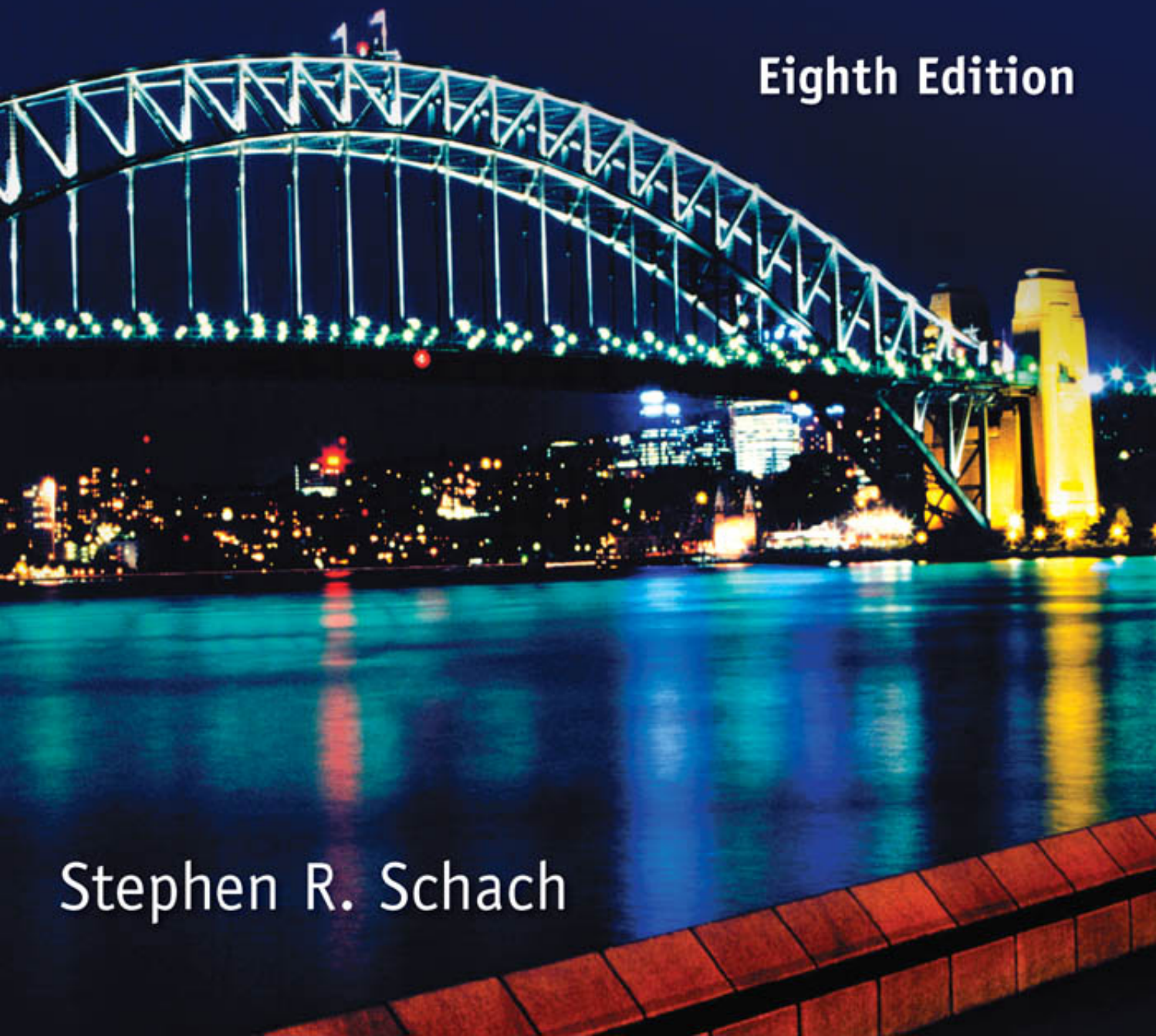
Object-Oriented and Classical Software Engineering

Eighth Edition



Stephen R. Schach

Đối tượng và hướng dẫn Cổ điển phần mềm Kỹ thuật

Octal version

Stephen R. Schach
Đại học Vanderbilt

The McGraw-Hill Companies

KỸ THUẬT PHẦN MỀM CỔ ĐIỂN VÀ ĐỊNH HƯỚNG ĐỐI TƯỢNG, PHÁT TRIỂN TÁI TẠO

This book is in on paper without axit.

1 2 3 4 5 6 7 8 9 0 DOC / DOC 1 0 9 8 7 6 5 4 3 2 1 0

ISBN 978-0-07-337618-9
MHID 0-07-337618-3

Phó chủ tịch Tổng biên tập:*Marty Lange* Bản xuất bản:*Raghothaman Srinivasan*
Chủ tịch EDP & Dịch vụ xuất bản Trung ương:*Kimberly Meriwether David* Biên tập viên phát triển:*Lora Neyens* Cao cấp quản lý:*Curt Reynolds* Project Management:*Melissa M. Leick* Người mua:*Kara Kudronowicz*

Thiết kế viên điều phối:*Brenda A. Rolwes*
Design bìa:*Studio Montage, St. Louis, Missouri* Ảnh bìa: ©*Hình ảnh Photodisc / Getty* Music Editor: *Glyph International* Type:*10/12 Times Roman* Máy trong:*RR Donnelley*

Biên mục dữ liệu của Quốc hội thư viện

Schach, Stephen R.
Kỹ thuật hướng đối tượng phần mềm và cổ điển / Stephen R. Schach. - Xuất bản lần thứ 8.
P. cm.
ISBN-13: 978-0-07-337618-9 (giấy kiềm)
ISBN-10: 0-07-337618-3 (giấy kiềm)
1. Phần mềm kỹ thuật. 2. Lập đối tượng hướng dẫn (Khoa học máy tính) 3. UML (Khoa học máy tính) 4. C ++ (Ngôn ngữ chương trình máy tính) I. Tên bài.

QA76.758.S318 2010
005.1'17 - dc22

2010020995

www.mhhe.com

Gửi Jackson và Mikaela

Sau đây là đăng ký nhãn hiệu:

ADF

Nhà parsing / Nhà thiết kế

Con kiến

Apache

kết quả

AS / 400

AT&T

Bachman bộ sản phẩm

Chuông thí nghiệm

Borland

Bugzilla

Trưởng mô hình Chrome to

about the use

ClearCase

ClearQuest

CMM

Ca cao

Cô-ca Cô-la

CORBA

CppUnit

CVS

DB2

Nhật Bản

Điện tử thành phần

Emeraude

Doanh nghiệp JavaBeans

eServer

Excel

Firefox

Tiêu điểm

Ford

Foundation FoxBASE lớp

thư viện

GCC

hewlett Packard

IBM

IMS / 360

Jackpot Java source code

number

JBuilder

JUnit

Linux

Hoa sen 1-2-3

Lucent Technologies

MacApp

Macintosh

Macintosh tool box

MacProject

Microsoft

Motif

MS-DOS

MVS / 360

Thiên nhiên

Netscape

New York Time

C object

Objective-C

ObjectWindows thư viện

1-800-flowers.com

Oracle

Oracle Developer Suite

OS / 360

OS / 370

OS / VS2

Palm Pilot

Parasoft

Note after it

Quyền xây dựng

Tiếp tục ngữ

TRƯỚC

Dự định

PureCoverage

PVCS

QARun

HỢP LÝ

Tinh tế Pro

use thi ca

Hoa hồng

SBC Communications

SilkTest

NHỰA CÂY

Phần mềm thông qua Hình ảnh

Solaris

SourceSafe

SPARCstation

mặt trời

Sun Enterprise

Sun Microsystems

Sun ONE Studio

Hệ thống kiến trúc

Cùng nhau

UNIX

VAX

Thư viện thành phần trực tiếp

Visual C ++

Visual J ++

VM / 370

VMS

Tạp chí Phố Wall

WebSphere

Win32

Windows 95

Windows 2000

Windows NT

Word

X11

Xrunner

XUnit

Nén đĩa

Chính Code

z10

Content

Lời nói đầu

Hầu hết các chương trình giảng dạy về khoa học máy tính và máy tính kỹ thuật hiện nay đều bao gồm một dự án phát triển phần mềm dựa trên bắt buộc nhóm. Trong một số trường hợp, dự án chỉ kéo dài một học kỳ hoặc một quý, nhưng một dự án phát triển phần mềm dựa trên nhóm kéo dài cả năm đang nhanh chóng trở thành tiêu chuẩn.

Trong một thế giới tưởng tượng, mọi sinh viên sẽ hoàn thành một khóa học về phần mềm kỹ thuật trước khi bắt đầu dự án dựa trên nhóm của mình ("chương trình giảng dạy hai giai đoạn"). Tuy nhiên, trên thực tế, nhiều sinh viên phải bắt đầu dự án của mình một phần trong khóa học kỹ thuật phần mềm của họ, hoặc thậm chí là khi bắt đầu khóa học ("chương trình giảng dạy bài hát").

Như đã được giải thích trong phần tiếp theo, cuốn sách này được sắp xếp theo cách mà nó có thể được sử dụng cho cả chương trình giảng dạy.

Mi octa version group way

Sách bao gồm hai phần chính: Phần B hướng dẫn sinh viên cách phát triển một phần mềm sản phẩm; Phần A cung cấp nền tảng lý thuyết cần thiết cho Phần B. 18 chương trình được sắp xếp như sau:

	Chương 1	Giới thiệu về kỹ thuật phần mềm Các khái niệm kỹ thuật phần mềm Kỹ thuật kỹ thuật phần mềm Các công nghệ mới nổi
Phần A	Chương 2 đến chương 9	
Phần B	Chương 10 đến chương 17	
	Chương 18	

Chương 10 is new. Nó chứa một bản tóm tắt chính tài liệu của Phần A. Khi chương trình giảng dạy hai giai đoạn được chấp hành, giảng viên sẽ dạy Phần A trước rồi đến Phần B (bỏ qua Chương 10, vì tài liệu của Chương 10 sẽ được thực hiện trình bày chuyên sâu trong Phần một). Đối với bài hát chương trình học, giảng viên đầu tiên của Phần B (để sinh viên có thể bắt đầu dự án của mình càng sớm càng tốt), sau đó là Phần A. Tài liệu của Chương 10 giúp sinh viên hiểu Phần B mà không cần học phần A. trước.

This second Cận cảnh có phản hồi trực tiếp: Chắc chắn rằng thuyết phục phải luôn được dạy trước khi thực hiện. Trên thực tế, các vấn đề về chương trình giảng dạy Nhiều giảng viên đã sử dụng ấn bản thứ bảy của cuốn sách này để giảng dạy tài liệu của Phần B trước Phần A. Đáng sợ là họ hài lòng nhất với phần kết quả. Họ báo cáo rằng sinh viên của họ được đánh giá cao hơn tài liệu thuyết trình của Phần A do kết quả của công việc dự án của họ. Đó là, làm việc theo dự án theo nhóm để sinh viên dễ tiếp thu và hiểu các khái niệm lý thuyết làm nền tảng cho phần mềm kỹ thuật.

Chi tiết hơn, tài liệu của ấn bản thứ tám có thể được dạy theo hai cách sau:

1. Giảng dạy chương trình hai giai đoạn

Phần A Phần B	Chương 1 (Giới thiệu về kỹ thuật phần mềm) Chương 2 đến 9 (Khái niệm kỹ thuật phần mềm) Chương 11 đến 17 (Kỹ thuật phần mềm) Chương 18 (Các công nghệ mới nổi) Sau đó, các sinh viên sẽ bắt đầu các dự án dựa trên nhóm của họ trong học tập hoặc quý tiếp theo.

2. Bài hát Giáo trình

	Chương 1 (Giới thiệu về phần mềm kỹ thuật) Chương 10 (Chính tài liệu từ Phần A) Các sinh viên hiện bắt đầu các dự án dựa trên nhóm của họ, song song với Nghiên cứu tài liệu của Phần B.
Phần B	Chương 11 đến 17 (Phần mềm kỹ thuật) Chương 2 đến 9
Phần A	(Khái niệm kỹ thuật phần mềm) Chương 18 (Các công nghệ mới nổi)

New feature of Eighth Edition

- Sách đã được cập nhật trong suốt.
- Tôi đã thêm mới hai chương trình. Như đã được giải thích trước đó, Chương 10, tóm tắt các điểm chính của Phần A, đã được đưa vào cuốn sách này có thể được sử dụng khi sinh viên bắt đầu các dự án học kỳ dựa trên nhóm của bài hát của họ. khóa học phần mềm kỹ thuật của họ. Khác mới chương trình, Chương 18, trình bày tổng quan về 10 công nghệ mới nổi, bao gồm
  - Công nghệ hướng về khía cạnh
  - Mô hình điều khiển công nghệ
  - Công nghệ dựa trên thành phần
  - Hướng dẫn công nghệ tới dịch vụ
  - Hội điện tử
  - Web kỹ thuật
  - Đám mây
  - Web 3.0
  - Máy tính bảo mật
  - Kiểm tra mô hình
- Tôi đã mở rộng đáng giá tài liệu về các thiết kế mẫu trong Chương 8, bao gồm cả một tiểu bang mới.
- Hai công cụ lý thuyết đã được thêm vào Chương 5: phân chia và chinh phục, và phân tách mối quan hệ.
- Phân tích hướng đối tượng về vấn đề máy tính bảng của Chương 13 đã phản hồi một tản nhiệt kiến trúc, phi hiện đại tập trung.
- Tham khảo tài liệu đã được cập nhật rộng rãi, tập trung vào hiện tại nghiên cứu.
- Có hơn 100 vấn đề mới.
- Có mới hộp chỉ trong trường hợp bạn muốn biết.

Các tính năng được giữ lại từ Ấn bản thứ bảy

- The most of the owner and theors are the method and the option is to play the object of mềm phần mềm. Trong cuốn sách này, sinh viên được tiếp xúc với cả lý thuyết và thực hành của Quy trình hợp tác nhất.
- Trong Chương 1, điểm mạnh của đối tượng mô hình được phân tích sâu.

- Vòng đời lặp đi lặp lại và gia tăng đã được giới thiệu sớm nhất có thể, cụ thể là trong Chương 2. Hơn nữa, giống như tất cả các phiên bản trước, nhiều đời mô hình khác nhau được trình bày, vì vậy sánh và đối chiếu. Ý kiến đặc biệt đến nhanh các quy tắc.
- Trong Chương 3 ("Phần mềm quy định"), quy trình làm việc (hoạt động) và quy trình của hợp tác quy trình được giới thiệu và sự cần thiết của các vòng đời mô hình được giải thích.
- Nhiều cách phần mềm nhóm tổ chức được trình bày trong Chương 4 ("Nhóm"), bao gồm các nhóm cho các chương trình nhanh và để phát triển open source phần mềm.
- Chương 5 ("Các công cụ thương mại") bao gồm thông tin về các loại công cụ CASE quan trọng.
- Tầm quan trọng của liên tục kiểm tra được nhấn mạnh trong Chương 6 ("Kiểm tra").
- Tiếp tục đối tượng là trọng tâm của sự chú ý trong Chương 7 ("Từ mô-đun đến Đối tượng").
- Các thiết kế mẫu vẫn là trọng tâm của Chương 8 ("Sử dụng tái sinh khả năng và tính năng động").
- IEEE tiêu chuẩn cho phần mềm quản lý dự án một lần nữa được trình bày trong Chương 9 ("Lập kế hoạch và Ước tính").
- Chương 11 ("Yêu cầu"), Chương 13 ("Phân tích hướng đối tượng") và Chương 14 ("Thiết kế") các phần lớn dành cho công việc quy tắc (hoạt động) của hợp nhất Quy trình. Vì ràng buộc quản lý, Chương 12 ("Cổ điển Phân tích") không thay đổi phần lớn.
- Document in Program 15 ("Thực hiện") phân biệt ràng buộc giữa thực hiện và tích hợp.
- Tầm quan trọng của việc bảo trì sau khi giao hàng được nhấn mạnh trong Chương 16.
- Chương 17 cung cấp bổ sung tài liệu về UML để chuẩn bị kỹ lưỡng cho sinh viên khi làm việc trong phần mềm công nghiệp. Đặc biệt của chương trình này được sử dụng cho những giảng viên sử dụng cuốn sách này cho khóa học kỹ thuật phần mềm tự động kéo dài hai thời kỳ. Trong kỳ học thứ hai, ngoài công việc phát triển dự án dựa trên nhóm hoặc dự án capstone, sinh viên có thể thu nhận thêm kiến thức về UML, ngoài những thứ cần thiết cho cuốn sách này.
- Như trước đây, có hai cấu hình nghiên cứu đang chạy. Nghiên cứu hình ảnh của tổ chức MSG và mô hình nghiên cứu về máy chủ đề đã được phát triển bằng cách sử dụng Quy trình nhất. Như thường lệ, Java và C ++ khai triển có sẵn trực tuyến tạiwww.mhhe.com/schach.
- Ngoài cấu hình điển nghiên cứu hai, đang chạy được sử dụng để minh họa toàn bộ đời sống, tám nghiên cứu điển hình nhỏ ghi rõ các chủ đề có thể, không hạn chế như vấn đề di chuyển, sàng lọc từng bước, thiết bị mẫu kế và bảo trì sau giao hàng.
- Trong tất cả các phiên bản trước, tôi đã nhấn mạnh tầm quan trọng của các tài liệu, bảo trì, tái sử dụng, tính năng động, kiểm tra và CASE. Trong bản ấn này, tất cả các khái niệm đều được ấn mạnh như nhau. Việc dạy cho sinh viên những ý tưởng mới nhất cũng chẳng ích gì khi họ đánh giá cao tầm quan trọng của những phần mềm cơ bản của phần mềm kỹ thuật.
- Như trong ấn bản thứ bảy, chú thích đặc biệt đến đối tượng định hướng mô hình, đối tượng định hướng, đối tượng thiết kế, quản lý chức năng của đối tượng hướng mô hình và kiểm tra và bảo vệ phần mềm hướng dẫn. Các chỉ số cho đối tượng hướng mô hình cũng được bao gồm. Ngoài ra, nhiều tham chiếu ngắn gọn được thực hiện cho các đối tượng, một đoạn văn bản hoặc thậm chí chỉ một câu có độ dài. Lý do là đối tượng mô hình không chỉ quan tâm đến cách các giai đoạn khác nhau được thực hiện mà còn thấm hút cách chúng ta nghĩ về phần mềm kỹ thuật. The artwork can be a time in this book tràn ngập.

- Phần mềm quy trình là khái niệm làm nền tảng cho toàn bộ cuốn sách. To check out the process, we must have the đo lường định mức những gì đang xảy ra với dự án.), và ISO / IEC 12207 đã được giữ lại.

