẩm trong tương lai. Cả hai hình thức tái sử dụng đều ảnh hưởng đến quá trình lập dự toán. Việc tái sử dụng trong quá trình phát triển rõ ràng làm giảm chi phí và thời gian. Các công thức đã được công bố cho thấy sự tiết kiệm như một chức năng của việc tái sử dụng này [Schach, 1994], nhưng những kết quả này liên quan đến mô hình cổ điển. Hiện tại, không có thông tin về chi phí và thời gian thay đổi như thế nào khi sử dụng lại trong quá trình phát triển sản phẩm hướng đối tượng.

Bây giờ chúng ta chuyển sang mục tiêu tái sử dụng các phần của dự án hiện tại. Có thể mất khoảng ba lần để thiết kế, triển khai, kiểm tra và ghi lại thành phần có thể tái sử dụng như một thành phần không thể sử dụng tương tự [Pittman, 1993]. Các ước tính chi phí và thời gian phải được sửa đổi để kết hợp thêm lao động này, và SPMP nói chung phải được điều chỉnh để kết hợp hiệu quả của nỗ lực tái sử dụng. Do đó, hai hoạt động tái sử dụng hoạt động ngược chiều nhau. Việc tái sử dụng các thành phần hiện có làm giảm nỗ lực tổng thể trong việc phát triển một sản phẩm hướng đối tượng, trong khi việc thiết kế các thành phần để tái sử dụng trong các sản phẩm trong tương lai làm tăng nỗ lực. Người ta mong đợi rằng, về lâu dài, số tiền tiết kiệm được do sử dụng lại các lớp học sẽ lớn hơn chi phí phát triển ban đầu, và đã có một số bằng chứng ủng hộ điều này [Lim, 1994].

9.8 Yêu cầu đào tạo

Khi chủ đề **đào tạo** được nêu ra trong các cuộc thảo luận với khách hàng, câu trả lời phổ biến là, "Chúng tôi không cần phải lo lắng về việc đào tạo cho đến khi sản phẩm hoàn thành, sau đó chúng tôi có thể đào tạo người dùng." Đây là một nhận xét hơi đáng tiếc, ngụ ý rằng nó chỉ có người dùng yêu cầu đào tạo. Trên thực tế, các thành viên của nhóm phát triển cũng có thể cần đào tạo, bắt đầu từ đào tạo về lập kế hoạch và ước lượng phần mềm. Khi các kỹ thuật phát triển phần mềm mới, chẳng hạn như kỹ thuật thiết kế mới hoặc quy trình kiểm thử, được sử dụng, thì mọi thành viên của nhóm sử dụng kỹ thuật mới phải được đào tạo.

Việc giới thiệu mô hình hướng đối tượng có những hệ quả đào tạo chính. Việc giới thiệu các công cụ phần cứng hoặc phần mềm như máy trạm hoặc môi trường tích hợp (xem Phần 15.24.2) cũng cần được đào tạo. Lập trình viên có thể cần được đào tạo về hệ điều hành của máy được sử dụng để phát triển sản phẩm cũng như ngôn ngữ thực thi. Việc đào tạo chuẩn bị tài liệu thường xuyên bị bỏ qua, bằng chứng là chất lượng kém của rất nhiều tài liệu. Các nhà khai thác máy tính chắc chắn yêu cầu một số

loại đào tạo để có thể chạy sản phẩm mới; họ cũng có thể yêu cầu đào tạo bổ sung nếu phần cứng mới được sử dụng.

Việc đào tạo bắt buộc có thể đạt được theo một số cách. Cách dễ nhất và ít gây gián đoạn nhất là đào tạo nội bộ, bởi các nhân viên hoặc chuyên gia tư vấn. Nhiều công ty cung cấp nhiều khóa đào tạo khác nhau, và các trường cao đẳng thường cung cấp các khóa đào tạo vào buổi tối. Các khóa học dựa trên World Wide Web là một lựa chọn thay thế khác.

Khi nhu cầu đào tạo đã được xác định và lập kế hoạch đào tạo, kế hoạch này phải được đưa vào SPMP.

9.9 Tiêu chuẩn tài liệu

Sự phát triển của một sản phẩm phần mềm đi kèm với rất nhiều **tài liệu** . Jones nhận thấy rằng 28 trang tài liệu được tạo trên mỗi 1000 hướng dẫn (KDSI) cho một sản phẩm thương mại nội bộ của IBM có kích thước khoảng 50 KDSI và khoảng 66 trang trên mỗi KDSI cho một sản phẩm phần mềm thương mại có cùng kích thước. Hệ điều hành IMS / 360 Phiên bản 2.3 có kích thước khoảng 166 KDSI và 157 trang tài liệu trên mỗi KDSI đã được tạo ra. Tài liệu thuộc nhiều loại khác nhau, bao gồm lập kế hoạch, kiểm soát, tài chính và kỹ thuật [Jones, 1986a]. Ngoài các loại tài liệu này, bản thân mã nguồn cũng là một dạng tài liệu; nhận xét trong mã tạo thành tài liệu bổ sung.

Một phần đáng kể của nỗ lực phát triển phần mềm được tiếp thu bởi tài liệu. Một cuộc khảo sát trên 63 dự án phát triển và 25 dự án bảo trì sau giao hàng cho thấy cứ 100 giờ dành cho các hoạt động liên quan đến mã thì 150 giờ được dành cho các hoạt động liên quan đến tài liệu [Boehm, 1981]. Đối với sản phẩm TRW lớn, tỷ lệ thời

gian dành cho các hoạt động tài liệu liên quan đến tăng đến 200 giờ mỗi 100 đang - liên quan đến giờ [Boehm et al., 1984].

Các tiêu chuẩn là cần thiết cho mọi loại tài liệu. Ví dụ, tính thống nhất trong tài liệu thiết kế làm giảm sự hiểu lầm giữa các thành viên trong nhóm và hỗ trợ nhóm SQA. Mặc dù nhân viên mới phải được đào tạo về các tiêu chuẩn tài liệu, nhưng không cần đào tạo thêm khi nhân viên hiện tại chuyển từ dự án này sang dự án khác trong tổ chức. Từ quan điểm của bảo trì sau giao hàng, các tiêu chuẩn mã hóa thống nhất hỗ trợ các lập trình viên bảo trì hiểu được mã nguồn. Tiêu chuẩn hóa thậm chí còn quan trọng hơn đối với các hướng dẫn sử dụng, bởi vì chúng phải được đọc bởi nhiều cá nhân, một số ít trong số họ là các chuyên gia máy tính. IEEE đã phát triển một tiêu chuẩn cho hướng dẫn sử dụng (IEEE Standard 1063 cho Tài liệu Người dùng Phần mềm).

Là một phần của quá trình lập kế hoạch, các tiêu chuẩn phải được thiết lập cho tất cả các tài liệu được tạo ra trong quá trình sản xuất phần mềm. Các tiêu chuẩn này được kết hợp trong SPMP. Khi một tiêu chuẩn hiện có được sử dụng, chẳng hạn như Tiêu chuẩn ANSI / IEEE cho Tài liệu Kiểm tra Phần mềm [ANSI / IEEE 829, 1991], tiêu chuẩn được liệt kê trong phần 2 của SPMP (tài liệu tham khảo). Nếu một tiêu chuẩn được viết đặc biệt cho nỗ lực phát triển, thì tiêu chuẩn đó sẽ xuất hiện trong phần 6.2 (phương pháp, công cụ và kỹ thuật).

Tài liệu là một khía cạnh thiết yếu của nỗ lực sản xuất phần mềm. Theo một nghĩa rất thực tế, sản phẩm *là* tài liệu, bởi vì không có tài liệu thì sản phẩm không thể được duy trì. Lập kế hoạch cho nỗ lực tài liệu đến từng chi tiết, và sau đó đảm bảo rằng kế hoạch được tuân thủ, là một thành phần quan trọng của quá trình sản xuất phần mềm thành công.

9.10 CASE Công cụ lập kế hoạch và ước tính

Một số công cụ có sẵn để tự động hóa COCOMO trung gian và COCOMO II. Để tăng tốc độ tính toán khi giá trị của một tham số được sửa đổi, một số triển khai COCOMO trung gian đã được triển khai trong các ngôn ngữ bảng tính như Lotus 1-2-3 và Excel. Để phát triển và cập nhật kế hoạch, một trình xử lý văn bản là điều cần thiết.

Các công cụ thông tin quản lý cũng rất hữu ích cho việc lập kế hoạch. Ví dụ, giả sử rằng một tổ chức phần mềm lớn có 150 lập trình viên. Công cụ lập lịch có thể giúp các nhà lập kế hoạch theo dõi những lập trình viên nào đã được giao cho các nhiệm vụ cụ thể và những công việc nào sẵn sàng cho dự án hiện tại.