- Sách vẫn là độc lập ngôn ngữ. Một vài ví dụ mã được hiển thị bằng C ++ và Java, và tôi đã cố gắng hết sức để giải quyết các chi tiết phụ thuộc vào ngôn ngữ và bảo đảm rằng các ví dụ mã đều rõ ràng như nhau đối với người dùng C ++ và Java. Ví dụ, thay vì sử dụngcoutfor start up C ++ andSystem.out.printlnĐối với Java đầu, tôi đã sử dụng giả mã lệnh*trong*. (Một ngoại lệ là dạng mới điển hình nghiên cứu, trong đó các chi tiết phát triển hoàn chỉnh được cung cấp trong cả C ++ và Java, như trước đây.)
- Như trong bảy lần xuất bản, này cuốn sách chứa hơn 600 tham chiếu tài liệu. Tôi cũng đã chọn hiện tại các bài kiểm tra như các bài báo và cổ điển có thông báo mới và phù hợp. Không có gì phải bàn cãi khi nói rằng phần mềm kỹ thuật là một lĩnh vực chuyển động nhanh chóng và làm điều đó sinh viên cần biết kết quả mới nhất và có thể tìm thấy chúng trong tài liệu. Đồng thời, nghiên cứu đầu tiên của ngày hôm nay dựa trên sự thật của ngày hôm nay và tôi thấy không có lý do gì để loại trừ một cũ tham khảo tài liệu hơn nếu những ý kiến của nó có thể áp dụng ngày hôm nay như ban đầu.
- Đối với các điều kiện tiên quyết, người đọc giả định rằng người đọc đã quen thuộc với ngôn ngữ lập trình cao cấp như C, C #, C ++ hoặc Java. Ngoài ra, người đọc được mong đợi đã tham gia một khóa học về dữ liệu cấu trúc.

Tại sao cổ điển mô hình vẫn được bao gồm

Show nay hầu như đều nhất trí ưu tiên đối tượng mô hình việt hơn từ điển mô hình. Theo đó, nhiều người hướng dẫn thông qua ấn bản thứ bảy*Phần mềm hướng dẫn kỹ thuật và cổ điển*đã chọn chỉ đối tượng hướng dẫn tài liệu trong cuốn sách đó. Tuy nhiên, khi được hỏi, các giảng viên chỉ ra rằng họ thích sử dụng một văn bản bao gồm cổ điển mô hình.

Lý do là vậy, mặc dù ngày càng có nhiều giáo viên hướng dẫn*dạy bảo*chỉ đối tượng hướng mô hình, họ vẫn tiếp tục*tham khảo*to cổ điển mô hình trong lớp; học sinh khó hiểu nhiều kỹ thuật hướng đối tượng trừ khi học sinh đó có một số ý tưởng về từ điển kỹ thuật mà từ đó hướng đối tượng kỹ thuật được định hình thành. Ví dụ, việc hiểu lớp thực hiện mô hình sẽ dễ dàng hơn đối với sinh viên đã được giới thiệu, thậm chí chỉ là một chuyến đi phượt, về mô hình mối quan hệ thực tế. Tương tự như vậy, phần giới thiệu ngắn gọn về chủ sở hữu trạng thái giúp người hướng dẫn dễ dàng hơn trong bài giảng dạy các trạng thái sơ đồ. Theo đó, tôi đã giữ lại cổ điển tài liệu trong lần xuất bản thứ tám, để những người hướng dẫn có sẵn cổ điển tài liệu cho các sư phạm mục tiêu.

The problem set

Như trong lần xuất bản thứ bảy, sách này có các vấn đề năm. Đầu tiên, có thiết kế dự án và định hướng đối tượng đang chạy ở cuối Chương 11, 13 và 14. Các dự án này đã được đưa vào bởi vì cách duy nhất để học cách thực hiện các yêu cầu, phân tích tích và thiết kế quy trình là từ bàn tay rộng rãi- về kinh nghiệm.

Thứ hai, mỗi cuối chương trình có một số bài tập làm nổi bật những điểm chính. This tập tin đóng kín; Kỹ thuật thông tin cho tất cả các tập bài có thể được tìm thấy trong cuốn sách này.

Thứ ba, có một phần mềm thời hạn dự án. Nó được thiết kế để giải quyết bởi các sinh viên làm việc theo nhóm ba người, số lượng thành viên trong nhóm nhỏ nhất mà không thể chuyển qua tiêu chuẩn điện thoại. Dự án thuật ngữ bao gồm 15 thành phần riêng biệt, mỗi thành phần liên kết với chương trình liên kết. Ví dụ, thiết kế là chủ đề của Chương 14, vì vậy trong chương đó, thành phần của dự án liên quan đến phần mềm thiết kế. Bằng cách chia một dự án lớn thành các phần nhỏ hơn, được xác định rõ ràng, người hướng dẫn có thể theo dõi tiến trình của lớp học chặt chẽ hơn. Cấu trúc của kỳ hạn dự án sao cho người hướng dẫn có thể tự áp dụng 15 thành phần cho bất kỳ dự án nào khác mà họ chọn.

Vì cuốn sách này được viết để sử dụng cho các sinh viên cao học cũng như sinh viên khóa trên, nên loại bài toán thứ tư dựa trên các nghiên cứu bài trong tài liệu về phần mềm kỹ thuật. Trong mỗi chương trình, một quan trọng bài báo đã được chọn; Bất cứ khi nào có thể, một bài báo liên quan đến các đối tượng phần mềm kỹ thuật đã được chọn. Học sinh được yêu cầu đọc bài báo và trả lời một câu hỏi liên quan đến nội dung của nó. Natural, the Guide to be you can do the thao tác giao bất kỳ điều gì khác biệt với các nghiên cứu sinh; Phần Read add at the end each program include many links of document.

Loại liên kết vấn đề đến hợp trường nghiên cứu. Loại vấn đề này lần đầu tiên được đưa ra trong lần xuất bản thứ ba để trả lời lại một số giảng viên cảm thấy rằng sinh viên của họ học được nhiều hơn bằng cách sửa đổi một sản phẩm hiện có hơn bằng cách phát triển new product from start. Nhiều kỹ sư phần mềm cao cấp trong ngành cũng đồng tình với quan điểm đó. Theo đó, mỗi chương trong đó bài tập tình huống được trình bày đều có các vấn đề yêu cầu sinh viên phải sửa đổi bài tập tình huống theo một cách nào đó. Ví dụ, trong một chương trình, sinh viên được yêu cầu thiết kế lại mô hình nghiên cứu bằng cách sử dụng một thiết kế kỹ thuật khác với kỹ thuật được sử dụng cho mô hình nghiên cứu.Trong một chương trình khác, học sinh được hỏi về hiệu quả của công việc thực hiện các bước của parsing hướng đối tượng theo một thứ tự khác. Để dễ dàng sửa đổi nguồn mã của cấu hình mô phỏng nghiên cứu,www.mhhe.com/schach.

Trang web cũng có tài liệu cho giáo viên hướng dẫn, bao gồm toàn bộ bài giảng Bài giảng PowerPoint và lời giải chi tiết cho tất cả các tập bài cũng như đồ án học tập.

Document on UML

This book using the UML Nếu sinh viên chưa có kiến thức về UML trước đó, thì tài liệu này có thể được giảng dạy theo hai cách. Tôi thích dạy UML trên cơ sở đúng lúc; nghĩa là, mỗi khái niệm UML được giới thiệu ngay trước khi nó cần thiết. Sau đây bảng mô tả giới thiệu UML cấu trúc được sử dụng trong cuốn sách này.

Build	So khớp phần UML sơ đồ được giới thiệu
Lớp sơ đồ, chú thích, kế thừa (tổng quát hóa), file, links, the use of the school redirect	Mục 7.7 Mục 11.4.3
Ca sử dụng sơ đồ, mô tả ca sử dụng	Mục 11.7
khuôn mẫu	Mục 13.1
Statechart	Mục 13.6
Tương tác sơ đồ (nhân vật tuần lễ bản đồ, giao tiếp sơ đồ)	Mục 13.15

Ngoài ra, Chương 17 chứa các phần giới thiệu về UML, bao gồm tài liệu ở trên và bên ngoài những thứ cần thiết cho cuốn sách này. Chương 17 có thể được giảng dạy bất cứ lúc nào; nó không phụ thuộc vào tài liệu trong 16 chương trình đầu tiên. Các chủ đề được cập nhật trong Chương 17 như sau:

Build	So khớp phần UML sơ đồ được giới thiệu
Layer sơ đồ, tổng hợp, đa dạng, thành phần, khái niệm, liên kết Ghi chú	Mục 17.2
	Mục 17.3
Use case field map	Mục 17.4
Khuôn mẫu	Mục 17,5
Tương tác sơ đồ	Mục 17.6
Statechart	Mục 17.7
Hoạt động sơ đồ	Mục 17.8
Bưu kiện	Mục 17,9
Sơ đồ thành phần	Mục 17.10
Khai triển sơ đồ	Mục 17.11

The source information on network

Một trang web đi kèm với văn bản có sẵn tạiwww.mhhe.com/schach.Trang web có Java và C ++ khai triển cũng như nguồn mã cho điển hình học về MSG dành cho sinh viên. Đối với hướng dẫn viên giáo dục, bài giảng PowerPoint có sẵn, lời giải chi tiết cho tất cả các bài tập và dự án học tập, và một hình ảnh thư viện. Để biết chi tiết, hãy liên hệ với của bạn bán hàng đại diện.

Khoa học điện tử Sách tùy chọn

Sách điện tử là một cách sáng tạo để sinh ra tiền tiết kiệm và đồng thời tạo ra một môi trường xanh hơn. Sách điện tử có thể giúp sinh viên tiết kiệm khoảng một nửa chi phí so với hệ thống truyền thông giáo dục và cung cấp các tính năng độc đáo như công cụ tìm kiếm sức mạnh, đánh dấu và khả năng chia sẻ chú thích với bạn học bằng điện tử.

McGraw-Hill cung cấp văn bản này dưới dạng sách điện tử. Để nói về các điện tử tùy chọn, hãy liên hệ với đại diện bán hàng của bạn McGraw-Hill hoặc truy cập trang webwww.coursesmart.comđể tìm hiểu thêm.

Nhìn nhận sự việc

Tôi đánh giá rất cao những ý kiến phản biện mang tính xây dựng và nhiều ý kiến đóng góp cho các phản biện của bảy lần xuất bản trước. Xin gửi đặc biệt cảm ơn tới những người đánh giá ấn bản này, bao gồm

Ramzi Bualuan
Đại học Notre Dame

Ruth Dameron
Đại học Colorado, Boulder

Werner Krandick
Drexel đại học

Mike McCracken
Viện Công nghệ Georgia

Nenad Medvidovic
Đại học Nam California

Saeed Monemi
Đại học Bách khoa California, Pomona

Taehyung Wang
Đại học Bang California, Northridge

Xiaojun Qi
Đại học Bang Utah

Jie Wei
Đại học Thành phố New York - Cao đẳng Thành phố

Liên quan đến các nhà xuất bản của tôi, McGraw-Hill, tôi biết ơn người sao chép Kevin Campbell và nhà thiết kế Brenda Rolwes. Một lời cảm ơn đặc biệt dành cho Melissa Welch của Studio Montage, người đã biến bức ảnh Cầu Cảng Sydney vào ban đêm thành trang tuyệt đẹp.

Cũng xin gửi lời cảm ơn đặc biệt tới Jean Naudé (Vaal Đại học Công nghệ, Secunda Cơ sở) vì đã có đồng tác giả Bút ký giải pháp cho Người hướng dẫn. Đặc biệt, Jean đã cung cấp một giải pháp hoàn chỉnh cho dự án thuật ngữ, bao gồm cả việc triển khai nó bằng cả Java và C ++. Trong quá trình làm việc trên ISM, Jean đã đưa ra nhiều đề xuất mang tính xây dựng để cải thiện cuốn sách. Tôi biết ơn Jean nhất.

Cuối cùng, như mọi khi, tôi cảm ơn vợ tôi, Sharon, vì cô ấy đã không ngừng ủng hộ và động viên. Như với tất cả các cuốn sách trước đây của tôi, tôi đã cố gắng hết sức để bảo đảm rằng các cam kết trong gia đình được ưu tiên hơn các công việc viết. Tuy nhiên, khi thời hạn xuất hiện, điều này không có nghĩa là nó cũng có thể thực hiện được. Những lúc như vậy, Sharon luôn hiểu, và điều này tôi biết ơn nhất.

Đó là đặc ân của tôi để dành tặng cuốn sách thứ bảy của tôi cho các cháu của tôi, Jackson và Mikaela, với tình yêu thương.

Stephen R. Schach

This page cố gắng để trống

Chapter 1

Phạm vi của phần mềm kỹ thuật

File tiêu đề

Sau khi học chương trình này, bạn sẽ có thể

- Xác định nghĩa của phần mềm kỹ thuật.
- Mô tả cổ điển phần mềm kỹ thuật số mô hình.
- Giải thích tại sao hiện đối tượng mô hình được chấp nhận như vậy.
- Thảo luận về các hoạt động của các mặt khác nhau của phần mềm kỹ thuật.
- Phân biệt giữa các điểm cổ điển và hiện đại về bảo trì.
- Thảo luận về tầm quan trọng của công việc lập kế hoạch, kiểm tra và liên tục thiết lập tài liệu.
- Đánh giá cao tầm quan trọng của công việc tuân thủ quy tắc đạo đức.

Một câu chuyện nổi tiếng về một giám đốc điều hành đã nhận được một đơn vị hóa do máy tính tạo ra với giá 0,00 đô la. Sau khi có một trận cười sảng khoái với bạn bè về "chiếc máy tính ngốc", vị giám đốc điều hành bỏ đơn. Một tháng sau, một tương tự đơn vị hóa, lần này được đánh dấu 30 ngày. Sau đó đến đơn thứ ba. Tư vấn đơn giản đến một tháng sau đó, kèm theo một tin nhắn chỉ có khả năng hoạt động nếu đơn vị hóa $ 0,00 không được thanh toán ngay lập tức.

Dự đoán thứ năm, được đánh dấu 120 ngày, không gợi ý gì - lỗ thông tin và thẳng thừng, đe dọa mọi hành pháp nếu đơn giản hóa không được thanh toán ngay lập tức. Lo sợ về việc xếp hạng tín nhiệm của tổ chức mình rơi vào máy điện lạnh, vị trí giám đốc điều hành đã gọi cho một người làm quen là kỹ sư phần mềm và nói lại toàn bộ câu chuyện xin lỗi. Cố gắng không cười, kỹ sư phần mềm bảo vệ người điều hành gửi 0,00 USD qua đường bưu điện. Điều này đã mang lại hiệu quả như mong muốn và một vài ngày sau chúng tôi sẽ nhận được biên lại $ 0,00. Giám đốc điều hành đã tiến hành nộp lệ phí nó đi trong trường hợp ngày nào đó trong tương lai máy tính có thể báo cáo rằng vẫn còn nợ 0,00 đô la.

This ngôn ngữ nổi tiếng có phần tiếp theo được biết đến hơn. Một vài ngày sau, giám đốc điều hành đã được triệu tập bởi giám đốc ngân hàng của mình. Nhân viên ngân hàng nâng lên và hỏi, "This must be a séc of your not?"

Giám đốc điều hành đã đồng ý rằng đúng như vậy.

"Bạn có phức tạp cho tôi biết tại sao bạn viết séc với giá 0,00 đô la không?" nhân viên ngân hàng hỏi. Vì vậy, toàn bộ câu chuyện đã được kể lại. Khi giám đốc điều hành đã hoàn thành, nhân viên ngân hàng quay sang anh ta và cô ấy hỏi, "Anh có biết tấm séc 0,00 đô la của anh không làm gì cả*của chúng tôi, tôi*máy tính hệ thống? "

Một máy tính chuyên nghiệp có thể bật cười trước câu chuyện này, mặc dù hơi lo lắng. Rốt cuộc, mỗi người trong chúng ta thiết kế hoặc triển khai một sản phẩm mà ở dạng ban đầu, kết quả tương đương với việc gửi những bức thư ảo quyệt với giá 0,00 đô la. Cho đến nay, chúng tôi luôn gặp lỗi này trong quá trình thử nghiệm. Nhưng tiếng cười của chúng tôi, tôi có phần trống rỗng, bởi vì trong tâm trí chúng tôi cảm thấy sợ hãi và sợ hãi một ngày nào đó chúng tôi sẽ không phát hiện ra lỗi trước khi sản phẩm được giao cho khách hàng.

Một phần mềm lỗi quyết định ít hài hước hơn đã được phát hiện vào ngày 9 tháng 11 năm 1979. Bộ Tư lệnh Không quân Chiến lược phải cảnh báo khi hệ thống máy tính mạng chỉ huy và kiểm tra quân sự trên toàn thế giới giới (WWMCCS) báo cáo rằng Liên Xô phóng tên lửa nhằm vào Hoa Kỳ [Neumann, Năm 1980]. The realments is a a new public model was modulation as real, same as in film*Những trò chơi chiến tranh*khoảng 5 năm sau. Mặc dù Bộ Quốc phòng Hoa Kỳ không đưa ra thông tin chi tiết về cơ chế chính xác mà dữ liệu thử nghiệm được thực hiện dữ liệu, có sự hợp lý khi đặt vấn đề là phần mềm bị lỗi. Toàn bộ hệ thống không được thiết kế để phân biệt giữa các mô phỏng và thực tế hoặc người dùng giao diện không bao gồm các yêu cầu kiểm tra để đảm bảo rằng cuối cùng của hệ thống người dùng có thể phân biệt thực tế với thực tế. Nói cách khác, một phần mềm lỗi, nếu vấn đề thực sự là do phần mềm, có thể được đưa ra cho văn bản minh họa như chúng ta biết đến một kết thúc khó chịu và kẹt cứng. (See only in the School Bạn muốn biết Hộp 1.1 để biết thông tin về các thảm họa do các phần mềm khác gây ra lỗi.)

Cho dù chúng tôi đang giải quyết vấn đề thanh toán hay phòng không, phần mềm lớn của chúng tôi được phân phối, vượt quá ngân sách và có những lỗi rút lại và không đáp ứng được yêu cầu của hàng. Phần mềm kỹ thuật là một nỗ lực để giải quyết các vấn đề này. Khác cách nói,**phần mềm kỹ thuật** is an learning is a target item is an production section is no error, is the time analysis and in range vi ngân sách, đáp ứng nhu cầu của khách hàng. Hơn nữa, phần mềm phải dễ dàng sửa đổi khi người dùng yêu cầu thay đổi.

Phạm vi kỹ thuật của phần mềm rất rộng. Một số khía cạnh của phần mềm kỹ thuật có thể được phân loại là toán học hoặc khoa học máy tính; khác biệt bên rơi vào các lĩnh vực kinh tế, quản lý hoặc tâm lý học. Để hiển thị phần lớn phạm vi của phần mềm kỹ thuật, bây giờ chúng ta cùng xem xét các khía cạnh khác nhau.

1.1 Các cạnh lịch sử

Có một tế bào thực hiện là máy phát điện bị hỏng, nhưng ít hơn so với các sản phẩm trả lương. Cầu đôi khi bị lật nhưng ít thường xuyên hơn đáng kể với hệ điều hành. Với niềm tin rằng thiết kế, phần mềm triển khai và phần mềm bảo trì có thể được thực hiện giống nhau

Trong trường hợp của mạng WWMCCS, thảm họa đã được ngăn chặn vào cuối phút. Tuy nhiên, hậu quả của các phần mềm khác lỗi đã gây ra vong. Ví dụ, từ năm 1985 đến năm 1987, ít nhất là hai bệnh nhân tử vong do sử dụng quá mức, phóng xạ nghiêm trọng đối với máy tính tốc độ cao của Therac-25 cung cấp [Leveson và Turner, 1993]. Nguyên nhân là do error in control software.