Các loại thông tin quản lý tổng quát hơn cũng là cần thiết. Một số công cụ quản lý có sẵn trên thị trường có thể được sử dụng để hỗ trợ quá trình lập kế hoạch và ước tính cũng như để giám sát toàn bộ quá trình phát triển. Chúng bao gồm MacProject và Microsoft Project.

9.11 Kiểm tra Kế hoạch Quản lý Dự án Phần mềm

Như đã chỉ ra ở đầu chương này, một lỗi trong kế hoạch quản lý dự án phần mềm có thể gây ra những ảnh hưởng nghiêm trọng về tài chính cho các nhà phát triển. Điều quan trọng là tổ chức phát triển không được đánh giá quá cao hoặc đánh giá thấp chi phí của dự án hoặc thời gian của nó. Vì lý do này, toàn bộ SPMP phải được nhóm SQA kiểm tra trước khi đưa ra các ước tính cho khách hàng. Cách tốt nhất để kiểm tra kế hoạch là kiểm tra kế hoạch.

Nhóm kiểm tra kế hoạch phải xem xét SPMP một cách chi tiết, đặc biệt chú ý đến ước tính chi phí và thời gian. Để giảm rủi ro hơn nữa, bất kể số liệu được sử dụng là gì, ước tính thời lượng và chi phí phải được tính toán độc lập bởi thành viên của nhóm SQA ngay sau khi các thành viên của nhóm lập kế hoạch xác định ước tính của họ.

Đánh giá chương

Chủ đề chính của chương này là tầm quan trọng của việc lập kế hoạch trong quy trình phần mềm (Phần 9.1). Một thành phần quan trọng của bất kỳ kế hoạch quản lý dự án phần mềm nào là ước tính thời lượng và chi phí (Phần 9.2). Một số chỉ số được đưa ra để ước tính kích thước của sản phẩm, bao gồm cả các điểm chức năng (Phần 9.2.1). Tiếp theo, các thước đo khác nhau để ước tính chi phí được mô tả, đặc biệt là COCOMO trung gian (Phần 9.2.3) và COCOMO II (Phần 9.2.4). Như được mô tả trong Phần 9.2.5, điều cần thiết là phải theo dõi tất cả các ước tính. Ba thành phần chính của kế hoạch quản lý dự án phần mềm — công việc phải thực hiện, nguồn lực để thực hiện nó và số tiền phải trả cho nó — được giải thích trong Phần 9.3. Một SPMP cụ thể, tiêu chuẩn IEEE, được nêu trong Phần 9.4 và được mô tả chi tiết trong Phần 9.5. Tiếp theo, hãy làm theo các phần về kiểm tra lập kế hoạch (Phần 9.6), lập kế hoạch các dự án hướng đối tượng (Phần 9.7), các yêu cầu đào tạo và tiêu chuẩn tài liệu cũng như ý nghĩa của chúng đối với quá trình lập kế hoạch (Phần 9.8 và 9.9). Các công cụ của CASE để lập kế hoạch và ước tính được mô tả trong Phần 9.10. Chương này kết thúc với tài liệu về kiểm thử kế hoạch quản lý dự án phần mềm (Phần 9.11).

Đối với Thêm nữa đọc hiểu

Tác phẩm bốn tập của Weinberg [Weinberg, 1992; Năm 1993; Năm 1994; 1997] cung cấp thông tin chi tiết về nhiều khía cạnh của quản lý phần mềm, cũng như [Bennatan, 2000] và [Reifer, 2000]. Số tháng 9 đến tháng 10 năm 2005 của *IEEE Software* có một số bài báo về quản lý phần mềm, đặc biệt là [Royce, 2005] và [Venugopal, 2005]; có các bài báo bổ sung trong số tháng 5 đến tháng 6 năm 2008. Cách

thành công của các nhà quản lý được giải thích trong [Procaccino và Verner, 2006]. Các cơ chế được các nhà quản lý dự án sử dụng để giám sát và kiểm soát các dự án phát triển phần mềm được thảo luận trong [McBride, 2008]. F hoặc thông tin khác về Tiêu chuẩn IEEE 1058 cho Kế hoạch Quản lý Dự án Phần mềm, bản thân tiêu chuẩn này nên được đọc kỹ [IEEE 1058, 1998]. Sự cần thiết phải lập kế hoạch cẩn thận được mô tả trong [McConnell, 2001].

Tác phẩm kinh điển của Sackman được mô tả trong [Sackman, Erikson, and Grant, 1968]. Một nguồn chi tiết hơn là [Sackman, 1970]. Tác động của kiến thức chuyên môn của lập trình viên đối với lập trình theo cặp được mô tả trong [Arisholm, Gallis, Dybå, và Sjøberg , 2007].

Một phân tích cẩn thận về các điểm chức năng, cũng như các cải tiến được đề xuất, xuất hiện trong [Symons, 1991]. Điểm mạnh và điểm yếu của các điểm chức năng được trình bày trong [Furey và Kitchenham , 1997]. Điểm lớp, một phần mở rộng của điểm chức năng cho các lớp, được giới thiệu trong [Costagliola , Ferrucci , Tortora, và Vitiello , 2005].

Giải thích lý thuyết cho COCOMO trung gian, cùng với đầy đủ chi tiết để thực hiện nó, xuất hiện trong [Boehm, 1981]. COCOMO II được mô tả trong [Boehm và cộng sự, 2000]. Các cách tăng cường dự đoán COCOMO được trình bày trong [Smith, Hale và Parrish, 2001]. Phần mở rộng của COCOMO cho các dòng sản phẩm phần mềm xuất hiện trong [In, Baik , Kim, Yang, và Boehm, 2006].

A riand và Wüst [2001] mô tả cách ước tính nỗ lực phát triển cho các sản phẩm hướng đối tượng. Ước tính cả kích thước và khuyết tật của các sản phẩm phần mềm hướng đối tượng được mô tả trong [Cartwright và Shepperd, 2000].

Dữ liệu năng suất phần mềm cho nhiều loại sản phẩm xử lý dữ liệu kinh doanh được trình bày trong [Maxwell và Forselius , 2000]; đơn vị của năng suất được sử dụng là điểm hàm trên giờ. Các thước đo năng suất khác được thảo luận trong [Kitchenham và Mendes, 2004]. Các lỗi trong nỗ lực phần mềm ước tính được phân tích trong [Jorgensen và Moløkken-Østvold , 2004]. Một phê bình về quy trình nghiên cứu được sử dụng thường xuyên để so sánh các mô hình ước lượng được đưa ra trong [Myrtveit , Stensrud và Shepperd, 2005]. Một probabilist mô hình để dự đoán nỗ lực phát triển phần mềm xuất hiện trong [Pendharkar , Subramanian, và Rodger, 2005]. Phân tích chi phí vượt mức cho các sản phẩm phần mềm được xây dựng với các mô hình vòng đời khác nhau xuất hiện trong [Moløkken-Østvold và Jorgensen, 2005]. Có một yêu cầu hiệu quả workflow có thể có tác động tích cực đến năng suất; điều này được thể hiện trong [Damian và Chisan , 2006]. Tác động của hình nón của độ không đảm bảo đối với ước tính lịch trình được phân tích trong [Little, 2006]. Đánh giá toàn diện về 304 nghiên cứu ước tính chi phí phát triển trên 76 tạp chí được trình bày trong [Jorgensen và Shepperd, 2007]. Phương pháp tiếp cận dựa trên bằng chứng để lựa chọn mô hình ước tính chi phí thích hợp cho một dự án nhất định được mô tả trong [Menzies và Hihn , 2006].

Các vấn đề

9 .1 Tại sao bạn cho rằng một số tổ chức phần mềm hoài nghi coi các *cột mốc* là *cối xay* ? (Gợi ý: Tra nghĩa bóng của *cối xay* trong từ điển.)

9.2 Bạn là kỹ sư phần mềm tại Pretoriuskop Software Developers. Một năm trước, người quản lý của bạn đã thông báo rằng sản phẩm tiếp theo của bạn sẽ bao gồm 8 fi les, 48 flow và 91 process.

(i) Sử dụng thước đo FFP, xác định kích thước của nó.

(ii) Đối với Nhà phát triển phần mềm Pretoriuskop , hằng số *d* trong phương trình (9.2) được xác định là $ 1021. Chỉ số FFP dự đoán chi phí nào?

(iii) Sản phẩm gần đây đã được hoàn thành với chi phí $ 135,200. Điều này cho bạn biết điều gì về năng suất của nhóm phát triển của bạn?

9.3 Một sản phẩm mục tiêu có 8 đầu vào đơn giản, 3 đầu vào trung bình và 11 đầu vào phức tạp. Có 57 đầu ra trung bình, 9 câu hỏi đơn giản, 13 giao diện chính chủ trung bình và 18 giao diện phức tạp. Xác định các điểm chức năng chưa điều chỉnh (*UFP*).