Ngoài ra, trong Chiến tranh vùng Vịnh năm 1991, một ngọn lửa tên Scud đã xuyên thủng lá chắn chống lửa tên của Người yêu nước và tấn công một doanh trại gần Dhahran, Ả Rập Xê-út. Tổng cộng, 28 người thiệt hại mạng và 98 người bị thương. Phần mềm của tên lửa Patriot có một thời gian tích lũy lỗi. Patriot được thiết kế để chỉ hoạt động trong vài giây mỗi lần, sau đó đồng hồ được đặt lại. Kết quả là, error no time as the image value and do it is not be Play. Tuy nhiên, trong Chiến khu Vịnh, dàn lửa Tên lửa tại Dhahran đã hoạt động liên tục trong hơn 100 giờ. Điều này làm chênh lệch thời gian tích lũy trở nên đủ lớn để kết xuất hệ thống không chính xác.

Trong Chiến tranh vùng Vịnh, Hoa Kỳ đã chuyển tên lửa Yêu nước cho Israel để bảo vệ Scuds. Lực lượng Israel đã phát hiện vấn đề về thời gian chỉ sau 8 giờ và báo cáo ngay cho nhà sản xuất ở Mỹ. Nhà sản xuất đã sửa lỗi nhanh nhất có thể, nhưng bi thảm, phần mềm mới xuất hiện một ngày sau vụ tấn công trực tiếp bởi Scud [Mellor, 1994].

May thay, rất hiếm khi xuất hiện trường hợp tử hoặc thương hiệu nghiêm trọng của phần mềm lỗi. Tuy nhiên, một lỗi có thể gây ra sự cố cho những người hàng nghìn tỷ. Ví dụ, vào tháng 2 năm 2003, một phần mềm lỗi đến Bộ Tài chính Hoa Kỳ gửi 50.000 điểm An sinh Xã hội đã được in ra mà không có tên của người thụ hưởng, do đó, không thể gửi hoặc chuyển tiền mặt [St. Petersburg Times Online, 2003]. Vào tháng 4 năm 2003, những người đi vay được SLM Corp. (thường được gọi là Sallie Mae) thông báo rằng hệ thống quản lý tài khoản cho vay sinh viên của họ đã được tính toán sai phần mềm lỗi từ năm 1992 nhưng chỉ được phát hiện vào cuối năm 2002. Gần 1 triệu người vay được biết rằng họ sẽ phải trả nhiều hơn,or under the format of the current thanh toán hàng tháng hoặc trả thêm lãi suất cho các khoản vay kéo dài quá thời hạn 10 năm ban đầu [GJSentinel.com, 2003]. Cả hai lỗi đều nhanh chóng được sửa chữa, nhưng họ giống nhau dẫn đến tài chính kết quả không nhỏ cho một triệu người.

Role chính phủ đánh giá quá cao ngân sách năm 2007 của mình€883.000.000 (hơn 1.100.000.000 $ tại thời điểm viết bài). This error is do the part soft error link with override overlayers engine mode is error [La Libre Online, 2007a; 2007b]. Bỉ quan thuế, sử dụng máy quét và quang học ký tự nhận dạng phần mềm để xử lý các tờ khai thuế. If phần mềm phải trả lại không thể đọc được, nó sẽ ghi lại phần nhập của người nộp thuế là€ 99,999,999,99 (hơn 125.000.000 $). Có lẽ, "con số kỳ diệu"€99.999.999.99 has been select to be well by the current people of the processors of process, do that pay return is the update after that will be used process. Điều này có hiệu quả khi các tờ khai thuế được phân tích cho các ngân sách mục tiêu, nhưng không có hiệu quả khi các tờ khai thuế được phân tích lại cho các ngân sách mục tiêu. Trớ trêu thay, sản phẩm phần mềm có bộ lọc để phát hiện các vấn đề, nhưng bộ lọc đã bị loại bỏ theo cách thủ công để tăng tốc độ xử lý.

Có ít lỗi nhất trong phần mềm. Đầu tiên, các phần mềm kỹ thuật cho rằng sẽ luôn có toàn bộ công việc kiểm tra trước khi xử lý thêm dữ liệu. Thứ hai, phần mềm cho phép ghi đè các bộ lọc theo cách thủ công.

Chỉ trong trường hợp bạn muốn biết Box 1.2

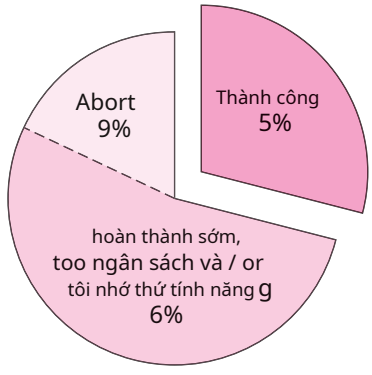
Như đã nêu trong Phần 1.1, mục đích của hội nghị Garmisch là làm cho phần mềm phát triển thành công như hệ thống truyền thông kỹ thuật. But doesn't mean is all all the media project are to public. Ví dụ, hãy xem công việc xây dựng cây cầu.

Vào tháng 7 năm 1940, xây dựng một cây cầu treo bắc qua Tacoma Narrows, ở Bang Washington, được hoàn thành. Ngay sau đó, người ta phát hiện cây lắc lư, oằn mình gây nguy hiểm trong điều kiện trời gió. Những ô tô màu này sẽ gần như biến mất vào thung lũng và sau đó xuất hiện trở lại khi phần đó của cây cầu tăng trở lại. This từ hành vi, cây cầu được đặt cho danh sách đặc biệt "Gertie phi nước đại." Cuối cùng, vào ngày 7 tháng 11 năm 1940, cây cầu bị cuốn trong một cơn gió 42 dặm một giờ; May May thay, tree request đã được đóng cửa lại trước đó vài giờ thông tin. Cuối cùng 15 phút trong cuộc đời của nó đã được ghi lại trên phim, hiện đã được lưu trữ trong Cơ quan đăng ký phim quốc gia Hoa Kỳ.

Một sự cố xây dựng có phần hài hước hơn đã xuất hiện vào tháng 1 năm 2004. Một cây cầu mới được xây dựng bắc qua sông Upper Rhine gần thị trấn Laufenberg của Đức, để kết nối Đức và Thụy Sĩ. Phần nửa yêu cầu của người Đức do đội ngũ kỹ sư người Đức thiết kế và thi công; hiệp Thụy Sĩ bởi một đội Thụy Sĩ. Khi hai phần được kết nối với nhau, ngay lập tức có thể thấy rằng một nửa của Đức cao hơn khoảng 21 inch (54 cm) so với một nửa của Thụy Sĩ. Cần phải làm lại thiết bị lớn để khắc phục sự cố, nhân nguyên là do sửa sai thực tế là "mực nước biển" được các kỹ sư Thụy Sĩ lấy là mức trung bình của Biển Địa Trung Hải, khi các kỹ sư Đức sử dụng Biển Bắc. To offset the chênh lệch mực nước biển, Thụy Sĩ có thể phải nâng lên 10,5 inch.Thay vào đó, nó đã được hạ xuống 10.

set foot as the media of the language skills, a group of NATO in the year 1967 đã đặt ra thuật ngữ *phần mềm kỹ thuật*. Tuyên bố xây dựng các phần mềm tương tự như các nhiệm vụ kỹ thuật khác đã được chứng nhận bởi Hội nghị Kỹ thuật Phần mềm NATO năm 1968 tổ chức tại Garmisch, Đức [Naur, Randell, và Buxton, 1976]. This certificate is not too a old man; chính cái tên của hội nghị đã phản hồi niềm tin rằng sản xuất phần mềm phải là một hoạt động giống như kỹ thuật (nhưng hãy xem Chỉ trong trường hợp bạn muốn biết ở Box 1.2). Một kết hợp của các cuộc hội thảo là phần mềm kỹ thuật, nên sử dụng các triết lý và mô hình của các kỹ thuật ngành đã được thiết lập để giải quyết những gì họ gọi là**phần mềm khủng hoảng**cụ thể là chất lượng phần mềm nói chung đến mức không thể chấp nhận được và thời hạn cũng như ngân sách không được trả lời.

Mặc dù có nhiều câu chuyện thành công về phần mềm, nhưng một tỷ lệ lớn không thể chấp nhận được các phần mềm của sản phẩm vẫn được xếp hàng, vượt quá ngân sách và với các lỗi bỏ sót. Ví dụ, Standish Group là một công ty nghiên cứu phân tích dự án phát triển phần mềm. Nghiên cứu của họ về dự án phát triển hoàn thành năm 2006 được nghiên cứu tóm tắt trong Hình 1.1 [Rubenstein, 2007]. Chỉ có 35 phần dự án được hoàn thành thành công, trong khi 19 phần trăm bị hủy bỏ trước khi hoàn thành hoặc không bao giờ được thực hiện. 46% is back of the project was complete and install on the computer of client. Tuy nhiên, những dự án đó vượt quá ngân sách, trễ hạn hoặc có ít tính năng và chức năng hơn so với ban đầu quy định.Nói cách khác, trong năm 2006, chỉ có hơn một trong ba dự án phát triển phần mềm là thành công;



Hình 1.1 Hậu quả trên 9.000 sự phát triển dự án hoàn thành vào năm 2006 [Rubenstein, 2007].

Những nguồn tài nguyên tác động của phần mềm khủng hoảng rất khủng khiếp. Trong một cuộc khảo sát khảo sát được thực hiện bởi Cutter Consortium [2002], những điều sau đây đã được báo cáo:

- Đáng kinh ngạc là 78% tổ chức công nghệ thông tin tham gia vào tranh chấp nhận kết thúc bằng điều kiện.
- Trong 67% các trường hợp đó, chức năng hoặc hiệu suất của phần mềm sản phẩm khi được phân phối không đo lường được theo yêu cầu của các nhà phát triển phần mềm.
- Trong 56% the truong hop, the day giao thong đã bị trượt vài lần.
- Trong đó 45 phần trăm phần trăm các trường, các phần tử quan trọng của lỗi sẽ không thể sử dụng được.

Rõ ràng là có quá ít phần mềm được phân phối đúng hạn, trong phạm vi ngân sách, không bị lỗi và đáp ứng nhu cầu của khách hàng. Để đạt được những mục tiêu này, một phần mềm kỹ thuật phải có nhiều kỹ năng, cả kỹ thuật và quản lý. Các kỹ năng này phải được áp dụng không chỉ cho trình cài đặt mà còn cho mọi bước sản xuất phần mềm, từ yêu cầu đến bảo trì sau giao hàng.

Conversation phần mềm vẫn còn tồn tại với chúng ta, khoảng 40 năm sau, cho chúng ta biết hai điều. First, software**tiến trình**, nghĩa là, cách chúng tôi sản xuất phần mềm, có những đặc tính và vấn đề độc đáo của riêng nó, mặc dù nó giống với hệ thống kỹ thuật ở nhiều cạnh. Thứ hai, phần mềm quản trị kinh doanh có thể nên được chuyển đổi tên thành**phần mềm trầm lắng**, nhận xét về độ dài thời gian và xấu số lượng đầu tiên của nó.

Bây giờ chúng ta xem xét các khía cạnh kinh tế của phần mềm kỹ thuật.

1.2 Các cạnh kinh tế

Một phần mềm tổ chức hiện đang sử dụng CT kỹ thuật mã hóa$_{old}$khám phá ra rằng CT mới mã hóa kỹ thuật$_{New}$will help to get code to be only in the ninthfimim time by CT$_{old}$and do that, with niem chi phí. Code Information as the command for that CT$_{New}$ là hợp tác kỹ thuật để sử dụng. Thực tế, mặc dù có thể chắc chắn rằng quy định

kỹ thuật nhanh hơn là lựa chọn kỹ thuật, tính toán kinh tế của phần mềm kỹ thuật có thể làm điều ngược lại.

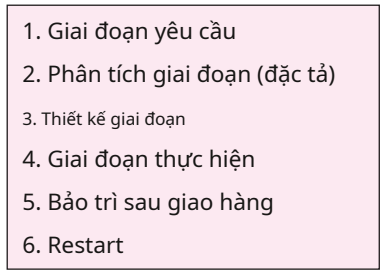
- Một lý do là chi phí giới thiệu công nghệ mới vào một tổ chức. Thực tế là mã hóa nhanh hơn 10% khi kỹ thuật CT_{New}được sử dụng có thể ít quan trọng hơn so với chi phí phát sinh trong giới thiệu CT_{New}vào tổ chức. Có thể cần phải hoàn thành hai hoặc ba dự án trước khi bù đắp chi phí đào tạo. Ngoài ra, khi tham gia các khóa học về CT_{New}, phần mềm nhân viên không thể làm việc hiệu quả. Ngay cả khi họ quay trở lại, một đường cong học tập có thể liên quan; may be mất nhiều hành động tháng với CT_{New}trước khi phần mềm chuyên gia trở nên thành thạo với CT_{New}as they are being with CT_{old}. Do đó, các dự án ban đầu sử dụng CT_{New}There can't be lost many time to complete the more if the group function is going to use CT_{old}. Tất cả các chi phí này cần được tính đến khi quyết định có chuyển sang CT hay không$_{New}$.
- Lý do thứ hai tại sao kinh tế học của phần mềm kỹ thuật có thể quy định rằng CT_{old} được giữ lại là hệ thống duy trì. CT mã hóa kỹ thuật$_{New}$thực sự có thể nhanh hơn 10% so với CT_{old}, và kết quả mã có thể có chất lượng tương đương theo quan điểm đáp ứng nhu cầu hiện tại của khách hàng. But the use of skill Engineering CT_{New} must do to the security code, make for CT chi phí$_{New}$cao hơn so với tuổi thọ của sản phẩm. Tất nhiên, nếu nhà phát triển phần mềm không chịu trách nhiệm về bất kỳ bảo trì nào sau khi phân phối, thì theo quan điểm của nhà phát triển đó, $CT_{New}$is an than lãnh đạo đề xuất. Rốt cuộc, CT sử dụng$_{New}$sẽ có giá thấp hơn 10 phần trăm. Khách hàng nên nhấn mạnh CT kỹ thuật$_{old}$được sử dụng và trả chi phí ban đầu cao hơn với kỳ vọng rằng phần mềm tổng chi phí toàn bộ đời sẽ thấp hơn. Thật không may, mục tiêu duy nhất của cả khách hàng và nhà cung cấp phần mềm là tạo mã càng nhanh càng tốt. Những bức ảnh lâu dài của công việc sử dụng một công cụ kỹ thuật có thể thường bị loại bỏ vì lợi ích ngắn hạn. Áp dụng các nguyên tắc kinh tế vào kỹ thuật phần mềm yêu cầu khách hàng phải chọn các kỹ thuật làm giảm thời gian dài.

This ví dụ liên quan đến mã hóa, sử dụng ít hơn 10% nỗ lực để phát triển phần mềm. Tuy nhiên, kinh tế nguyên tắc cũng áp dụng cho tất cả các mặt khác của phần mềm sản xuất.

Bây giờ chúng tôi xem xét tầm quan trọng của bảo trì.

1.3 Bảo trì các cạnh

Trong phần này, chúng tôi mô tả bảo trì trong ngữ cảnh của phần mềm vòng đời. MỘT**đời mô hình**is a step description must be done when build a soft product. Nhiều người khác nhau trong vòng lặp mô hình đã được xuất đề; một số trong số chúng được mô tả trong Chương 2. Vì hầu như luôn dễ dàng thực hiện một chuỗi các nhiệm vụ nhỏ hơn một nhiệm vụ lớn, tổng thể mô hình có thể được chia thành một chuỗi các bước nhỏ hơn, được gọi is**giai đoạn**. Khác nhau về số lượng giai đoạn giữa các mô hình - từ ít nhất là bốn đến nhiều nhất là tám. Ngược lại với đời sống mô hình, là một lý thuyết mô tả về những gì nên làm, chuỗi các bước thực hiện được thực hiện trên một sản phẩm phần mềm, từ khám phá khái niệm cho đến khi nghỉ cuối cùng, được gọi là**vòng đời**của sản phẩm đó. Trên thực tế, các giai đoạn của đời sống của một sản phẩm phần mềm có thể không được thực hiện chính xác như được định nghĩa chỉ trong đời mô hình, đặc biệt là khi vượt quá thời gian và chi phí.



Hình 1.2
Sáu giai đoạn cổ điển vòng đời mô hình.

đang gặp phải. Người ta định vị lại nhiều phần mềm dự án vì thiếu thời gian hơn là bởi vì tất cả các công cụ khác cộng lại [Brooks, 1975].

Cho đến cuối những năm 1970, hầu hết các tổ chức đang sản xuất phần mềm sử dụng làm mô hình đời của họ, cái mà bây giờ được gọi là**Nước mô hình**. Có nhiều thể biến của mô hình này, nhưng nhìn chung, một sản phẩm được phát triển bằng cách sử dụng mô hình này cổ điển trải qua sáu giai đoạn có thể hiển thị trong Hình 1.2. Các giai đoạn này có thể không tương ứng với các giai đoạn của bất kỳ một tổ chức cụ thể nào, nhưng chúng tôi đủ gần với hầu hết các thông tin cho mục tiêu của cuốn sách. Tương tự, chính tên của từng giai đoạn khác nhau giữa các tổ chức. Những cái tên được sử dụng ở đây cho các giai đoạn khác nhau đã được lựa chọn càng chung càng tốt với hy vọng rằng người đọc sẽ cảm thấy thoải mái với họ.

1. *Giai đoạn yêu cầu.*Trọng**yêu cầu giai đoạn**, khái niệm được khám phá và tinh chỉnh, và các yêu cầu của khách hàng được gợi ý.
2. *Phân tích giai đoạn (đặc tả).*Khách hàng yêu cầu được phân tích và trình bày dưới dạng**thiết kế tài liệu**, "Những gì sản phẩm phải làm." Các**phân tích giai đoạn**đôi khi được gọi là**kỹ thuật giai đoạn**. Vào cuối giai đoạn này, một kế hoạch được vạch ra,**kế hoạch phần mềm dự án quản lý**, phần mềm phát triển mô tả được xuất một cách đầy đủ chi tiết.
3. *Giai đoạn thiết kế*. Các kỹ thuật thông tin trải qua hai liên kết thiết kế quy trình trong quá trình **thiết kế giai đoạn**. Before before**kiến trúc thiết kế**, in that all the product was shared to be small to part, be call is**module**. Sau đó, mỗi mô-đun được thiết kế; This thủ tục được gọi là**chi tiết thiết kế**. Hai results**thiết kế tài liệu**description "cách sản phẩm hoạt động."
4. *Giai đoạn thực hiện*. Các thành phần khác nhau trải qua**mã hóa**and try (**kiểm tra vị trí**) private. Sau đó, các thành phần của sản phẩm được kết hợp và kiểm tra tổng thể; this is call is**nhập hội**. Khi các nhà phát triển hài lòng rằng sản phẩm hoạt động chính xác, nó sẽ được khách hàng kiểm tra (**kiểm tra chấp nhận**). Các**thực hiện giai đoạn**kết thúc khi sản phẩm được khách hàng chấp nhận và cài đặt trên máy tính của khách hàng. (Chúng tôi đã thấy trong Chương 15 rằng mã hóa và tích hợp nên được thực hiện thành bài hát.)
5. *Bảo trì sau giao hàng.*Sản phẩm được sử dụng để thực hiện các nhiệm vụ mà nó đã được phát triển. This time, it was duy trì.**Bảo trì sau giao hàng** bao gồm tất cả các thay đổi đối với sản phẩm sau khi sản phẩm được giao và cài đặt trên máy tính của khách hàng và vượt qua kiểm tra chấp nhận. Bảo trì sau giao hàng

Chỉ trong trường hợp bạn muốn biết Box 1.3

Một trong những kết quả được trích dẫn rộng rãi nhất trong phần mềm kỹ thuật là 17,4 phần trăm nỗ lực bảo trì sau giao hàng là người phục hồi về bản chất; 18,2 phần trăm là ứng dụng; 60,3 phần trăm là hoàn thành; and 4,1 phần trăm có thể được phân loại là "khác". This result is get from a message output of year 1978 [Lientz, Swanson, and Tompkins, 1978].