9.4 Nếu tổng mức độ ảnh hưởng đối với sản phẩm của Bài toán 9.3 là 47, hãy xác định số điểm chức năng.

9,5 Tại sao bạn nghĩ rằng, mặc dù nhược điểm của nó, dòng mã (LOC hoặc KDSI) được sử dụng rất rộng rãi như một thước đo kích thước sản phẩm?

9.6 Bạn chịu trách nhiệm phát triển sản phẩm nhúng 62-KDSI trên danh nghĩa ngoại trừ kích thước cơ sở dữ liệu được đánh giá rất cao và việc sử dụng các công cụ phần mềm thấp. Sử dụng COCOMO trung gian, ước tính nỗ lực tính theo tháng của người là bao nhiêu?

9.7 Bạn phụ trách phát triển hai sản phẩm chế độ hữu cơ 31-KDSI. Cả hai đều là danh nghĩa về mọi mặt ngoại trừ sản phẩm P1 có độ phức tạp cực cao và sản phẩm P2 có độ phức tạp cực thấp. Để phát triển sản phẩm, bạn có hai nhóm tùy ý. Nhóm A có năng lực phân tích viên, kinh nghiệm ứng dụng và khả năng lập trình viên rất cao. Đội A cũng có kinh nghiệm máy ảo và kinh nghiệm ngôn ngữ lập trình cao. Đội B được đánh giá rất thấp về cả năm thuộc tính.

(i) Tổng nỗ lực (tính theo tháng) nếu đội A phát triển sản phẩm P1 và đội B phát triển sản phẩm P2 là bao nhiêu?

(ii) Tổng nỗ lực (tính theo tháng) nếu đội B phát triển sản phẩm P1 và đội A phát triển sản phẩm P2 là bao nhiêu?

(iii) Việc phân công cán bộ nào trong hai cách trước đây hợp lý hơn? Trực giác của bạn có được hỗ trợ bởi những dự đoán của COCOMO trung gian không?

9.8 Bạn chịu trách nhiệm phát triển một sản phẩm chế độ hữu cơ 48-KDSI được coi là danh nghĩa về mọi mặt.

(i) Giả sử chi phí là 10.100 đô la mỗi người / tháng, dự án ước tính chi phí là bao nhiêu?

(ii) Toàn bộ nhóm phát triển của bạn từ chức khi bắt đầu dự án. Bạn đủ may mắn để có thể thay thế đội danh nghĩa bằng một đội có kinh nghiệm và năng lực rất cao, nhưng chi phí cho mỗi người / tháng sẽ tăng lên 13.400 đô la. Bạn dự kiến thu được (hoặc mất) bao nhiêu tiền do thay đổi nhân sự?

9.9 Bạn chịu trách nhiệm phát triển phần mềm cho một sản phẩm sử dụng một tập hợp các thuật toán mới được phát triển để tính toán các tuyến đường tiết kiệm chi phí nhất cho một công ty vận tải đường bộ lớn. Sử dụng

COCOMO trung gian, bạn xác định rằng giá thành của sản phẩm sẽ là $ 470,000. Tuy nhiên, để kiểm tra, bạn yêu cầu một thành viên trong nhóm của bạn ước tính nỗ lực bằng cách sử dụng các điểm chức năng. Cô ấy báo cáo rằng chỉ số điểm chức năng dự đoán chi phí là 985.000 đô la, lớn hơn gấp đôi so với dự đoán COCOMO của bạn. Bạn làm gì bây giờ?

9.10 Chứng tỏ rằng phân bố Rayleigh [phương trình (9.9)] đạt giá trị lớn nhất khi $t = k$. Tìm mức tiêu thụ tài nguyên tương ứng.

9.11 Kế hoạch bảo trì sau giao hàng sản phẩm được coi là "thành phần bổ sung" của kế hoạch quản lý dự án phần mềm IEEE. Hãy nhớ rằng mọi sản phẩm tầm thường đều được bảo trì và chi phí bảo trì sau giao hàng trung bình gấp khoảng hai hoặc ba lần chi phí phát triển sản phẩm, làm thế nào điều này có thể hợp lý?

9.12 Tại sao các dự án phát triển phần mềm tạo ra quá nhiều tài liệu?

9.13 (Dự án thời hạn) Hãy xem xét dự án Chocoholics Anonymous được mô tả trong Phụ lục A. Tại sao không thể ước tính chi phí và thời gian hoàn toàn dựa trên thông tin trong Phụ lục A?

9.14 (Các bài đọc về Kỹ thuật phần mềm) Người hướng dẫn của bạn sẽ phân phối các bản sao của [Costagliola , Ferrucci , Tortora, và Vitiello , 2005]. Bạn có bị thuyết phục bởi việc xác thực theo kinh nghiệm của điểm lớp không?

Tài liệu tham khảo

[Albrecht, 1979] AJ A LBRECHT , "Đo lường năng suất phát triển ứng dụng", *Kỷ yếu của Hội nghị chuyên đề phát triển ứng dụng SHARE / GUIDE của IBM* , Monterey, CA, tháng 10 năm 1979, trang 83–92.

[ANSI / IEEE 829, 1991] *Tài liệu Kiểm tra Phần mềm* , ANSI / IEEE 829-1991, Viện Tiêu chuẩn Quốc gia Hoa Kỳ, Viện Kỹ sư Điện và Điện tử, New York, 1991.

[Arisholm , Gallis , Dybå , và Sjøberg , 2007] E. A RISHOLM , H. G ALLIS , T. D YBÅ , VÀ DIK S JØBERG , "Đánh giá lập trình theo cặp liên quan đến độ phức tạp của hệ thống và kiến thức chuyên môn của lập trình viên," *Giao dịch IEEE trên phần mềm Kỹ thuật* **33** (tháng 2 năm 2007), trang 65–86.

[Bennatan , 2000] EM B ENNATAN , *Đúng thời gian trong ngân sách: Thực hành và kỹ thuật quản lý dự án phần mềm* , xuất bản lần thứ 3, John Wiley và Sons, New York, 2000.

[Boehm, 1981] BW B OEHM , *Kinh tế Kỹ thuật Phần mềm*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[Boehm, 1984] BW B OEHM , "Kinh tế kỹ thuật phần mềm", *Giao dịch IEEE về Kỹ thuật phần mềm* **SE-10** (tháng 1 năm 1984), trang 4–21.

[Boehm và cộng sự, 1984] BW B OEHM , MH P ENEDO , ED S TUCKLE , RD Wvania , VÀ AB P YSTER , "Một môi trường phát triển phần mềm để cải thiện năng suất," *IEEE Computer* **17** (tháng 6 năm 1984), trang 30 –44.

[Boehm et al., 2000] BW B OEHM , C. Một BTS , AW B ROWN , S. C HULANI , BK C LARK , E. H OROWITZ , R. M ADACHY , D. R EIFER , VÀ B. S TEECE , *Ước tính chi phí phần mềm với COCOMO II* , Prentice Hall, Upper Saddle River, NJ, 2000.

[Briand và Wust 2001] LC B RIAND VÀ J. W Ust , "Nỗ lực phát triển mô hình hóa trong ObjectOriented Systems Sử dụng Thiết kế Properties," *IEEE giao dịch trên phần mềm Kỹ thuật* **27** (tháng 11 năm 2001), pp. 963-86.

[Cartwright và Shepperd, 2000] M. C ARTWRIGHT VÀ M. S HEPPERD , "Điều tra thực nghiệm về hệ thống phần mềm hướng đối tượng," *Giao dịch IEEE về Kỹ thuật phần mềm* **26** (tháng 8 năm 2000), trang 786–95.

[Costagliola , Ferrucci , Tortora, và Vitiello , 2005] G. C OSTAGLIOLA , F. F ERRUCCI , G. T ORTORA , VÀ G. V ITIELLO , "Điểm lớp: Phương pháp tiếp cận ước tính kích thước của hệ thống hướng đối tượng," *Giao dịch IEEE về Kỹ thuật phần mềm* **31** (tháng 1 năm 2005), trang 52–74.

[Damian và Chisan , 2006] D. D AMIAN VÀ J. C HISAN , “Nghiên cứu thực nghiệm về mối quan hệ phức tạp giữa quy trình kỹ thuật yêu cầu và các quy trình khác dẫn đến phần thưởng trong quản lý năng suất, chất lượng và rủi ro,” *Giao dịch IEEE trên phần mềm Kỹ thuật* **32** (tháng 7 năm 2006), trang 433–53.

[Devenny , 1976] T. D EVENNY , “Một Nghiên cứu Khám phá về Ước tính Chi phí Phần mềm tại Bộ phận Hệ thống Điện tử,” Luận văn số GSM / SM / 765–4, Viện Công nghệ Không quân, Dayton, OH, 1976.

[Furey và Kitchenham , 1997] S. F UREY VÀ B. K ITCHENHAM , “Các điểm chức năng”, *Phần mềm IEEE* **14** (tháng 3 - tháng 4 năm 1997), trang 28–32.