Tuy nhiên, kết quả trong bài báo đó không bắt nguồn từ *đo* về bảo trì dữ liệu. Thay vào đó, các tác giả tiến hành một cuộc khảo sát về các nhà quản lý bảo trì, những người được yêu cầu *ước tính* dành bao nhiêu thời gian cho từng hạng mục trong toàn bộ tổ chức của họ và cho biết họ tự tin như thế nào về ước tính của mình. Cụ thể hơn, những người quản lý bảo trì phần mềm tham gia được hỏi thông tin phản hồi của họ có dựa trên hợp lý chính xác dữ liệu, tối thiểu dữ liệu hay không; 49,3 phần trăm nói rằng câu trả lời của họ dựa trên hợp lý chính xác dữ liệu, 37,7 phần trăm phần trăm dựa trên tối thiểu dữ liệu và 8,7 phần trăm phần trăm không có dữ liệu.

Trên thực tế, người ta nên đặt câu hỏi bình thường về công việc có bất kỳ người trả lời nào có "hợp lý chính xác dữ liệu" về thời gian tỷ lệ dành cho bảo trì hạng mục trong cuộc khảo sát hay không; hầu hết chúng tôi có lẽ thậm chí không có "tối thiểu dữ liệu". Trong đó tham khảo khảo sát, những người tham gia được yêu cầu tỷ lệ phần trăm bảo trì bao gồm các hạng mục như "sửa lỗi cấp" hoặc "kỳ hạn sửa lỗi"; from this raw information, the percation rate of secure security application, sửa chữa và hoàn tất đã được suy ra. Phần mềm kỹ thuật chỉ bắt đầu nổi lên như một ngành học vào năm 1978, và các nhà quản lý bảo trì phần mềm phải thu thập chi tiết thông tin cần thiết để trả lời một cuộc khảo sát như vậy là ngoại lệ.Thật vậy, theo thuật ngữ hiện đại, vào năm 1978 hầu như mọi tổ chức vẫn ở cấp độ CMM 1 (xem Phần 3.13).

Do đó, chúng tôi có cơ sở vững chắc để đặt câu hỏi liệu phân bổ thực tế của các hoạt động bảo trì sau giao hàng vào năm 1978 có giống như ước tính của các nhà quản lý tham gia cuộc khảo sát hay không. Việc phân bổ các hoạt động bảo trì chắc chắn không có gì giống như ngày nay. Ví dụ: kết quả về dữ liệu bảo trì thực tế cho nhân Linux [Schach và cộng sự, 2002] và trình biên dịch gcc [Schach và cộng sự, 2003] cho thấy rằng ít nhất 50 phần trăm bảo trì sau giao hàng là sửa chữa, trái ngược với 17,4 phần trăm con số được yêu cầu trong cuộc khảo sát.

---

bao gồm **bảo trì sửa chữa** (hoặc là **sửa chữa phần mềm**), bao gồm các loại bỏ các lỗi còn sót lại trong khi vẫn giữ nguyên các công cụ biểu tượng, cũng như **nâng cao** (hoặc cập nhật phần mềm), bao gồm các thay đổi đối với các công cụ thể hiện và triển khai các thay đổi đó. Lần lượt, có cường độ tăng cường hai loại. First is **bảo trì hoàn**, những thay đổi mà khách hàng cho rằng sẽ cải thiện hiệu quả của sản phẩm, chẳng hạn như chức năng bổ sung hoặc giảm thời gian phản hồi. Thứ hai là **bảo trì thích ứng**, những thay đổi được thực hiện để đáp ứng với những thay đổi trong môi trường mà sản phẩm hoạt động, chẳng hạn như phần cứng / hệ điều hành mới hoặc các quy định mới của chính phủ. (Để có cái nhìn sâu sắc về ba loại bảo trì sau giao hàng, hãy xem Chỉ trong trường hợp bạn muốn biết ở Hộp 1.3.)

6. *Sự nghỉ hưu.* **Sự nghỉ hưu** xảy ra khi sản phẩm bị xóa khỏi dịch vụ. Điều này xảy ra khi chức năng được cung cấp bởi sản phẩm không còn có ích cho tổ chức khách hàng.

Bây giờ chúng ta xem xét định nghĩa của *Sự bảo trì* chi tiết hơn.

1.3.1 Quan điểm cổ điển và hiện đại về bảo trì

Trong những năm 1970, sản xuất phần mềm được coi là bao gồm hai hoạt động riêng biệt được thực hiện tuần tự: *sự phát triển* theo dõi bởi *Sự bảo trì*. Bắt đầu từ đầu, sản phẩm phần mềm được phát triển, và sau đó được cài đặt trên máy tính của khách hàng. Bất kỳ thay đổi nào đối với phần mềm sau khi cài đặt trên máy tính của khách hàng và được khách hàng chấp nhận, cho dù để sửa lỗi còn sót lại hoặc mở rộng chức năng, đều được cấu thành bảo trì cổ điển [IEEE 610.12, 1990]. Do đó, cách mà phần mềm được phát triển theo kiểu cổ điển có thể được mô tả là **mô hình phát triển sau đó bảo trì**.

Đây là một **định nghĩa thời gian**; nghĩa là, một hoạt động được phân loại là phát triển hoặc bảo trì tùy thuộc vào thời điểm nó được thực hiện. Giả sử rằng một lỗi trong phần mềm được phát hiện và sửa chữa một ngày sau khi phần mềm được cài đặt. Theo định nghĩa, điều này tạo thành bảo trì cổ điển. Nhưng nếu lỗi giống hệt được phát hiện và sửa vào ngày trước khi phần mềm được cài đặt, về mặt định nghĩa, điều này tạo nên sự phát triển cổ điển. Bây giờ, giả sử rằng một sản phẩm phần mềm vừa được cài đặt nhưng khách hàng muốn tăng chức năng của sản phẩm phần mềm. Theo cổ điển, điều đó sẽ được mô tả là bảo trì hoàn hảo. Tuy nhiên, nếu khách hàng muốn thay đổi tương tự được thực hiện ngay trước khi sản phẩm phần mềm được cài đặt, thì đây sẽ là sự phát triển cổ điển. Một lần nữa, không có sự khác biệt nào giữa bản chất của hai hoạt động,

Ngoài những mâu thuẫn như vậy, hai lý do khác giải thích tại sao mô hình phát triển-bảo trì ngày nay là không thực tế:

1. Ngày nay, chắc chắn việc thi công một sản phẩm mất một năm trở lên không còn là chuyện lạ. Trong thời gian này, các yêu cầu của khách hàng có thể thay đổi. Ví dụ: khách hàng có thể khăng khăng rằng sản phẩm hiện được triển khai trên một bộ xử lý nhanh hơn, vừa mới có sẵn. Ngoài ra, tổ chức khách hàng có thể đã mở rộng sang Bỉ trong khi quá trình phát triển đang được tiến hành và sản phẩm hiện phải được sửa đổi để nó cũng có thể xử lý doanh số bán hàng ở Bỉ. Để xem sự thay đổi trong các yêu cầu có thể ảnh hưởng đến vòng đời phần mềm như thế nào, hãy giả sử rằng các yêu cầu của khách hàng thay đổi trong khi thiết kế đang được phát triển. Nhóm kỹ sư phần mềm phải tạm ngừng phát triển và sửa đổi tài liệu đặc tả để phản ánh các yêu cầu đã thay đổi. Hơn nữa, sau đó có thể cần phải sửa đổi thiết kế, nếu các thay đổi đối với các thông số kỹ thuật đòi hỏi các thay đổi tương ứng đối với các phần đó của thiết kế đã hoàn thành. Chỉ khi những thay đổi này được thực hiện mới có thể tiếp tục phát triển. Nói cách khác, các nhà phát triển phải thực hiện "bảo trì" rất lâu trước khi sản phẩm được cài đặt.
2. Vấn đề thứ hai với mô hình phát triển-sau đó-bảo trì cổ điển nảy sinh do cách chúng ta xây dựng phần mềm hiện nay. Trong kỹ thuật phần mềm cổ điển, một đặc điểm của phát triển là nhóm phát triển đã xây dựng sản phẩm mục tiêu bắt đầu từ đầu. Ngược lại, do chi phí sản xuất phần mềm ngày nay cao, các nhà phát triển cố gắng sử dụng lại các phần của sản phẩm phần mềm hiện có trong sản phẩm phần mềm được xây dựng ở bất cứ đâu (việc sử dụng lại được thảo luận chi tiết trong Chương 8). Do đó, mô hình phát triển-sau đó-bảo trì ngày nay là không thích hợp vì việc tái sử dụng quá phổ biến.

Một cách nhìn thực tế hơn về bảo trì được đưa ra trong tiêu chuẩn cho các quy trình vòng đời do Tổ chức Tiêu chuẩn hóa Quốc tế (ISO) xuất bản.

Chỉ trong trường hợp bạn muốn biết Hộp 1.4

Tổ chức Tiêu chuẩn hóa Quốc tế (ISO) là một mạng lưới các viện tiêu chuẩn quốc gia của 147 quốc gia, với ban thư ký trung ương có trụ sở tại Geneva, Thụy Sĩ. ISO đã xuất bản hơn 13.500 tiêu chuẩn được quốc tế chấp nhận, từ các tiêu chuẩn về tốc độ phim ảnh ("số ISO") đến nhiều tiêu chuẩn được trình bày trong cuốn sách này. Ví dụ, ISO 9000 được thảo luận trong Chương 3.

ISO không phải là một từ viết tắt. Nó có nguồn gốc từ tiếng Hy Lạp - , Ý nghĩa*bình đẳng*, các gốc của tiền tố tiếng Anh*iso-* được tìm thấy trong các từ chẳng hạn như*đồng vị*,*isobar*, và *cân bằng*. Tổ chức Tiêu chuẩn hoá Quốc tế đã chọn ISO làm tên viết tắt để tránh có nhiều từ viết tắt phát sinh từ việc dịch tên "Tổ chức Tiêu chuẩn hoá Quốc tế" sang các ngôn ngữ của các nước thành viên khác nhau. Thay vào đó, để đạt được tiêu chuẩn quốc tế, một dạng viết tắt phổ biến của tên nó đã được chọn.

và Ủy ban Kỹ thuật Điện Quốc tế (IEC). Nghĩa là, bảo trì là quá trình xảy ra khi "phần mềm trải qua các sửa đổi đối với mã và tài liệu liên quan do sự cố hoặc nhu cầu cải tiến hoặc thích ứng" [ISO / IEC 12207, 1995]. Về mặt này**định nghĩa hoạt động**, bảo trì xảy ra bất cứ khi nào một lỗi được khắc phục hoặc các yêu cầu thay đổi, bất kể việc này diễn ra trước hay sau khi lắp đặt sản phẩm. Sau đó, Viện Kỹ sư Điện và Điện tử (IEEE) và Liên minh Công nghiệp Điện tử (EIA) đã thông qua định nghĩa này [IEEE / EIA 12207.0-1996, 1998] khi các tiêu chuẩn IEEE được sửa đổi để phù hợp với ISO / IEC 12207. (Xem Chỉ trong trường hợp Bạn muốn biết Hộp 1.4 để biết thêm về ISO.)

Trong cuốn sách này, thuật ngữ*bảo trì giao hàng sau*đề cập đến định nghĩa IEEE 1990 về bảo trì như bất kỳ thay đổi nào đối với phần mềm sau khi nó đã được phân phối và cài đặt trên máy tính của khách hàng, và*bảo trì hiện đại*hoặc chỉ**Sự bảo trì**đề cập đến định nghĩa ISO / IEC 1995 về các hoạt động khắc phục, hoàn thiện hoặc thích ứng được thực hiện bất kỳ lúc nào. Do đó, bảo trì sau giao hàng là một tập hợp con của bảo trì (hiện đại).

1.3.2Tầm quan trọng của việc bảo trì sau giao hàng

Đôi khi người ta nói rằng chỉ những sản phẩm phần mềm xấu mới được bảo trì sau khi phân phối. Thực tế thì ngược lại: Sản phẩm xấu thì vứt đi, ngược lại sản phẩm tốt được sửa chữa và nâng cao trong 10, 15, thậm chí 20 năm. Hơn nữa, một sản phẩm phần mềm là một mô hình của thế giới thực và thế giới thực luôn thay đổi. Do đó, phần mềm phải được bảo trì liên tục để nó luôn phản ánh chính xác thế giới thực.

Ví dụ: nếu thuế suất bán hàng thay đổi từ 6 đến 7 phần trăm, hầu hết mọi sản phẩm phần mềm liên quan đến việc mua hoặc bán đều phải thay đổi. Giả sử sản phẩm chứa câu lệnh C ++

const floatsalesTax - 6,0;

hoặc câu lệnh Java tương đương

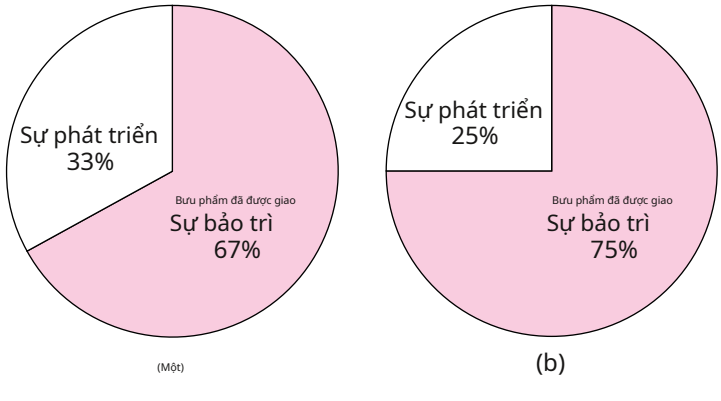
công khai tĩnh cuối cùng floatthuế doanh thu -**(flyến mạch)**6,0;

tuyên bố rằngthuế doanh thulà một hằng số dấu phẩy động được khởi tạo thành giá trị6,0.Trong trường hợp này, việc bảo trì tương đối đơn giản. Với sự hỗ trợ của trình soạn thảo văn bản, giá trị6.0 được thay thế bởi7.0và mã được biên dịch lại và liên kết lại. Tuy nhiên, nếu thay vì sử dụng tên thuế doanh thu,giá trị thực tế6.0đã được sử dụng trong sản phẩm ở bất cứ nơi nào giá trị của thuế bán hàng được viện dẫn, khi đó một sản phẩm như vậy cực kỳ khó sửa đổi. Ví dụ, có thể có các lần xuất hiện của giá trị6.0trong mã nguồn sẽ được thay đổi thành7.0nhưng bị bỏ qua hoặc các trường hợp của 6.0không đề cập đến thuế bán hàng nhưng được thay đổi không chính xác thành 7,0.Việc tìm ra những lỗi này hầu như luôn luôn khó khăn và tốn thời gian. Trên thực tế, với một số phần mềm, về lâu dài, việc vứt bỏ sản phẩm và mã hóa lại nó có thể ít tốn kém hơn thay vì cố gắng xác định hằng số nào trong số nhiều hằng số cần được thay đổi và cách thực hiện sửa đổi.

Thế giới thực thời gian thực cũng liên tục thay đổi. Các tên lửa mà máy bay chiến đấu phản lực trang bị có thể được thay thế bằng một mẫu mới, yêu cầu thay đổi thành phần điều khiển vũ khí của hệ thống điện tử hàng không liên quan. Một động cơ sáu xi-lanh sẽ được cung cấp như một tùy chọn trong một chiếc ô tô bốn xi-lanh phổ biến; điều này ngụ ý thay đổi các máy tính trên tàu điều khiển hệ thống phun nhiên liệu, thời gian, v.v.

Nhưng chỉ có bao nhiêu thời gian (= tiền bạc) được dành cho việc bảo trì giao hàng sau? Biểu đồ tròn trong Hình 1.3 (a) cho thấy rằng, khoảng 40 năm trước, khoảng 2/3 tổng chi phí phần mềm dành cho bảo trì sau giao hàng; dữ liệu thu được bằng cách lấy thông tin trung bình từ nhiều nguồn khác nhau, bao gồm [Elshoff, 1976], [Daly, 1977], [Zelkowitz, Shaw, và Gannon, 1979], và [Boehm, 1981]. Dữ liệu mới hơn cho thấy rằng một tỷ lệ thậm chí còn lớn hơn được dành cho bảo trì sau giao hàng. Nhiều tổ chức dành 70–80 phần trăm hoặc hơn ngân sách phần mềm của họ để bảo trì sau giao hàng [Yourdon, 1992; Hatton, 1998], như trong Hình 1.3 (b).

Đáng ngạc nhiên là tỷ lệ phần trăm chi phí trung bình của các giai đoạn phát triển cổ điển hầu như không thay đổi. Điều này được thể hiện trong Hình 1.4, so sánh dữ liệu được sử dụng để lấy Hình 1.3 (a) với dữ liệu gần đây hơn về 132 dự án Hewlett-Packard [Grady, 1994].



HÌNH 1.3 Gần đúng chi phí trung bình phần trăm của sự phát triển và giao hàng sau Sự bảo trì (a) giữa 1976 và 1981 và (b) giữa 1992 và 1998.

HÌNH 1.4 So sánh tỷ lệ phần trăm chi phí trung bình gần đúng của các giai đoạn phát triển cổ điển cho các dự án khác nhau từ năm 1976 đến 1981 và cho 132 dự án Hewlett-Packard gần đây hơn.

	Các dự án khác nhau giữa năm 1976 và 1981	132 Thêm gần đây Dự án Hewlett-Packard
Yêu cầu và phân tích	21%	18%
(đặc điểm kỹ thuật) các giai đoạn		
Giai đoạn thiết kế	18	19
Giai đoạn thực hiện		
Mã hóa (bao gồm cả thử nghiệm đơn vị)	36	34
Hội nhập	24	29

Bây giờ hãy xem xét lại tổ chức phần mềm hiện đang sử dụng kỹ thuật mã hóa $CT_{cũ}$ mà học CT đó$_{Mới}$ sẽ giảm thời gian viết mã 10 phần trăm. Ngay cả khi $CT_{Mới}$ không có ảnh hưởng xấu đến việc bảo trì, một người quản lý phần mềm sắc sảo sẽ suy nghĩ kỹ trước khi thay đổi các phương pháp viết mã. Toàn bộ nhân viên phải được đào tạo lại, mua các công cụ phát triển phần mềm mới, và có thể thuê thêm các nhân viên có kinh nghiệm về kỹ thuật mới. Tất cả chi phí và sự gián đoạn này phải được chịu đựng để giảm chi phí phần mềm nhiều nhất là 0,85% bởi vì, như thể hiện trong Hình 1.3 (b) và 1.4, mã hóa cùng với kiểm thử đơn vị chỉ chiếm trung bình 34% của 25% hoặc 8,5%. tổng chi phí phần mềm.

Bây giờ, giả sử một kỹ thuật mới giúp giảm 10% chi phí bảo trì sau giao hàng được phát triển. Điều này có lẽ nên được giới thiệu ngay lập tức, vì trung bình, nó sẽ giảm chi phí tổng thể xuống 7,5 phần trăm. Chi phí liên quan đến việc thay đổi sang kỹ thuật này là một cái giá nhỏ phải trả cho khoản tiết kiệm tổng thể lớn như vậy.