[IEEE 1058, 1998] “Tiêu chuẩn IEEE cho các kế hoạch quản lý dự án phần mềm.” IEEE Std. 1058-1998, Viện Kỹ sư Điện và Điện tử, New York, 1998.

[In, Baik , Kim, Yang, và Boehm, 2006] HP I N , J. B AIK , S. K IM , Y. Y ANG , VÀ B. B OEHM , “Một QualityBased Chi phí ước tính mô hình cho Line Life sản phẩm Chu kỳ, ” *Truyền thông của ACM* **49** (tháng 12 năm 2006), trang 85–88.

[Jones, 1986a] C. J ONES , *Năng suất lập trình*, McGraw-Hill, New York, 1986.

[Jones, 1987] C. J ONES , Thư gửi biên tập viên, *IEEE Computer* **20** (tháng 12 năm 1987), tr. 4.

[Jorgensen và Moløkken-Østvold , 2004] M. J ORGENSEN và K. M OLØKKEN -Ø STVOLD , “Lý do gây ra lỗi ước tính nỗ lực phần mềm: Tác động của vai trò người trả lời, Phương pháp tiếp cận thu thập thông tin và Phương pháp phân tích dữ liệu,” *Giao dịch IEEE về Kỹ thuật phần mềm* **30** (tháng 12 năm 2004), trang 993–1007.

[Jorgensen và Shepperd, 2007] M. J ORGENSEN VÀ M. S HEPPERD , “Đánh giá có hệ thống về Nghiên cứu ước tính chi phí phát triển phần mềm,” *Giao dịch IEEE về Kỹ thuật phần mềm* **32** (tháng 1 năm 2007), trang 33–53.

[Kitchenham và Mendes, 2004] B. K ITCHENHAM VÀ E. M ENDES , “Đo lường năng suất phần mềm bằng cách sử dụng nhiều biện pháp kích thước”, *Giao dịch IEEE về Kỹ thuật phần mềm* **30** (tháng 12 năm 2004), trang 1023–35.

[Lim, 1994] WC L IM , “Ảnh hưởng của việc tái sử dụng đối với chất lượng, năng suất và kinh tế,” *IEEE Software* **11** (tháng 9 năm 1994), trang 23–30.

[Little, 2006] T. L ITTLE , “Ước tính lịch trình và sự không chắc chắn xung quanh hình nón của sự không chắc chắn,” *IEEE Software* **23** (tháng 5 - tháng 6 năm 2006), trang 48–54.

[Maxwell và Forselius , 2000] KD M AXWELL VÀ P. F ORSELIUS , “Năng suất phát triển phần mềm đo điểm chuẩn,” *IEEE Software* **17** (tháng 1 - tháng 2 năm 2000), trang 80–88.

[McBride, 2008] T. M C B RIDE , “Các cơ chế quản lý dự án phát triển phần mềm,” *Tạp chí Hệ thống và Phần mềm* **81** (tháng 12 năm 2008), trang 2386–95.

[McConnell, 2001] S. M C C ONNELL , “Cứu Sins chết người Kế hoạch dự án,” *IEEE Software* **18** (November-December 2001), tr. 5-7.

[Menzies và Hihn , 2006] T. M ENZIES VÀ J. H IHN , “Chi phí Evidence-Based Ước cho BetterQuality phần mềm,” *IEEE Software* **23** (July-August 2006), pp. 64-66.

[Moløkken-Østvold và Jorgensen, 2005] K. M OLØKKEN -Ø STVOLD VÀ M. J ORGENSEN , “So sánh các lần vượt dự án phần mềm — Mô hình phát triển linh hoạt so với tuần tự”, *Giao dịch IEEE về Kỹ thuật phần mềm* **31** (tháng 9 năm 2005), tr. 754–66.

[Myrtveit , Stensrud , and Shepperd, 2005] I. M YRTVEIT , E. S TENSRUD , VÀ M. S HEPPERD , “Độ tin cậy và tính hợp lệ trong các nghiên cứu so sánh của các mô hình dự đoán phần mềm,” *Giao dịch IEEE về Kỹ thuật phần mềm* **31** (tháng 5 năm 2005), trang 380–91.

[Norden, 1958] PV N ORDEN , “Phù hợp với đường cong cho mô hình lập lịch trình nghiên cứu và phát triển ứng dụng”, *Tạp chí Nghiên cứu và Phát triển* **2** *của IBM* (tháng 7 năm 1958), trang 232–48.