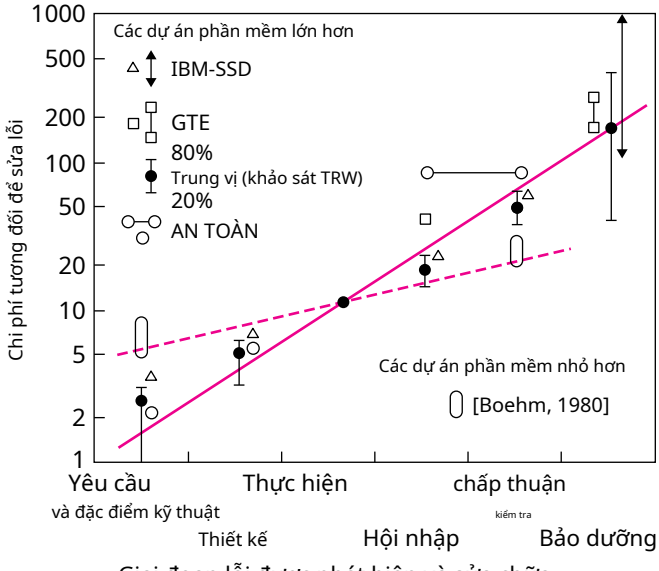
Bởi vì bảo trì sau giao hàng rất quan trọng, một khía cạnh chính của kỹ thuật phần mềm bao gồm các kỹ thuật, công cụ và thực hành đó dẫn đến giảm chi phí bảo trì sau giao hàng.

1.4 Các yêu cầu, phân tích và các khía cạnh thiết kế

Các chuyên gia phần mềm là con người và do đó đôi khi mắc lỗi trong khi phát triển sản phẩm. Kết quả là, sẽ có một lỗi trong phần mềm. Nếu sai sót được thực hiện trong khi đưa ra các yêu cầu, lỗi dẫn đến có thể cũng sẽ xuất hiện trong các thông số kỹ thuật, thiết kế và mã. Rõ ràng, chúng ta sửa lỗi càng sớm thì càng tốt.

Chi phí tương đối của việc sửa lỗi ở các giai đoạn khác nhau trong vòng đời phần mềm cổ điển được thể hiện trong Hình 1.5 [Boehm, 1981]. Hình này phản ánh dữ liệu từ IBM [Fagan, 1974], GTE [Daly, 1977], dự án Safeguard [Stephenson, 1976] và một số dự án TRW nhỏ hơn [Boehm, 1980]. Đường liền nét trong Hình 1.5 là phù hợp nhất cho dữ liệu liên quan đến các dự án lớn hơn và đường đứt nét là phù hợp nhất cho các dự án nhỏ hơn. Đối với mỗi giai đoạn của vòng đời phần mềm cổ điển, chi phí tương đối tương ứng để phát hiện và sửa lỗi

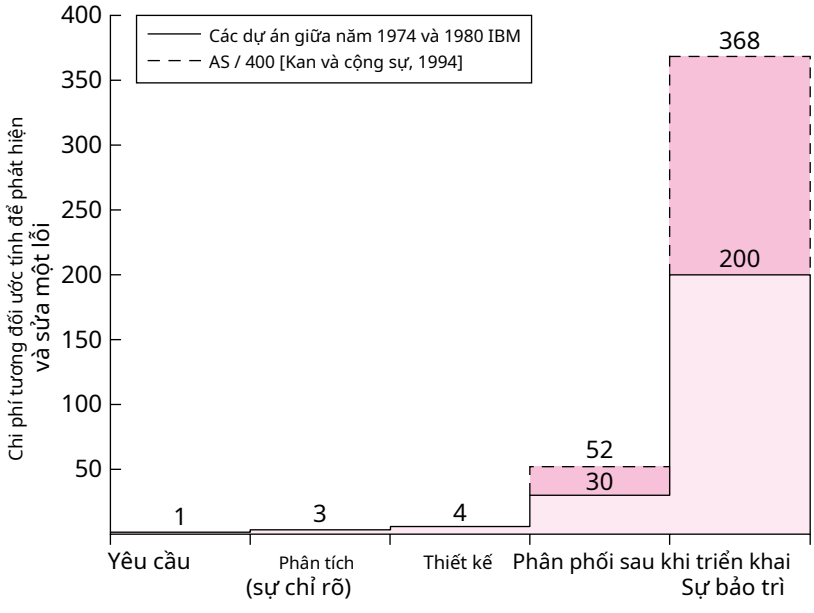
HÌNH 1.5 Chi phí sửa lỗi tương đối ở mỗi giai đoạn của vòng đời phần mềm cổ điển. Đường liền nét là phù hợp nhất cho dữ liệu liên quan đến các dự án phần mềm lớn hơn và đường đứt nét là phù hợp nhất cho các dự án phần mềm nhỏ hơn. (Barry Boehm, *Kinh tế Kỹ thuật Phần mềm,* © 1981, tr. 40. Được điều chỉnh bởi sự cho phép của Prentice Hall, Inc., Englewood Cliffs, NJ.)



lỗi được mô tả trong Hình 1.6. Mỗi bước trên đường liền nét trong Hình 1.6 được xây dựng bằng cách lấy điểm tương ứng trên đường thẳng liền nét của Hình 1.5 và vẽ dữ liệu trên thang tuyến tính.

Giả sử tốn 40 đô la để phát hiện và sửa lỗi ic cụ thể trong giai đoạn thiết kế. Từ đường liền nét trong Hình 1.6 (các dự án từ năm 1974 đến 1980), lỗi tương tự đó sẽ chỉ tốn khoảng 30 đô la đến f ix trong giai đoạn phân tích. Nhưng trong quá trình bảo trì sau giao hàng, lỗi đó sẽ tốn khoảng 2000 đô la để phát hiện và sửa chữa. Dữ liệu mới hơn cho thấy giờ đây việc phát hiện lỗi sớm thậm chí còn quan trọng hơn. Đường đứt nét trong Hình 1.6 cho thấy chi phí phát hiện và sửa lỗi trong quá trình phát triển phần mềm hệ thống cho IBM AS / 400 [Kan và cộng sự, 1994]. Trung bình, cùng một lỗi sẽ có giá từ $ 3680 đến f ix trong quá trình bảo trì phần mềm AS / 400 sau giao hàng.

Lý do khiến chi phí sửa lỗi tăng quá cao có liên quan đến việc phải làm gì để sửa lỗi. Trong giai đoạn đầu của vòng đời phát triển, về cơ bản sản phẩm chỉ tồn tại trên giấy và việc sửa lỗi có thể chỉ đơn giản là thực hiện thay đổi đối với tài liệu. Thái cực khác là một sản phẩm đã được giao cho khách hàng. Ít nhất, sửa lỗi tại thời điểm đó có nghĩa là chỉnh sửa mã, biên dịch lại và liên kết lại, sau đó kiểm tra cẩn thận xem vấn đề đã được giải quyết hay chưa. Tiếp theo, điều quan trọng là phải kiểm tra xem việc thực hiện thay đổi có tạo ra vấn đề mới ở nơi khác trong sản phẩm hay không. Tất cả các tài liệu liên quan, bao gồm cả sách hướng dẫn, cần được cập nhật. Cuối cùng, sản phẩm sửa chữa phải được chuyển giao



HÌNH 1.6
Đường liền nét miêu tả điểm trên đường liền mạch của Hình 1.5 âm mưu trên một quy mô tuyến tính. Dấu gạch ngang mô tả dòng dữ liệu mới hơn.

và được cài đặt lại. Đạo lý của câu chuyện là thế này: Chúng ta phải sớm tìm ra lỗi lầm, nếu không sẽ phải trả giá đắt. Do đó, chúng tôi nên sử dụng các kỹ thuật để phát hiện lỗi trong các giai đoạn yêu cầu và phân tích (đặc điểm kỹ thuật).

Cần có thêm những kỹ thuật như vậy. Các nghiên cứu đã chỉ ra [Boehm, 1979] rằng từ 60 đến 70 phần trăm tất cả các lỗi được phát hiện trong các dự án lớn là các lỗi về yêu cầu, phân tích hoặc thiết kế. Các kết quả mới hơn từ việc kiểm tra cho thấy ưu thế này của các yêu cầu, phân tích hoặc lỗi thiết kế (kiểm tra là một nhóm kiểm tra tỉ mỉ tài liệu, như được mô tả trong Phần 6.2.3). Trong 203 lần kiểm tra phần mềm Phòng thí nghiệm Sức đẩy Phản lực cho chương trình không gian liên hành tinh không người lái của NASA, trung bình, khoảng 1,9 lỗi được phát hiện trên mỗi trang của tài liệu đặc tả, 0,9 lỗi trên mỗi trang của thiết kế, nhưng chỉ có 0,3 lỗi trên mỗi trang mã [Kelly, Sherif, và Hops, 1992].

Do đó, điều quan trọng là chúng tôi phải cải thiện các yêu cầu, kỹ thuật phân tích và thiết kế của mình, không chỉ để các lỗi có thể được phát hiện sớm nhất có thể mà còn vì các lỗi về yêu cầu, phân tích và thiết kế chiếm một tỷ lệ lớn trong tất cả các lỗi. Cũng như ví dụ trong Phần 1.3 cho thấy rằng giảm 10% chi phí bảo trì sau giao hàng sẽ giảm chi phí tổng thể khoảng 7,5%, giảm 10% yêu cầu, phân tích và lỗi thiết kế làm giảm 6–7% tổng số lỗi.

Việc có rất nhiều lỗi được đưa ra sớm trong vòng đời phần mềm làm nổi bật một khía cạnh quan trọng khác của kỹ thuật phần mềm: các kỹ thuật mang lại các yêu cầu, thông số kỹ thuật và thiết kế tốt hơn.

Hầu hết phần mềm được sản xuất bởi một nhóm kỹ sư phần mềm chứ không phải bởi một cá nhân chịu trách nhiệm về mọi khía cạnh của vòng đời phát triển và bảo trì. Bây giờ chúng ta xem xét các tác động của điều này.

1.5 Các khía cạnh phát triển nhóm

Giá thành của phần cứng tiếp tục giảm nhanh chóng. Một chiếc máy tính lớn của những năm 1950 có giá hơn một triệu đô la trước khi giảm phát kém hơn đáng kể về mọi mặt so với một chiếc máy tính xách tay ngày nay có giá dưới 1000 đô la. Do đó, các tổ chức có thể dễ dàng mua được phần cứng có thể chạy các sản phẩm lớn, tức là các sản phẩm quá lớn (hoặc quá phức tạp) để một người thực hiện trong thời gian giới hạn cho phép. Ví dụ: nếu một sản phẩm phải được giao trong vòng 18 tháng nhưng sẽ mất 15 năm để hoàn thành một chuyên gia phần mềm duy nhất, thì sản phẩm đó phải được phát triển bởi một nhóm. Tuy nhiên, phát triển nhóm dẫn đến các vấn đề giao tiếp giữa các thành phần mã và các vấn đề giao tiếp giữa các thành viên trong nhóm.

Ví dụ, mô-đun mã Jeff và JulietPvàq,tương ứng, trong đó mô-đunP mô-đun cuộc gọiq.Khi Jeff viết mãP,anh ấy chèn một cuộc gọi đếnqvới năm đối số trong danh sách đối số. Mã Julietqvới năm đối số, nhưng theo thứ tự khác với đối số của Jeff. Một số công cụ phần mềm, chẳng hạn như trình thông dịch và trình tải Java, hoặc*xơ vải*đối với C (Mục 8.11.4), phát hiện vi phạm kiểu như vậy nhưng chỉ khi các đối số được hoán đổi thuộc loại khác nhau; nếu chúng cùng loại, thì vấn đề có thể không được phát hiện trong một thời gian dài. Có thể tranh luận rằng đây là một vấn đề thiết kế, và nếu các mô-đun được thiết kế cẩn thận hơn, vấn đề này đã không xảy ra. Điều đó có thể đúng, nhưng trên thực tế, một thiết kế thường bị thay đổi sau khi bắt đầu viết mã và thông báo về thay đổi có thể không được gửi cho tất cả các thành viên của nhóm phát triển. Do đó, khi một thiết kế ảnh hưởng đến hai hoặc nhiều lập trình viên đã bị thay đổi, giao tiếp kém có thể dẫn đến các vấn đề giao diện mà Jeff và Juliet đã trải qua. Loại vấn đề này ít có khả năng xảy ra hơn khi chỉ có một cá nhân chịu trách nhiệm về mọi khía cạnh của sản phẩm,

Nhưng các vấn đề về giao diện chỉ là phần nổi của tảng băng chìm khi nói đến các vấn đề có thể phát sinh khi phần mềm được phát triển bởi các nhóm. Trừ khi nhóm được tổ chức đúng cách, bạn có thể lãng phí một lượng thời gian không đáng kể trong các cuộc hội thảo giữa các thành viên trong nhóm. Giả sử rằng một lập trình viên mất 1 năm để hoàn thành một sản phẩm. Nếu cùng một nhiệm vụ được giao cho một nhóm gồm sáu lập trình viên, thời gian hoàn thành nhiệm vụ thường gần hơn 1 năm so với 2 tháng dự kiến và chất lượng của mã kết quả cũng có thể thấp hơn nếu toàn bộ nhiệm vụ đã được giao. cho một cá nhân (xem Phần 4.1). Bởi vì một tỷ lệ đáng kể phần mềm ngày nay được phát triển và duy trì bởi các nhóm, phạm vi kỹ thuật phần mềm phải bao gồm các kỹ thuật để đảm bảo rằng các nhóm được tổ chức và quản lý đúng cách.

Như đã được trình bày trong các phần trước, phạm vi của kỹ thuật phần mềm là rất rộng. Nó bao gồm mọi bước của vòng đời phần mềm, từ các yêu cầu cho đến khi ngừng cung cấp dịch vụ sau phân phối. Nó cũng bao gồm các khía cạnh con người, chẳng hạn như tổ chức nhóm; khía cạnh kinh tế; và các khía cạnh pháp lý, chẳng hạn như luật bản quyền. Tất cả các khía cạnh này được kết hợp một cách ngầm định trong định nghĩa về kỹ thuật phần mềm được đưa ra ở đầu chương này, rằng kỹ thuật phần mềm là một ngành có mục đích là sản xuất phần mềm không có lỗi được phân phối đúng thời hạn, trong phạm vi ngân sách và đáp ứng nhu cầu của người dùng.

Chúng ta quay trở lại các giai đoạn cổ điển của Hình 1.2 để hỏi tại sao không có giai đoạn lập kế hoạch, thử nghiệm hoặc tài liệu.

1.6 Tại sao không có giai đoạn lập kế hoạch

Rõ ràng là không thể phát triển một sản phẩm phần mềm mà không có kế hoạch. Do đó, điều cần thiết là phải có**giai đoạn lập kế hoạch**khi bắt đầu dự án.

Điểm mấu chốt là, cho đến khi biết chính xác cái gì sẽ được phát triển, không có cách nào có thể lập được một kế hoạch chi tiết và chính xác. Do đó, ba loại hoạt động lập kế hoạch diễn ra khi một sản phẩm phần mềm được phát triển bằng cách sử dụng mô hình cổ điển:

1. Khi bắt đầu dự án, lập kế hoạch sơ bộ diễn ra để quản lý các yêu cầu và giai đoạn phân tích.
2. Một khi những gì sẽ được phát triển được biết một cách chính xác,*kế hoạch quản lý dự án phần mềm*(SPMP) được vẽ lên. Điều này bao gồm ngân sách, yêu cầu về nhân sự và lịch trình chi tiết. Sớm nhất chúng tôi có thể lập kế hoạch quản lý dự án là khi tài liệu đặc tả đã được khách hàng chấp thuận, tức là vào cuối giai đoạn phân tích. Cho đến thời điểm đó, việc lập kế hoạch phải là sơ bộ và từng phần.
3. Trong suốt dự án, ban quản lý cần giám sát SPMP và theo dõi mọi sai lệch so với kế hoạch.

Ví dụ: giả sử SPMP cho một dự án cụ thể nói rằng toàn bộ dự án sẽ mất 16 tháng và giai đoạn thiết kế sẽ mất 4 tháng trong số đó. Sau một năm, ban lãnh đạo nhận thấy rằng toàn bộ dự án dường như đang tiến triển chậm hơn nhiều so với dự đoán. Một cuộc điều tra chi tiết cho thấy, cho đến nay, 8 tháng đã được dành cho giai đoạn thiết kế, vẫn còn lâu mới hoàn thành. Dự án gần như chắc chắn sẽ phải bị bỏ dở, và số tiền bỏ ra cho đến nay bị lãng phí. Thay vào đó, ban quản lý nên theo dõi tiến độ theo từng giai đoạn và nhận thấy, sau nhiều nhất là 2 tháng, một vấn đề nghiêm trọng trong giai đoạn thiết kế. Vào thời điểm đó, một quyết định có thể được đưa ra để tiến hành như thế nào là tốt nhất. Bước đầu tiên thông thường trong tình huống như vậy là gọi một chuyên gia tư vấn để xác định xem dự án có khả thi hay không và xác định xem nhóm thiết kế có đủ năng lực để thực hiện nhiệm vụ hay không hoặc rủi ro trong quá trình tiến hành là quá lớn. Dựa trên báo cáo của nhà tư vấn, các giải pháp thay thế khác nhau hiện đang được xem xét, bao gồm giảm phạm vi sản phẩm mục tiêu, sau đó thiết kế và triển khai một sản phẩm ít tham vọng hơn. Chỉ khi tất cả các phương án thay thế khác được coi là không thể thực hiện được thì dự án mới phải bị hủy bỏ. Trong trường hợp của một dự án cụ thể, việc hủy bỏ này sẽ diễn ra sớm hơn khoảng 6 tháng nếu ban quản lý theo dõi kế hoạch chặt chẽ, tiết kiệm được một khoản tiền đáng kể. bao gồm giảm phạm vi của sản phẩm mục tiêu, sau đó thiết kế và triển khai một sản phẩm ít tham vọng hơn. Chỉ khi tất cả các phương án thay thế khác được coi là không thể thực hiện được thì dự án mới phải bị hủy bỏ. Trong trường hợp của một dự án cụ thể, việc hủy bỏ này sẽ diễn ra sớm hơn khoảng 6 tháng nếu ban quản lý theo dõi kế hoạch chặt chẽ, tiết kiệm được một khoản tiền đáng kể. bao gồm giảm phạm vi của sản phẩm mục tiêu, sau đó thiết kế và triển khai một sản phẩm ít tham vọng hơn. Chỉ khi tất cả các phương án thay thế khác được coi là không thể thực hiện được thì dự án mới phải bị hủy bỏ. Trong trường hợp của một dự án cụ thể, việc hủy bỏ này sẽ diễn ra sớm hơn khoảng 6 tháng nếu ban quản lý theo dõi kế hoạch chặt chẽ, tiết kiệm được một khoản tiền đáng kể.

Tóm lại, không có giai đoạn lập kế hoạch riêng biệt. Thay vào đó, các hoạt động lập kế hoạch được thực hiện trong suốt vòng đời. Tuy nhiên, có những lúc các hoạt động lập kế hoạch chiếm ưu thế. Chúng bao gồm việc bắt đầu dự án (lập kế hoạch sơ bộ) và trực tiếp sau khi tài liệu đặc tả đã được khách hàng ký vào (kế hoạch quản lý dự án phần mềm).

1.7 Tại sao không có giai đoạn thử nghiệm

Điều cần thiết là phải kiểm tra tỉ mỉ một sản phẩm phần mềm sau khi nó đã được phát triển. Theo đó, có thể đặt ra câu hỏi tại sao không có giai đoạn thử nghiệm sau khi sản phẩm đã được triển khai.

Thật không may, việc kiểm tra một sản phẩm phần mềm khi nó đã sẵn sàng để giao cho khách hàng là quá muộn. Ví dụ, nếu có lỗi trong tài liệu đặc tả, lỗi này sẽ được chuyển sang thiết kế và triển khai. Có những lúc trong quy trình phần mềm khi việc kiểm thử được thực hiện gần như loại trừ hoàn toàn các hoạt động khác. Điều này xảy ra vào cuối mỗi giai đoạn (**xác minh**) và đặc biệt đúng trước khi sản phẩm được bàn giao cho khách hàng (**Thẩm định**). Mặc dù có những thời điểm kiểm tra chiếm ưu thế, nhưng không bao giờ nên có những lúc không thực hiện kiểm tra. Nếu thử nghiệm được coi là một phần riêng biệt (**thử nghiệm**)**giai đoạn**, thì có một nguy cơ rất thực tế là việc thử nghiệm sẽ không được thực hiện liên tục trong suốt mọi giai đoạn của quá trình phát triển và bảo trì sản phẩm.

Nhưng ngay cả điều này là không đủ. Điều cần thiết là liên tục kiểm tra một sản phẩm phần mềm. Việc kiểm tra tỉ mỉ sẽ tự động đi kèm với mọi hoạt động phát triển và bảo trì phần mềm. Giai đoạn thử nghiệm riêng biệt không tương thích với mục tiêu đảm bảo rằng một sản phẩm phần mềm luôn không có lỗi nhất có thể.

Mọi tổ chức phát triển phần mềm nên có một nhóm độc lập có trách nhiệm chính là đảm bảo rằng sản phẩm được giao là thứ mà khách hàng cần và sản phẩm đã được xây dựng đúng theo mọi cách. Nhóm này được gọi là *Đảm bảo chất lượng phần mềm* (SQA) nhóm. Các **phẩm chất** của phần mềm là mức độ đáp ứng các thông số kỹ thuật của phần mềm. Chất lượng và đảm bảo chất lượng phần mềm được mô tả chi tiết hơn trong Chương 6, cũng như vai trò của SQA trong việc thiết lập và thực thi các tiêu chuẩn.

1.8 Tại sao không có giai đoạn lập tài liệu

Cũng như không bao giờ nên có một giai đoạn lập kế hoạch hoặc giai đoạn thử nghiệm riêng biệt, cũng không bao giờ nên có một giai đoạn **giai đoạn tài liệu**. Ngược lại, tại mọi thời điểm, tài liệu của một sản phẩm phần mềm phải đầy đủ, chính xác và cập nhật. Ví dụ, trong giai đoạn phân tích, tài liệu đặc điểm kỹ thuật phải phản ánh phiên bản hiện tại của thông số kỹ thuật và điều này cũng đúng với các giai đoạn khác.

1. Một lý do tại sao điều cần thiết là đảm bảo rằng tài liệu luôn được cập nhật là doanh thu lớn về nhân sự trong ngành công nghiệp phần mềm. Ví dụ, giả sử rằng tài liệu thiết kế không được lưu giữ hiện tại và người thiết kế trưởng rời đi để nhận một công việc khác. Hiện tại, việc cập nhật tài liệu thiết kế để phản ánh tất cả những thay đổi được thực hiện trong khi thiết kế hệ thống là vô cùng khó khăn.
2. Hầu như không thể thực hiện các bước của một giai đoạn cụ thể trừ khi tài liệu của giai đoạn trước đó đầy đủ, chính xác và cập nhật. Ví dụ, một tài liệu đặc tả không đầy đủ chắc chắn phải dẫn đến một thiết kế không đầy đủ và sau đó triển khai không đầy đủ.
3. Hầu như không thể kiểm tra xem một sản phẩm phần mềm có hoạt động bình thường hay không trừ khi có sẵn các tài liệu nêu rõ sản phẩm phần mềm đó hoạt động như thế nào.
4. Việc bảo trì hầu như không thể thực hiện được trừ khi có một bộ tài liệu đầy đủ và chính xác mô tả chính xác những gì mà phiên bản hiện tại của sản phẩm thực hiện.

Do đó, cũng như không có giai đoạn lập kế hoạch riêng biệt hoặc giai đoạn thử nghiệm, không có giai đoạn lập tài liệu riêng. Thay vào đó, lập kế hoạch, kiểm tra và tài liệu phải là các hoạt động đi kèm với tất cả các hoạt động khác trong khi một sản phẩm phần mềm đang được xây dựng.

Bây giờ chúng ta kiểm tra mô hình hướng đối tượng.

1.9 Mô hình hướng đối tượng

Trước năm 1975, hầu hết các tổ chức phần mềm không sử dụng các kỹ thuật cụ thể; mỗi cá nhân đã làm việc theo cách riêng của mình. Những bước đột phá lớn đã được thực hiện trong khoảng từ năm 1975 đến năm 1985, với sự phát triển của cái gọi là**có cấu trúc**hoặc là**mô hình cổ điển**. Các kỹ thuật cấu thành mô hình cổ điển bao gồm phân tích hệ thống có cấu trúc (Phần 12.3), phân tích luồng dữ liệu (Phần 14.3), lập trình có cấu trúc và kiểm tra có cấu trúc (Phần 15.13.2). Những kỹ thuật này có vẻ rất hứa hẹn khi lần đầu tiên được sử dụng. Tuy nhiên, thời gian trôi qua, họ tỏ ra kém thành công hơn ở hai khía cạnh:

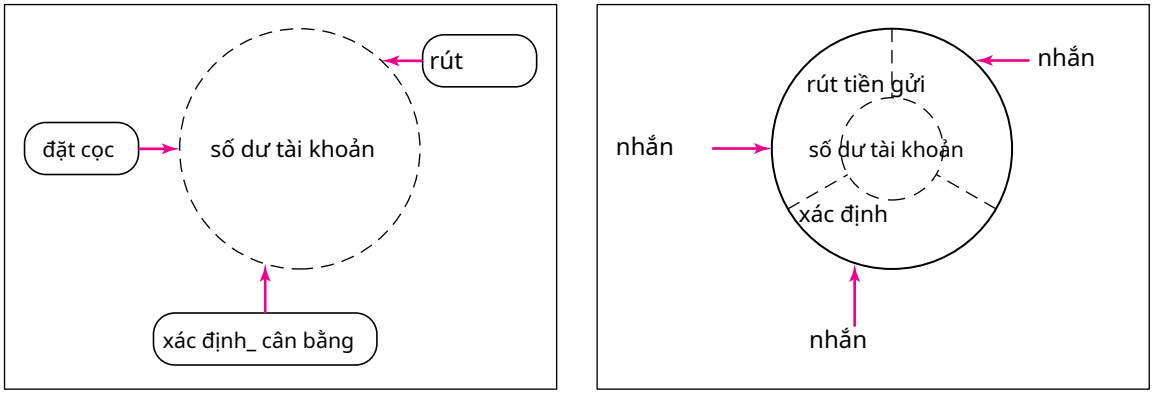
1. Các kỹ thuật đôi khi không thể đối phó với kích thước ngày càng tăng của các sản phẩm phần mềm. Có nghĩa là, các kỹ thuật cổ điển là phù hợp khi xử lý các sản phẩm quy mô nhỏ (thường là 5000 dòng mã) hoặc thậm chí các sản phẩm quy mô vừa với 50.000 dòng mã. Tuy nhiên, ngày nay, các sản phẩm quy mô lớn gồm 500.000 dòng mã là tương đối phổ biến; kể cả những sản phẩm có dòng mã từ 5 triệu trở lên cũng không bị coi là bất thường. Tuy nhiên, các kỹ thuật cổ điển thường không thể mở rộng quy mô đủ để xử lý sự phát triển của các sản phẩm lớn hơn ngày nay.
2. Mô hình cổ điển không đáp ứng được kỳ vọng trước đó trong quá trình bảo trì sau giao hàng. Một động lực chính thúc đẩy sự phát triển của mô hình cổ điển cách đây khoảng 40 năm là trung bình, 2/3 ngân sách phần mềm được dành cho việc bảo trì sau giao hàng (xem Hình 1.3). Thật không may, mô hình cổ điển đã không giải quyết được vấn đề này; như đã chỉ ra trong Phần 1.3.2, nhiều tổ chức vẫn dành 70–80 phần trăm hoặc hơn thời gian và công sức của họ cho việc bảo trì sau giao hàng [Yourdon, 1992; Hatton, 1998].

Một lý do chính cho sự thành công hạn chế của mô hình cổ điển là các kỹ thuật cổ điển hoặc là hướng hoạt động hoặc hướng thuộc tính (dữ liệu) nhưng không phải cả hai. Các thành phần cơ bản của một sản phẩm phần mềm là các hoạt động của sản phẩm và các thuộc tính mà các hoạt động đó vận hành. Ví dụ,verify_average_height[1]là một phép toán hoạt động trên một tập hợp các độ cao (thuộc tính) và trả về giá trị trung bình của các độ cao đó (thuộc tính). Một số kỹ thuật cổ điển, chẳng hạn như phân tích luồng dữ liệu (Phần 14.3), là hướng hoạt động. Đó là, các kỹ thuật đó tập trung vào các hoạt động của sản phẩm; các thuộc tính có tầm quan trọng thứ yếu. Ngược lại, các kỹ thuật như phát triển hệ thống Jackson (Phần 14.5) là hướng thuộc tính. Sự nhấn mạnh ở đây là các thuộc tính; các hoạt động hoạt động trên các thuộc tính ít quan trọng hơn.

Ngược lại, mô hình hướng đối tượng coi cả thuộc tính và hoạt động đều quan trọng như nhau. Một cách đơn giản để xem một đối tượng là một tạo tác phần mềm thống nhất kết hợp cả các thuộc tính và các hoạt động được thực hiện trên các thuộc tính (an **tạo tác**là một thành phần của sản phẩm phần mềm, chẳng hạn như tài liệu đặc tả, mô-đun mã hoặc sách hướng dẫn). Định nghĩa này về một đối tượng chưa hoàn chỉnh và được bổ sung sau trong cuốn sách, một lần*di sản*đã được xác định (Phần 7.8). Tuy nhiên, định nghĩa nắm bắt được phần lớn bản chất của một đối tượng.

[1]Trong cuốn sách này, tên của một biến trong một sản phẩm phần mềm cổ điển được viết bằng cách sử dụng quy ước cổ điển là tách các phần của tên biến bằng dấu gạch dưới, ví dụ:this_is_a_classical_variable.Một biến trong sản phẩm phần mềm hướng đối tượng được viết theo quy ước hướng đối tượng là sử dụng chữ hoa để đánh dấu phần bắt đầu phần mới của tên biến; Ví dụ,thisIsAnObjectOrientedVariable.

HÌNH 1.7 So sánh việc triển khai tài khoản ngân hàng bằng cách sử dụng (a) mô hình cổ điển và (b) mô hình hướng đối tượng. Đường liền nét màu đen bao quanh đối tượng biểu thị chi tiết về cáchsố dư tài khoảnđược thực hiện không được biết bên ngoài đối tượng.



(Một)

(b)

Tài khoản ngân hàng là một ví dụ về một đối tượng (xem Hình 1.7). Thành phần thuộc tính của đối tượng làsố dư tài khoản.Các hoạt động có thể được thực hiện trên số dư tài khoản đó bao gồmđặt cọctiền trong tài khoản,rúttiền từ tài khoản, và xác địnhBalance.Đối tượng tài khoản ngân hàng kết hợp một thuộc tính với ba thao tác được thực hiện trên thuộc tính đó trong một tạo tác duy nhất. Theo quan điểm của mô hình cổ điển, một sản phẩm giao dịch với ngân hàng sẽ phải kết hợp một thuộc tính, số dư tài khoản,và ba hoạt động,gửi tiền, rút tiền,vàxác định_độ cân bằng.

Cho đến nay, dường như có rất ít sự khác biệt giữa hai cách tiếp cận. Tuy nhiên, một điểm mấu chốt là cách thức mà một đối tượng được thực hiện. Cụ thể, chi tiết về cách các thuộc tính của một đối tượng được lưu trữ không được biết từ bên ngoài đối tượng. Đây là một ví dụ về "che giấu thông tin", được thảo luận chi tiết hơn trong Phần 7.6. Trong trường hợp đối tượng tài khoản ngân hàng được hiển thị trong Hình 1.7 (b), phần còn lại của sản phẩm phần mềm biết rằng có một thứ như là số dư trong đối tượng tài khoản ngân hàng, nhưng nó không có ý tưởng về định dạng củasố dư tài khoản.Có nghĩa là, không có kiến thức bên ngoài đối tượng về việc liệu số dư tài khoản được thực hiện dưới dạng số nguyên hay số dấu phẩy động hoặc một trường (thành phần) của một số cấu trúc lớn hơn. Rào cản thông tin bao quanh đối tượng này được biểu thị bằng đường liền nét màu đen trong Hình 1.7 (b), mô tả một triển khai sử dụng mô hình hướng đối tượng. Ngược lại, một đường đứt nét bao quanh số dư tài khoảntrong Hình 1.7 (a), bởi vì tất cả các chi tiết củasố dư tài khoảnđược biết đến với các mô-đun trong quá trình triển khai bằng cách sử dụng mô hình cổ điển và giá trị củasố dư tài khoảndo đó có thể được thay đổi bởi bất kỳ người nào trong số họ.

Quay trở lại Hình 1.7 (b), triển khai hướng đối tượng, nếu khách hàng gửi $ 10 vào tài khoản, thì a**nhắn**được gửi đếnđặt cọcphương thức của đối tượng có liên quan yêu cầu nó tăngsố dư tài khoảnthuộc tính $ 10 (a**phương pháp**là một triển khai của một hoạt động). Cácđặt cọcphương pháp nằm trong đối tượng tài khoản ngân hàng và biết cáchsố dư tài khoảnđược thực thi; điều này được biểu thị bằng đường tròn đứt nét bên trong

sự vật. Nhưng không có thực thể nào bên ngoài đối tượng cần kiến thức này. Rằng ba phương pháp trong Hình 1.7 (b) khiênsố dư tài khoảntừ phần còn lại của sản phẩm tượng trưng cho sự bản địa hóa kiến thức này. Thực tế là các chi tiết triển khai là cục bộ của một đối tượng minh họa điểm đầu tiên trong số nhiều điểm mạnh của mô hình hướng đối tượng:

1. Xem xét bảo trì giao hàng sau. Giả sử rằng sản phẩm ngân hàng đã được xây dựng bằng cách sử dụng mô hình cổ điển. Nếu cách mộtsố dư tài khoảnđược biểu diễn được thay đổi từ (giả sử) một số nguyên thành một trường của cấu trúc, sau đó mọi phần của sản phẩm đó có liên quan đến mộtsố dư tài khoảnphải được thay đổi và những thay đổi này phải được thực hiện một cách nhất quán. Ngược lại, nếu mô hình hướng đối tượng được sử dụng, thì các thay đổi chỉ cần được thực hiện trong chính đối tượng tài khoản ngân hàng. Không phần nào khác của sản phẩm có kiến thức về cáchsố dư tài khoảnđược triển khai, vì vậy không bộ phận nào khác có thể có quyền truy cập vàosố dư tài khoản.Do đó, không có phần nào khác của sản phẩm ngân hàng cần phải thay đổi. Theo đó, mô hình hướng đối tượng giúp bảo trì nhanh hơn và dễ dàng hơn, đồng thời có cơ hội giới thiệu**hồi quy lỗi**(nghĩa là, lỗi do vô tình đưa vào một bộ phận của sản phẩm do hậu quả của việc thực hiện một thay đổi dường như không liên quan đến bộ phận khác của sản phẩm) được giảm đáng kể.
2. Ngoài việc bảo trì, mô hình hướng đối tượng cũng giúp cho việc phát triển dễ dàng hơn. Trong nhiều trường hợp, một đối tượng có một đối tác vật lý. Ví dụ, một đối tượng tài khoản ngân hàng trong sản phẩm ngân hàng tương ứng với một tài khoản ngân hàng thực tế tại ngân hàng mà sản phẩm này đang được triển khai. Như sẽ được trình bày trong Phần B, mô hình hóa đóng một vai trò quan trọng trong mô hình hướng đối tượng. Sự tương ứng chặt chẽ giữa các đối tượng trong sản phẩm và đối tác của chúng trong thế giới thực sẽ dẫn đến phần mềm chất lượng tốt hơn.
3. Các đối tượng được thiết kế tốt là các đơn vị độc lập. Như đã được giải thích, một đối tượng bao gồm cả thuộc tính và các hoạt động được thực hiện trên các thuộc tính. Nếu tất cả các hoạt động được thực hiện trên các thuộc tính của một đối tượng được bao gồm trong đối tượng đó, thì đối tượng đó có thể được coi là một thực thể độc lập về mặt khái niệm. Mọi thứ trong sản phẩm liên quan đến phần của thế giới thực được mô hình hóa bởi đối tượng đó đều có thể được tìm thấy trong chính đối tượng đó. Sự độc lập về khái niệm này đôi khi được gọi là**sự đóng gói** (Mục 7.4). Nhưng có thêm một hình thức độc lập, độc lập về thể chất. Trong một đối tượng được thiết kế tốt, ẩn thông tin đảm bảo rằng các chi tiết triển khai được ẩn khỏi mọi thứ bên ngoài đối tượng đó. Hình thức giao tiếp được phép duy nhất là gửi một thông điệp đến đối tượng để thực hiện một hoạt động cụ thể. Cách thức mà hoạt động được thực hiện hoàn toàn do đối tượng tự chịu trách nhiệm. Vì lý do này, thiết kế hướng đối tượng đôi khi được gọi là **thiết kế hướng đến trách nhiệm**[Wirfs-Brock, Wilkerson và Wiener, 1990] hoặc **thiết kế theo hợp đồng**[Meyer, 1992]. (Để biết một cách nhìn khác về thiết kế theo hướng trách nhiệm, hãy xem Hộp 1.5 Chỉ trong Trường hợp Bạn muốn Biết, lấy từ một ví dụ trong [Budd, 2002].) Một cách khác để xem xét cả đóng gói và ẩn thông tin là các trường hợp tách biệt các mối quan tâm (Mục 5.4).
4. Một sản phẩm được xây dựng bằng cách sử dụng mô hình cổ điển được thực hiện như một tập hợp các mô-đun, nhưng về mặt khái niệm, nó về cơ bản là một đơn vị duy nhất. Đây là một lý do tại sao mô hình cổ điển ít thành công hơn khi áp dụng cho các sản phẩm lớn hơn. Ngược lại, khi mô hình định hướng đối tượng được sử dụng đúng cách, sản phẩm thu được bao gồm một số đơn vị nhỏ hơn, phần lớn độc lập. Mô hình hướng đối tượng làm giảm mức độ phức tạp của một sản phẩm phần mềm và do đó đơn giản hóa cả việc phát triển và bảo trì.