[Pendharkar , Subramanian và Rodger, 2005] PC P ENDHARKAR , GH S UBRAMANIAN , VÀ JA R ODGER , “Mô hình xác suất để dự đoán nỗ lực phát triển phần mềm”, *Giao dịch IEEE về Kỹ thuật phần mềm* **31** (tháng 7 năm 2005), trang 615–24 .

[Pittman, 1993] M. P ITTMAN , “Bài học kinh nghiệm trong quản lý phát triển hướng đối tượng,” *IEEE Software* **10** (tháng 1 năm 1993), trang 43–53.

[Procaccino và Verner, 2006] JD P ROCACCINO VÀ JM V ERNER , “Thực tiễn phát triển phần mềm công nghiệp nhanh nhẹn như thế nào?” *Tạp chí Hệ thống và Phần mềm* **79** (tháng 11 năm 2006), trang 1541–51.

[Putnam, 1978] LH P UTNAM , “Giải pháp thực nghiệm chung cho vấn đề ước tính và định cỡ phần mềm vĩ mô”, *Giao dịch IEEE về Kỹ thuật phần mềm* **SE-4** (tháng 7 năm 1978), trang 345–61.

[Reifer , 2000] DJ R EIFER , “Software Management: The Good, the Bad, and the Ugly,” *IEEE Software* **17** (Tháng 3 - Tháng 4 năm 2000), trang 73–75.

[Royce, 2005] W. R OYCE , “Phong cách quản lý phần mềm thành công: Chỉ đạo và cân bằng,” *IEEE Software* **22** (tháng 9 - tháng 10 năm 2005), trang 40–47.

[Sackman , 1970] H. S ACKMAN , *Giải quyết vấn đề con người - máy tính: Đánh giá thực nghiệm về chia sẻ thời gian và xử lý hàng loạt*, Auerbach, Princeton, NJ, 1970.

[Sackman , Erikson, và Grant, 1968] H. S ACKMAN , WJ E RIKSON , VÀ EE G rant , “thăm dò thực nghiệm nghiên cứu so sánh trực tuyến và Offl ine Performance Lập trình,” *Truyền thông của ACM* **11** (tháng 1 năm 1968), tr. 3 –11.

[Schach , 1994] SR S CHACH , “Tác động kinh tế của việc tái sử dụng phần mềm đối với việc bảo trì,” *Tạp chí Bảo trì phần mềm: Nghiên cứu và Thực hành* **6** (tháng 7 đến tháng 8 năm 1994), trang 185–96.

[Smith, Hale và Parrish, 2001] RK S MITH , JE H ALE , AND AS P ARRISH , “Một nghiên cứu thực nghiệm sử dụng các mẫu phân công nhiệm vụ để cải thiện độ chính xác của ước tính nỗ lực phần mềm,” *Giao dịch IEEE về kỹ thuật phần mềm* **27** (tháng 3 năm 2001), trang 264–71.

[Symons, 1991] CR S YMONS , *Định cỡ phần mềm và ước tính: Mk II FPA* , John Wiley và Sons, Chichester, UK, 1991.

[Van der Poel và Schach , 1983] KG VAN DER P OEL VÀ SR S CHACH , "Một Metric Phần mềm cho Chi phí ước tính và Effi ciency Đo lường trong xử lý dữ liệu phát triển hệ thống," *Journal of Systems và phần mềm* **3** (tháng 9 năm 1983), pp. 187–91.

[Venugopal, 2005] C. V ENUGOPAL , "Bộ mục tiêu duy nhất: Mô hình mới cho sự thành công của IT Megaproject," *IEEE Software* **22** (Tháng 9 - Tháng 10 năm 2005), trang 48–53.

[Weinberg, 1992] GM W EINBERG , *Quản lý phần mềm chất lượng: Tư duy hệ thống* , Vol. 1, Nhà Dorset, New York, 1992.

[Weinberg, 1993] GM W EINBERG , *Quản lý phần mềm chất lượng: Đo lường bậc nhất* , Tập. 2, Nhà Dorset, New York, 1993.

[Weinberg, 1994] GM W EINBERG , *Quản lý phần mềm chất lượng: Hành động đồng thời* , Vol. 3, Nhà Dorset, New York, 1994.

[Weinberg, 1997] GM W EINBERG , *Quản lý phần mềm chất lượng: Dự đoán thay đổi* , Vol. 4, Nhà Dorset, New York, 1997.

Trang này cố ý để trống