Chỉ trong trường hợp bạn muốn biết — Hộp 1.5

Giả sử rằng bạn sống ở New Orleans và bạn muốn gửi một bó hoa Ngày của Mẹ cho mẹ của bạn ở Chicago. Một chiến lược là tham khảo các trang màu vàng của Chicago (trên World Wide Web), xác định cửa hàng bán hoa nào ở gần căn hộ của mẹ bạn nhất và đặt hàng với người bán hoa đó. Một cách tiện lợi hơn là đặt hoa tại**1-800-flowers.com,**để lại toàn bộ trách nhiệm giao hoa cho công ty đó. Nó không liên quan ở đâu**1-800-flowers.com**được đặt tại địa chỉ thực tế hoặc người bán hoa nào được giao hàng cho bạn. Trong mọi trường hợp, công ty không tiết lộ thông tin đó, một ví dụ của việc che giấu thông tin.

Theo cách tương tự, khi một thông điệp được gửi đến một đối tượng, không chỉ hoàn toàn không liên quan đến cách yêu cầu được thực hiện, mà đơn vị gửi thông điệp thậm chí còn không được phép biết cấu trúc bên trong của đối tượng. Bản thân đối tượng hoàn toàn chịu trách nhiệm về mọi chi tiết của việc thực hiện thông điệp.

5. Mô hình hướng đối tượng thúc đẩy tái sử dụng; bởi vì các đối tượng là các thực thể độc lập, chúng thường có thể được sử dụng trong các sản phẩm trong tương lai (nhưng xem Vấn đề 1.17). Việc tái sử dụng các đối tượng này làm giảm thời gian và chi phí của cả việc phát triển và bảo trì, như đã giải thích trong Chương 8.

Khi mô hình hướng đối tượng được sử dụng, vòng đời phần mềm cổ điển của Hình 1.2 đã được sửa đổi. Hình 1.8 so sánh mô hình vòng đời của mô hình cổ điển với mô hình của mô hình hướng đối tượng.

Sự khác biệt đầu tiên dường như là thuần túy về mặt thuật ngữ; từ*giai đoạn*được sử dụng cho mô hình cổ điển, trong khi*quy trình làm việc*được sử dụng cho mô hình hướng đối tượng. Trên thực tế, như sẽ được giải thích chi tiết trong Chương 2, không có sự tương ứng giữa một giai đoạn và một quy trình làm việc. Ngược lại, hai thuật ngữ hoàn toàn khác biệt, và sự khác biệt này là hình ảnh thu nhỏ sự khác biệt giữa các mô hình vòng đời làm cơ sở cho hai mô hình.

Trong chương này, chúng ta xem xét sự khác biệt khác giữa hai mô hình, vai trò của các mô-đun (trong mô hình cổ điển) so với vai trò của các đối tượng (trong mô hình định hướng đối tượng). Đầu tiên hãy xem xét giai đoạn thiết kế của mô hình cổ điển. Như đã nêu trong Phần 1.3, giai đoạn này được chia thành hai giai đoạn con: thiết kế kiến trúc tiếp theo là thiết kế chi tiết. Trong giai đoạn con thiết kế kiến trúc, sản phẩm được phân tách thành các thành phần, được gọi là*mô-đun*. Sau đó, trong giai đoạn con thiết kế chi tiết, các cấu trúc dữ liệu và thuật toán của từng mô-đun được thiết kế lần lượt. Cuối cùng, trong giai đoạn thực hiện, các mô-đun này được thực hiện.

Nếu mô hình hướng đối tượng được sử dụng thay thế, một trong các bước của quy trình phân tích hướng đối tượng là xác định các lớp. Bởi vì một lớp là một loại mô-đun, thiết kế kiến trúc được thực hiện trong quy trình phân tích hướng đối tượng.

HÌNH 1.8 So sánh vòng đời mô hình của cổ điển mô hình và đối tượng-định hướng mô hình.

Mô hình cổ điển	Mô hình hướng đối tượng
1. Giai đoạn yêu cầu	1. Yêu cầu quy trình làm việc
2. Giai đoạn phân tích (đặc tả)	2. Quy trình phân tích hướng đối tượng
3. Giai đoạn thiết kế	3. Quy trình thiết kế hướng đối tượng
4. Giai đoạn thực hiện	4 . Quy trình thực hiện hướng đối tượng
5. Bảo trì sau giao hàng	5. Bảo trì sau giao hàng
6. Nghỉ hưu	6. Nghỉ hưu

HÌNH 1.9
Sự khác biệt giữa cổ điển mô hình và đối tượng-định hướng mô hình.

Mô hình cổ điển	Mô hình hướng đối tượng
2. Giai đoạn phân tích (đặc tả) • Xác định sản phẩm dùng để làm gì	2. Quy trình phân tích hướng đối tượng • Xác định sản phẩm dùng để làm gì • Trích xuất các lớp
3. Giai đoạn thiết kế • Thiết kế kiến trúc (trích xuất các mô-đun) • Thiết kế chi tiết	3. Quy trình thiết kế hướng đối tượng • Thiết kế chi tiết
4. Giai đoạn thực hiện • Mã hóa các mô-đun bằng một ngôn ngữ lập trình thích hợp • Tích hợp	4 . Quy trình thực hiện hướng đối tượng • Mã các lớp bằng một ngôn ngữ lập trình hướng đối tượng thích hợp • Tích hợp

Do đó, phân tích hướng đối tượng đi xa hơn giai đoạn phân tích (đặc tả) tương ứng của mô hình cổ điển. Điều này được thể hiện trong Hình 1.9.

Sự khác biệt giữa hai mô hình này có hậu quả lớn. Khi mô hình cổ điển được sử dụng, hầu như luôn có sự chuyển đổi rõ ràng giữa giai đoạn phân tích và giai đoạn thiết kế. Sau cùng, mục đích của giai đoạn phân tích là xác định*gì*sản phẩm là để làm, trong khi mục đích của giai đoạn thiết kế là quyết định*Làm sao*để làm điều đó. Ngược lại, khi phân tích hướng đối tượng được sử dụng, các đối tượng đi vào vòng đời ngay từ đầu. Các đối tượng được trích xuất trong quy trình phân tích, được thiết kế trong quy trình thiết kế và được mã hóa trong quy trình thực hiện. Do đó, mô hình hướng đối tượng là một cách tiếp cận tích hợp; quá trình chuyển đổi từ quy trình làm việc sang quy trình làm việc trơn tru hơn nhiều so với mô hình cổ điển, giảm số lượng lỗi phát sinh trong quá trình phát triển.

Như đã đề cập, việc định nghĩa một đối tượng chỉ đơn thuần là một tạo tác phần mềm mà nó đóng gói cả các thuộc tính và hoạt động và thực hiện nguyên tắc ẩn thông tin là không đủ. Một định nghĩa đầy đủ hơn được đưa ra trong Chương 7, nơi các đối tượng được xem xét chuyên sâu.

1.10 Mô hình hướng đối tượng trong phối cảnh

Hình 1.1 là bằng chứng về nhiều thiếu sót của mô hình cổ điển (có cấu trúc). Tuy nhiên, mô hình hướng đối tượng hoàn toàn không phải là thuốc chữa bách bệnh cho tất cả các bệnh:

- Giống như tất cả các phương pháp tiếp cận sản xuất phần mềm, mô hình hướng đối tượng phải được sử dụng đúng cách; nó cũng dễ dàng sử dụng sai mô hình hướng đối tượng như bất kỳ mô hình nào khác.
- Khi được áp dụng một cách chính xác, mô hình hướng đối tượng có thể giải quyết một số (nhưng không phải tất cả) các vấn đề của mô hình cổ điển.
- Mô hình hướng đối tượng có một số vấn đề của riêng nó, như được mô tả trong Phần 7.9.
- Mô hình hướng đối tượng là cách tiếp cận tốt nhất hiện nay. Tuy nhiên, giống như tất cả các công nghệ, nó chắc chắn sẽ được thay thế bởi một công nghệ vượt trội trong tương lai.

Trong cuốn sách này, điểm mạnh và điểm yếu của cả mô hình cổ điển và mô hình hướng đối tượng đều được chỉ ra trong bối cảnh của chủ đề cụ thể đang thảo luận. Do đó, sự so sánh của hai mô hình không xuất hiện ở một nơi duy nhất mà được trải rộng trên toàn bộ cuốn sách.

Bây giờ chúng tôi xác định một số thuật ngữ kỹ thuật phần mềm.

1.11 Thuật ngữ

Các **khách hàng** là cá nhân muốn một sản phẩm được xây dựng (phát triển). Các **các nhà phát triển** là các thành viên của nhóm chịu trách nhiệm xây dựng sản phẩm đó. Các nhà phát triển có thể chịu trách nhiệm về mọi khía cạnh của quy trình phần mềm, từ các yêu cầu trở đi, hoặc họ có thể chỉ chịu trách nhiệm về việc triển khai một sản phẩm đã được thiết kế sẵn.

Cả khách hàng và nhà phát triển có thể là một phần của cùng một tổ chức. Ví dụ: khách hàng có thể là chuyên gia tính toán chính của một công ty bảo hiểm và các nhà phát triển một nhóm do phó chủ tịch phụ trách phát triển phần mềm của công ty bảo hiểm đó đứng đầu. Điều này được gọi là **phát triển phần mềm nội bộ**. Other face, with **đồng phần mềm** khách hàng và các nhà phát triển là thành viên của các tổ chức hoàn toàn độc lập. Ví dụ: khách hàng có thể là một chức năng cấp cao trong Bộ Quốc phòng và các nhân viên phát triển của một nhà thầu quốc phòng lớn chuyên về phần mềm cho vũ khí hệ thống. Ở quy mô nhỏ hơn nhiều, khách hàng có thể là kế toán viên thực thi một người và nhà phát triển là sinh viên kiếm thu nhập bằng cách phát triển phần mềm bán thời gian.

Liên quan thứ hai đến phần mềm sản xuất là **người dùng**. Người dùng là người hoặc những người thay mặt khách hàng đã ủy quyền sản phẩm và người sẽ sử dụng phần mềm. Trong ví dụ về công ty bảo hiểm, người dùng có thể là đại lý bảo hiểm, họ sẽ sử dụng phần mềm để chọn các chính sách phù hợp nhất. Trong một số trường hợp, khách hàng và người dùng là cùng một người (ví dụ: kế toán đã thảo luận trước đó).

Trái ngược với phần mềm tùy chỉnh đắt tiền được phát triển cho một khách hàng, nhiều bản sao của phần mềm, chẳng hạn như bộ xử lý văn bản hoặc bảng tính, được bán với giá thấp hơn nhiều cho một số lượng lớn người mua. Có nghĩa là, các nhà sản xuất phần mềm đó (chẳng hạn như Microsoft hoặc Borland) thu hồi chi phí phát triển sản phẩm theo số lượng bán ra. Loại phần mềm này thường được gọi là **phần mềm thương mại bán sẵn (COTS)**. Thuật ngữ trước đây cho loại phần mềm này là **phần mềm bao bọc** bởi vì hộp chứa CD hoặc đĩa, sách hướng dẫn và thỏa thuận cấp phép hầu như luôn được bọc lại. Ngày nay, phần mềm COTS thường được tải xuống qua World Wide Web — không có hộp nào để thu gọn lại. Vì lý do này, phần mềm COTS ngày nay đôi khi được gọi là **phần mềm bấm**. Phần mềm COTS được phát triển cho "thị trường"; nghĩa là, phần mềm không được nhắm mục tiêu đến một khách hàng hoặc người dùng cụ thể cho đến khi nó được phát triển và có sẵn để mua.

Open source code software đang trở nên cực kỳ phổ biến. Một sản phẩm mở nguồn mã phần mềm được phát triển và duy trì bởi một nhóm tình nguyện viên và bất kỳ ai cũng có thể tải xuống và sử dụng miễn phí. Open source code the product are used to width bao gồm các hệ điều hành Linux, Web Firefox duyệt web và Web Apache máy chủ. Timeout *open source code* đề cập đến tính năng sử dụng của nguồn mã đối với tất cả, không giống như hầu hết các sản phẩm thương mại chỉ bán phiên bản thực thi. Bởi vì bất kỳ người dùng nào mở nguồn mã sản phẩm cũng có thể xem kỹ lưỡng nguồn mã và thông báo lỗi cho nhà phát triển, nên nhiều sản phẩm mở nguồn mã phần mềm có chất lượng cao. Hậu quả mong đợi của công khai chất lượng bản in của các lỗi trong open source phần mềm đã được Raymond chính thức hóa trong *Nhà thờ và Chợ* as *Luật Linus*, được đặt theo tên của Linus Torvalds, người tạo ra Linux [Raymond, 2000]. **Luật Linus** say that "well-level up enough label, all the well-being." Nói cách khác, nếu có đủ cá nhân xem kỹ lưỡng nguồn mã của một sản phẩm mở nguồn phần mềm, thì ai đó sẽ có thể xác định được lỗi đó và đề xuất cách khắc phục nó (nhưng hãy xem chỉ trong trường hợp bạn muốn biết ở Hộp 1.6). A liên quan nguyên tắc là "Phát hành sớm. Phát hành thường xuyên "[Raymond, 2000].

Chỉ trong trường hợp bạn muốn biết Box 1.6

Rõ ràng là càng có nhiều người kiểm tra cẩn thận một đoạn mã, thì càng có nhiều người có khả năng sẽ tìm ra và sửa được lỗi trong đoạn mã đó. Theo đó, Định luật Linus có lẽ nên được gọi là "Chủ nghĩa thực sự của Torvalds".

Có nghĩa là, các nhà phát triển mở nguồn mã có xu hướng dành ít thời gian hơn cho việc kiểm tra so với các nhà phát triển đóng mã nguồn, họ thích phát hành phiên bản mới của sản phẩm ngay sau khi nó hoàn thành, to re many an results for users.

Một từ được sử dụng trên hầu hết các trang của cuốn sách là**phần mềm**. Phần mềm không chỉ bao gồm mã ở dạng máy tính có thể đọc được mà còn bao gồm tất cả các tài liệu là nội dung phần của mọi dự án. Phần mềm bao gồm đặc tả tài liệu, thiết kế tài liệu, các loại tài liệu pháp lý và kế toán, kế hoạch quản lý dự án phần mềm và các phần mềm quản lý tài liệu khác cũng như tất cả các hướng dẫn sổ tay loại .

Kể từ những năm 1970, sự khác biệt giữa**chương trình**and a**system**đã trở nên mờ. Trong "những ngày xưa tốt đẹp", sự phân biệt rất rõ ràng. Một chương trình là một đoạn mã tự trị, thường ở dạng một bộ bài đục lỗ có thể thực hiện được. Một hệ thống là một tập hợp các chương trình có liên quan. Một hệ thống có thể bao gồm các chương trìnhP, Q, R,vàS. Băng từT_1đã được gắn kết, và sau đó lập trìnhPđã chạy. Nó khiến một bộ bài dữ liệu được đọc vào và được tạo ra dưới dạng băng đầu raT_2vàT_3. BăngT_2sau đó được quấn lại và lập trìnhQđã được điều hành, sản xuất băngT_4dưới dạng đầu ra. Chương trìnhRbây giờ đã hợp nhất băngT_3vàT_4thành băngT_5;T_5phục vụ như đầu vào cho chương trình S,trong đó đã in một loạt báo cáo.

So sánh tình huống đó với một sản phẩm, đang chạy trên máy có bộ xử lý truyền thông mặt trước và trình quản lý cơ sở dữ liệu phía sau, thực hiện điều khiển nhà máy thép theo thời gian thực. Một phần mềm duy nhất điều khiển nhà máy thép làm được nhiều việc hơn so với hệ thống kiểu cũ, nhưng xét về các định nghĩa cổ điển về chương trình và hệ thống, phần mềm này chắc chắn là một chương trình. Để thêm vào sự nhầm lẫn, thuật ngữ*hệ thống*bây giờ cũng được sử dụng để biểu thị sự kết hợp phần cứng-phần mềm. Ví dụ, hệ thống điều khiển chuyến bay trên máy bay bao gồm cả máy tính trên máy bay và phần mềm chạy trên chúng. Tùy thuộc vào người sử dụng thuật ngữ, hệ thống điều khiển chuyến bay cũng có thể bao gồm các bộ điều khiển, chẳng hạn như cần điều khiển, gửi lệnh đến máy tính và các bộ phận của máy bay, chẳng hạn như cánh lật, được điều khiển bởi máy tính. Hơn nữa, trong bối cảnh phát triển phần mềm truyền thống, thuật ngữ**phân tích hệ thống**đề cập đến hai giai đoạn đầu tiên (yêu cầu và giai đoạn phân tích) và**thiết kế hệ thống**đề cập đến giai đoạn thứ ba (giai đoạn thiết kế).

Để giảm thiểu sự nhầm lẫn, cuốn sách này sử dụng thuật ngữ**sản phẩm**để biểu thị một phần mềm không tầm thường. Có hai lý do cho quy ước này. Đầu tiên chỉ đơn giản là để loại bỏ sự nhầm lẫn của chương trình và hệ thống bằng cách sử dụng thuật ngữ thứ ba. Lý do thứ hai quan trọng hơn. Cuốn sách này đề cập đến quy trình sản xuất phần mềm, tức là cách chúng tôi sản xuất phần mềm và kết quả cuối cùng của một quy trình được gọi là*sản phẩm*. Cuối cùng, thuật ngữ*hệ thống*được sử dụng theo nghĩa hiện đại, nghĩa là phần cứng và phần mềm kết hợp, hoặc như một phần của các cụm từ được chấp nhận rộng rãi, chẳng hạn như hệ điều hành và hệ thống thông tin quản lý.

Hai từ được sử dụng rộng rãi trong bối cảnh kỹ thuật phần mềm là*phương pháp luận*và*mô hình*. Vào những năm 1970, từ**phương pháp luận**bắt đầu được sử dụng với nghĩa là "một cách phát triển một sản phẩm phần mềm"; từ thực sự có nghĩa là "khoa học của các phương pháp." Sau đó, vào những năm 1980, từ**mô hình**đã trở thành một từ thông dụng chính của thế giới kinh doanh, như trong cụm từ, "Đó là một mô hình hoàn toàn mới." Công nghiệp phần mềm sớm

Chỉ trong trường hợp bạn muốn biết — Hộp 1.7

Cách sử dụng đầu tiên của từ này *sâu bọ* để biểu thị một lỗi là do cố Chuẩn Đô đốc Grace Murray Hopper, một trong những người thiết kế COBOL. Vào ngày 9 tháng 9 năm 1945, một con bướm đêm đã bay vào máy tính Mark II mà Hopper và các đồng nghiệp của cô sử dụng tại Harvard và nằm giữa các tấm tiếp xúc của một rơ le. Theo đó, thực sự đã có một lỗi trong hệ thống. Hopper đã ghi lỗi vào nhật ký và viết, "Đã tìm thấy trường hợp lỗi thực tế đầu tiên." Nhật ký, với con bướm đêm vẫn còn được đính kèm, nằm trong Bảo tàng Hải quân tại Trung tâm Vũ khí Bề mặt Hải quân, ở Dahlgren, Virginia.

Mặc dù đây có thể là lần đầu tiên sử dụng *sâu bọ* trong bối cảnh máy tính, từ này đã được sử dụng trong tiếng lóng kỹ thuật vào thế kỷ 19 [Shapiro, 1994]. Ví dụ, Thomas Alva Edison đã viết vào ngày 18 tháng 11 năm 1878, “Điều này tạo ra và sau đó điều đó — 'Bâu' — chẳng hạn như những sai lầm và khó khăn nhỏ như vậy được gọi là. . . ” [Josephson, 1992]. Một trong những định nghĩa của *sâu bọ* trong ấn bản năm 1934 của *Từ điển tiếng Anh mới của Webster* là, "Một khiếm khuyết trong bộ máy hoặc hoạt động của nó." Rõ ràng từ nhận xét của Hopper rằng cô ấy cũng đã quen với việc sử dụng từ này trong bối cảnh đó; nếu không, cô ấy sẽ giải thích những gì cô ấy muốn nói.

---

bắt đầu sử dụng từ này *mô hình* trong các cụm từ **mô hình hướng đối tượng** và cổ điển (hoặc **truyên thông**) **mô hình** nghĩa là "một phong cách phát triển phần mềm." Đây là một sự lựa chọn thuật ngữ không may mắn khác, bởi vì một mô thức là một mô hình hoặc một khuôn mẫu. Những độc giả hiểu lầm bị xúc phạm bởi sự hư hỏng này của ngôn ngữ tiếng Anh được nhiệt liệt mời thay mặt tác giả nhận những lời khen ngợi về độ chính xác ngôn ngữ; anh ấy mệt mỏi với việc nghiêng mình trước những chiếc cối xay gió.

Phương pháp luận hoặc mô hình là một thành phần của toàn bộ quy trình phần mềm. Ngược lại, một **kĩ thuật** là một thành phần của một phần của quy trình phần mềm. Ví dụ bao gồm kỹ thuật mã hóa, kỹ thuật tài liệu và kỹ thuật lập kế hoạch.

Khi một lập trình viên tạo ra một **sai lầm**, hậu quả của sai lầm đó là một **lỗi** trong mã. Việc thực thi sản phẩm phần mềm sau đó dẫn đến **thất bại**, nghĩa là, hành vi không chính xác được quan sát thấy của sản phẩm do lỗi. Một **lỗi** là số tiền mà một kết quả không chính xác. Các điều khoản *sai lầm*, *lỗi*, *thất bại*, và *lỗi* được định nghĩa trong Tiêu chuẩn IEEE 610.12, “Bảng chú giải thuật ngữ kỹ thuật phần mềm” [IEEE 610.12, 1990], được khẳng định lại vào năm 2002 [Tiêu chuẩn IEEE, 2003]. Từ **khuyết điểm** là một thuật ngữ chung đề cập đến lỗi, hỏng hóc hoặc lỗi. Vì lợi ích của sự chính xác, trong cuốn sách này, chúng tôi hạn chế tối đa việc sử dụng thuật ngữ ô *khuyết điểm*.

Một thuật ngữ cần tránh càng xa càng tốt là **sâu bọ** (lịch sử của từ này nằm trong Trường hợp bạn muốn biết Hộp 1.7). Thời hạn *sâu bọ* ngày nay chỉ đơn giản là một cách viết tắt cho một *lỗi*. Mặc dù nói chung không có tác hại thực sự nào trong việc sử dụng các từ ngữ, từ *sâu bọ* có âm bội không có lợi cho việc sản xuất phần mềm tốt. Cụ thể, thay vì nói, "Tôi đã mắc lỗi", một lập trình viên sẽ nói, "Một lỗi len lỏi vào mã" (không phải *của tôi* mã nhưng *các* mã), từ đó chuyển trách nhiệm về lỗi từ lập trình viên sang lỗi. Không ai đổ lỗi cho một lập trình viên khi gặp một ca bệnh cúm, vì bệnh cúm là do con bọ cúm gây ra. Đề cập đến một sai lầm như một lỗi là một cách để chối bỏ trách nhiệm. Ngược lại, lập trình viên nói, “Tôi đã mắc sai lầm”, là một chuyên gia máy tính chịu trách nhiệm về hành động của mình.

Sự nhầm lẫn đáng kể xung quanh thuật ngữ hướng đối tượng. Ví dụ, ngoài thuật ngữ **thuộc tính** cho một thành phần dữ liệu của một đối tượng, thuật ngữ **biến số đưa ra** đôi khi được sử dụng trong văn học hướng đối tượng. Trong Java, thuật ngữ này là **biến cá thể**. Trong C ++, thuật ngữ **fitrường học** được sử dụng và trong Visual Basic .NET, thuật ngữ này là **bất động sản**. Đối với việc thực hiện các hoạt động của một đối tượng, thuật ngữ *phương pháp* thường được sử dụng; trong

C ++, tuy nhiên, thuật ngữ này là**chức năng thành viên**. Trong C ++, một*thành viên*của một đối tượng tham chiếu đến một thuộc tính (“trường”) hoặc một phương thức. Trong Java, thuật ngữ*fitrường học*được sử dụng để biểu thị một thuộc tính (“biến phiên bản”) hoặc một phương thức. Để tránh nhầm lẫn, nếu có thể, các thuật ngữ chung*thuộc tính*và *phương pháp*được sử dụng trong cuốn sách này.

May mắn thay, một số thuật ngữ được chấp nhận rộng rãi. Ví dụ: khi một phương thức trong một đối tượng được gọi, điều này hầu như được gọi là**gửi một tin nhắn**đối tượng.

1.12 Các vấn đề đạo đức

Chúng tôi kết thúc chương này trên một lưu ý cảnh báo. Sản phẩm phần mềm được phát triển và duy trì bởi con người. Nếu những cá nhân đó làm việc chăm chỉ, thông minh, nhạy bén, cập nhật và trên hết,*có đạo đức*, thì rất có thể cách mà các sản phẩm phần mềm mà họ phát triển và bảo trì sẽ đạt yêu cầu. Thật không may, điều ngược lại cũng đúng như nhau.

Hầu hết các hiệp hội dành cho các chuyên gia có mã là**đạo đức học**mà tất cả các thành viên của nó phải tuân thủ. Hai hiệp hội lớn dành cho các chuyên gia máy tính, Hiệp hội Máy tính Máy tính (ACM) và Hiệp hội Máy tính của Viện Kỹ sư Điện và Điện tử (IEEE-CS) đã cùng nhau thông qua Bộ Quy tắc Đạo đức và Thực hành Nghề nghiệp Kỹ thuật Phần mềm làm tiêu chuẩn cho việc giảng dạy và thực hành kỹ thuật phần mềm [IEEE / ACM, 1999]. Nó dài dòng, vì vậy một phiên bản ngắn, bao gồm một phần mở đầu và tám nguyên tắc, cũng đã được sản xuất. Đây là phiên bản ngắn:

Quy tắc đạo đức và thực hành nghề nghiệp của ngành kỹ thuật phần mềm[2](Phiên bản 5.2)

theo khuyến nghị của Lực lượng đặc nhiệm hỗn hợp IEEE-CS / ACM về
Đạo đức Kỹ thuật phần mềm và Thực hành nghề nghiệp

Phiên bản ngắn

Mở đầu

Phiên bản ngắn của mã tóm tắt các nguyện vọng ở mức độ trừu tượng cao; các điều khoản được bao gồm trong phiên bản đầy đủ đưa ra các ví dụ và chi tiết về cách những nguyện vọng này thay đổi cách chúng ta hoạt động với tư cách là các chuyên gia kỹ thuật phần mềm. Nếu không có nguyện vọng, các chi tiết có thể trở nên hợp pháp và tẻ nhạt; không có chi tiết, khát vọng có thể trở nên thanh cao nhưng trống rỗng; cùng với nhau, các nguyện vọng và các chi tiết tạo thành một mã gắn kết.

Các kỹ sư phần mềm phải cam kết làm cho việc phân tích, đặc tả, thiết kế, phát triển, kiểm tra và bảo trì phần mềm trở thành một nghề có lợi và được tôn trọng. Theo cam kết của họ đối với sức khỏe, an toàn và phúc lợi của công chúng, các kỹ sư phần mềm phải tuân thủ Tám nguyên tắc sau:

1. *Công cộng*—Kỹ sư phần mềm phải hành động nhất quán vì lợi ích công cộng.
2. *Khách hàng và Nhà tuyển dụng*—Các kỹ sư phần mềm phải hành động theo cách có lợi nhất cho khách hàng và người sử dụng lao động của họ phù hợp với lợi ích công cộng.

[2]© 1999 bởi Viện Kỹ sư Điện và Điện tử, Inc. và Hiệp hội Máy tính Máy tính, Inc.

3. *Sản phẩm*—Các kỹ sư phần mềm phải đảm bảo rằng các sản phẩm của họ và các sửa đổi liên quan đáp ứng các tiêu chuẩn chuyên môn cao nhất có thể.
4. *Sự phán xét*—Kỹ sư phần mềm phải duy trì tính toàn vẹn và độc lập trong đánh giá chuyên môn của họ.
5. *Ban quản lý*— Các nhà lãnh đạo và quản lý kỹ thuật phần mềm phải đăng ký và thúc đẩy cách tiếp cận có đạo đức để quản lý phát triển và bảo trì phần mềm.
6. *Chuyên nghiệp*—Kỹ sư phần mềm phải nâng cao tính chính trực và danh tiếng của nghề nghiệp phù hợp với lợi ích công cộng.
7. *Đồng nghiệp*—Kỹ sư phần mềm phải công bằng và hỗ trợ các đồng nghiệp của họ.
số 8. *Bản thân*-Các kỹ sư phần mềm phải tham gia vào quá trình học tập suốt đời về việc thực hành nghề nghiệp của họ và sẽ thúc đẩy cách tiếp cận đạo đức đối với việc thực hành nghề nghiệp.

Các quy tắc đạo đức của các xã hội khác dành cho các chuyên gia máy tính cũng thể hiện tình cảm tương tự. Điều quan trọng đối với tương lai nghề nghiệp của chúng tôi là chúng tôi tuân thủ nghiêm ngặt các quy tắc đạo đức như vậy.

Trong Chương 2, chúng tôi xem xét các mô hình vòng đời khác nhau để làm sáng tỏ thêm về sự khác biệt giữa mô hình cổ điển và mô hình hướng đối tượng.

Chương Kiểm tra lại

Kỹ thuật phần mềm được định nghĩa (Phần 1.1) là một ngành có mục đích là sản xuất phần mềm không có lỗi, đáp ứng nhu cầu của người dùng và được phân phối đúng hạn và trong phạm vi ngân sách. Để đạt được mục tiêu này, các kỹ thuật thích hợp phải được sử dụng trong suốt quá trình sản xuất phần mềm, bao gồm cả khi thực hiện phân tích (đặc tả) và thiết kế (Phần 1.4) và bảo trì sau giao hàng (Phần 1.3). Kỹ thuật phần mềm giải quyết tất cả các bước của vòng đời phần mềm và kết hợp các khía cạnh của nhiều lĩnh vực kiến thức khác nhau của con người, bao gồm kinh tế học (Phần 1.2) và khoa học xã hội (Phần 1.5). Không có giai đoạn lập kế hoạch riêng (Phần 1.6), không có giai đoạn thử nghiệm (Phần 1.7), và không có giai đoạn lập tài liệu (Phần 1.8). Trong Phần 1.9, các đối tượng được giới thiệu và so sánh giữa mô hình cổ điển và hướng đối tượng được thực hiện. Sau đó, mô hình hướng đối tượng được đánh giá (Phần 1.10). Tiếp theo, trong Phần 1.11, thuật ngữ được sử dụng trong cuốn sách này sẽ được giải thích. Cuối cùng, các vấn đề đạo đức được thảo luận trong Phần 1.12.

Vì Hơn nữa Đọc

Nguồn thông tin sớm nhất về phạm vi kỹ thuật phần mềm là [Boehm, 1976]. Tương lai của kỹ thuật phần mềm được thảo luận trong [Finkelstein, 2000]. Tình trạng hiện tại của hoạt động kỹ thuật phần mềm được mô tả trong nhiều bài báo trong số tháng 11 đến tháng 12 năm 2003 của *Phần mềm IEEE.* Một cuộc điều tra về các yếu tố dẫn đến phát triển phần mềm thành công xuất hiện trong [Procaccino, Verner và Lorenzet, 2006].

Để biết tầm quan trọng của việc bảo trì sau giao hàng trong kỹ thuật phần mềm và cách lập kế hoạch cho nó, hãy xem [Parnas, 1994]. Phát triển phần mềm cho các sản phẩm dựa trên COTS là chủ đề của [Brownsword, Oberndorf và Sledge, 2000]. Việc thu nhận các thành phần COTS được mô tả trong [Ulkuniemi và Seppanen, 2004] và trong [Keil và Tiwana, 2005]. Quản lý rủi ro khi phần mềm được phát triển bằng cách sử dụng các thành phần COTS được mô tả trong [Li và cộng sự, 2008]. Số tháng 7 đến tháng 8 năm 2005 của *Phần mềm IEEE* có sáu bài báo về tích hợp các thành phần COTS vào các sản phẩm phần mềm, bao gồm [Donzelli và cộng sự, 2005] và [Yang, Bhuta, Boehm, và Port, 2005]. Đánh giá lại việc quản lý rủi ro xuất hiện trong [Bannerman, 2008].

Rủi ro trong hệ thống doanh nghiệp được mô tả trong [Scott và Vessey, 2002] và trong hệ thống thông tin nói chung trong [Longstaff, Chittister, Pethia, và Haimes, 2000]. Zvegintzov [1998] giải thích về việc ít dữ liệu chính xác về thực hành kỹ thuật phần mềm thực sự có sẵn như thế nào.

Thực tế là toán học làm nền tảng cho kỹ thuật phần mềm được nhấn mạnh trong [Devlin, 2001]. Tầm quan trọng của kinh tế học trong kỹ thuật phần mềm được thảo luận trong [Boehm và Huang, 2003]. Số tháng 11 đến tháng 12 năm 2002 của *Phần mềm IEEE* chứa một số bài báo về kinh tế kỹ thuật phần mềm.

Hai cuốn sách kinh điển về khoa học xã hội và kỹ thuật phần mềm là [Weinberg, 1971] và [Shneiderman, 1980]. Cả hai cuốn sách đều không yêu cầu kiến thức trước về tâm lý học hoặc khoa học hành vi nói chung.

Tác phẩm vượt thời gian của Brooks [1975], *Người đàn ông thần thoại-Tháng*, là phần giới thiệu rất được khuyến khích về thực tế của kỹ thuật phần mềm. Cuốn sách bao gồm tài liệu về tất cả các chủ đề được đề cập trong chương này.

Phần giới thiệu tuyệt vời về phần mềm nguồn mở là [Raymond, 2000]. Paulsen, Succi, và Eberlein [2004] trình bày một nghiên cứu thực nghiệm so sánh các sản phẩm phần mềm mã nguồn mở và mã nguồn đóng. Tái sử dụng các thành phần mã nguồn mở được mô tả trong [Madanmohan và De ', 2004]. Một loạt các bài báo về phần mềm nguồn mở xuất hiện trong số tháng 1 / tháng 2 năm 2004 của *Phần mềm IEEE* và trong số 2 năm 2005, của *Tạp chí Hệ thống của IBM*. Vấn đề liệu phần mềm nguồn mở có dẫn đến tăng cường bảo mật hay không được thảo luận trong [Hoepman và Jacobs, 2007]. Tác động qua lại giữa doanh nghiệp và phần mềm nguồn mở là chủ đề của [Watson và cộng sự, 2008], [Ven, Verelst, và Mannaert, 2008], và [Wesselius, 2008].

Một phần giới thiệu tuyệt vời về mô hình hướng đối tượng là [Budd, 2002]. Ba dự án thành công được thực hiện bằng cách sử dụng mô hình hướng đối tượng được mô tả trong [Capper, Colgate, Hunter và James, 1994], với một phân tích chi tiết. Một cuộc khảo sát về thái độ của 150 nhà phát triển phần mềm có kinh nghiệm đối với mô hình hướng đối tượng được báo cáo trong [Johnson, 2000]. Về đạo đức, một quy tắc đạo đức chung cho cả các chuyên gia kinh doanh và phần mềm được trình bày trong [Payne và Landry, 2006].

Điều khoản quan trọng

Các vấn đề

1.1 Bạn chịu trách nhiệm tự động hóa một thực hành kiến trúc nhiều địa điểm. Chi phí phát triển phần mềm đã được ước tính là $ 530,000. Cần thêm khoảng bao nhiêu tiền để bảo trì phần mềm sau giao hàng?

1.2 Có cách nào để dung hòa định nghĩa thời gian cổ điển về bảo trì với định nghĩa vận hành mà chúng ta đang sử dụng hiện nay không? Giải thich câu trả lơi của bạn.

1.3 Bạn là nhà tư vấn kỹ thuật phần mềm. Giám đốc thông tin của một công ty phân phối xăng dầu khu vực muốn bạn phát triển một sản phẩm phần mềm sẽ thực hiện tất cả các chức năng kế toán của công ty và cung cấp thông tin trực tuyến cho nhân viên trụ sở chính về các đơn đặt hàng và tồn kho trong các bể chứa khác nhau của công ty. Máy tính là bắt buộc đối với 21 nhân viên kế toán, 15 nhân viên đặt hàng và 37 nhân viên kho lưu trữ. Ngoài ra, 14 nhà quản lý cần quyền truy cập vào dữ liệu. Công ty sẵn sàng trả 30.000 USD cho phần cứng và phần mềm cùng nhau và muốn có sản phẩm phần mềm hoàn chỉnh trong 4 tuần. Bạn nói gì với anh ta? Hãy nhớ rằng công ty của bạn muốn công việc kinh doanh của tập đoàn của anh ấy, bất kể yêu cầu của anh ấy vô lý đến mức nào.

1.4 Bạn là phó đô đốc trong Hải quân Velorian. Nó đã được quyết định kêu gọi một tổ chức phát triển phần mềm để phát triển phần mềm điều khiển cho một thế hệ tên lửa đối hạm mới. Bạn phụ trách giám sát dự án. Để bảo vệ chính phủ Veloria, bạn đưa ra những điều khoản nào trong hợp đồng với các nhà phát triển phần mềm?

1.5 Bạn là một kỹ sư phần mềm có công việc là giám sát sự phát triển của phần mềm trong Bài toán 1.4. Liệt kê những cách mà công ty của bạn có thể không đạt được hợp đồng với hải quân. Những nguyên nhân có thể xảy ra của những thất bại như vậy là gì?

1.6 Chín tháng sau khi giao hàng, lỗi được phát hiện trong phần mềm của sản phẩm phân tích mRNA bằng thuốc thử Stein-Röntgen. Chi phí sửa lỗi là $ 18,900. Nguyên nhân của lỗi là một câu không rõ ràng trong tài liệu đặc tả. Chi phí khoảng bao nhiêu để sửa lỗi trong giai đoạn phân tích?

1.7 Giả sử rằng lỗi trong Vấn đề 1.6 đã được phát hiện trong giai đoạn thực hiện. Sau đó nó sẽ có chi phí sửa chữa khoảng bao nhiêu?

1.8 Bạn là chủ tịch của một tổ chức xây dựng phần mềm quy mô lớn. Bạn hiển thị Hình 1.6 cho nhân viên của mình, thúc giục họ sớm tìm ra lỗi trong vòng đời phần mềm. Có người phản hồi rằng thật phi lý khi mong đợi bất kỳ ai xóa lỗi trước khi họ nhập sản phẩm. Ví dụ, làm thế nào mọi người có thể loại bỏ một lỗi trong khi thiết kế đang được sản xuất nếu lỗi được đề cập là lỗi mã hóa? Bạn trả lời gì?

1.9 Mô tả một tình huống trong đó khách hàng, nhà phát triển và người dùng là cùng một người.

1.10 Những vấn đề nào có thể phát sinh nếu khách hàng, nhà phát triển và người dùng là cùng một người? Làm thế nào những vấn đề này có thể được giải quyết?