Lời cảm ơn

Chúng em xin chân thành cảm ơn Khoa Công nghệ Thông tin, trường Đại học Khoa học Tự nhiên, TpHCM đã tạo điều kiện cho chúng em thực hiện đề tài tốt nghiệp này.

Chúng em xin gửi lời cảm ơn chân thành đến Thầy Lâm Quang Vũ đã tận tình hướng dẫn, chỉ bảo chúng em trong suốt thời gian thực hiện đề tài. Trong quá trình làm luận văn với Thầy, chúng em đã học hỏi được nhiều kiến thức bổ ích và kinh nghiệm quý báu làm nền tảng cho quá trình học tập, làm việc và nghiên cứu sau này.

Chúng em cũng xin chân thành cảm ơn quý Thầy Cô trong Khoa CNTT đã tận tình giảng dạy, trang bị cho chúng em những kiến thức cần thiết trong suốt quá trình học tập và nghiên cứu tại Khoa, và cũng xin gửi lòng biết ơn sâu sắc đến của chúng em đến các anh chị phòng Selab, đặc biệt là các anh Trương Thiên Đỉnh, anh Đỗ Lệnh Hùng Sơn, anh Phạm Minh Tuấn, chị Nguyễn Thị Thu Thủy, những người đã dìu dắt, định hướng và giúp đỡ chúng em rất nhiều trong quá trình thực hiện đề tài này.

Chúng con luôn ghi nhớ công ơn sinh thành, dưỡng dục của Ba, Mẹ. Ba mẹ đã luôn giúp đỡ, động viên chúng con trong học tập cũng như trong cuộc sống.

Trong quá trình thực hiện đề tài, chúng tôi cũng nhận được sự giúp đỡ và động viên hết sức chân tình của các bạn cùng lớp, xin hãy ghi nhận ở chúng tôi lòng biết ơn sâu sắc.

Mặc dù đã cố gắng hoàn thành luận văn với tất cả sự nỗ lực của bản thân, nhưng luận văn chắc chắn không tránh khỏi những thiếu sót, kính mong quý Thầy Cô tận tình chỉ bảo.

Một lần nữa, chúng em xin chân thành cảm ơn đến tất cả mọi người.

Tp. Hồ Chí Minh 07/2005

Sinh viên thực hiện

Phạm Thị Ngọc Vân – Trương Thị Ngọc Phượng

Tóm tắt luận văn

Ngày nay, công nghệ phần mềm đóng vai trò quan trọng trong hầu hết các lĩnh vực. Hầu như những hệ thống tiên tiến quan trọng đều có sự tham gia, hỗ trợ của các hệ thống phần mềm. Chính vì vậy mà chi phí, lịch biểu và chất lượng phần mềm cũng là các yếu tố mà cả những tổ chức sử dụng phần mềm và những tổ chức phát triển phần mềm đều rất quan tâm.

Xuất phát từ những nhu cầu trên, nhiều nỗ lực cải tiến quy trình đã được thực hiện. CMM (Capability Maturity Model) ra đời, đã thể hiện là một mô hình cải tiến độ trưởng thành phần mềm hữu dụng. Tuy nhiên, mô hình này chỉ áp dụng cho tổ chức và nó vẫn còn nhiều hạn chế về mặt phương pháp thực hiện. Hơn nữa, khi các tổ chức tiếp cận và chuyển lên cấp độ 3 của CMM (CMM có 5 cấp độ), họ nhận thấy rằng sự hoàn chỉnh hơn nữa phụ thuộc vào sự phát triển của quy trình phần mềm cá nhân. Chính vì thế, từ năm 1989, PSP đã được phát triển bởi Watts S. Humphrey để đáp ứng việc phát triển liên quan đến việc làm thế nào để đưa một tổ chức vượt xa hơn cấp độ 2 của CMM. Cuối năm 1994, CMU và SEI (Carnegie Mellon University and Software Engineering Institute) đã công bố qui trình phần mềm cá nhân (Personal Software Process – PSP) như là một mô hình hỗ trợ việc phát triển qui trình cho từng kỹ sư phát triển phần mềm.

Qui trình phần mềm cá nhân tập trung vào việc cải thiện hiệu quả làm việc và chất lượng công việc của kỹ sư. Hai khía cạnh chính mà PSP tập trung hỗ trợ là:

- Quản lý thời gian và kế hoạch – quy trình lên kế hoạch
- Quản lý chất lượng sản phẩm – quy trình quản lý sai sót

Về cả 2 mặt lý thuyết và thực tế, qui trình phần mềm cá nhân cải thiện rất nhiều trong chất lượng làm việc của kỹ sư. Tuy nhiên, việc thực hiện rộng rãi PSP ở phạm vi cá nhân và trong môi trường công nghiệp còn khó khăn vì mức độ nghiêm ngặt của nó. Nhưng dù sao đi nữa, PSP cũng hứa hẹn sẽ được sử dụng rộng rãi vì tính hiệu quả của nó không những cho các cá nhân làm phần mềm mà còn cho tất cả mọi người.

MỤC LỤC

Danh mục các ký hiệu, các chữ viết tắt

A/FR	Appraisal to Failure Ratio
CMM	Capability Maturity Model
CMU	Carnegie Mellon University
COQ	Cost of Quality
KLOC	kiloline of code
LOC	line of code
PSP	Personal Software Process
SEI	Software Engineering Institute
TSP	Team Software Process

Khoa CNTT - ĐHKHTN Tp.HCM

Danh mục các bảng

Danh mục các hình vẽ

Lời mở đầu

Đi cùng với xu thế phát triển mạnh mẽ của ngành công nghệ khác trên thế giới, công nghệ phần mềm cũng đang mở ra một cánh cửa cho các tiếp cận tiến bộ. Khá nhiều công ty, tổ chức đã nhận thức được tầm quan trọng của ngành công nghệ này và đã có những bước tiếp cận đáng ghi nhận. Tuy nhiên, song song với những phát triển này, mặt hạn chế về chất lượng phần mềm vẫn đã và đang là mối quan tâm của nhiều người, nhiều tổ chức. Là sinh viên của bộ môn công nghệ phần mềm, chúng em đã được tiếp cận và tìm hiểu khá nhiều qui trình hỗ trợ và nâng cao chất lượng phần mềm. Khi bắt tay vào thực hiện luận văn, chúng em đã tìm hiểu một số qui trình phần mềm như: Agile, CMM, TSP và PSP. Mỗi qui trình trên đều có những mặt vượt trội riêng và nhìn chung mục đích chính của chúng cũng để nâng cao chất lượng sản phẩm phần mềm làm ra. Tuy nhiên, trong những qui trình này, chúng em lựa chọn PSP là đề tài cho luận văn vì những lý do sau:

- PSP hỗ trợ cho cá nhân : Để phát triển một phần mềm theo yêu cầu của khách hàng, chúng ta cần một đội ngũ nhiều kỹ sư. Nhưng chung quy lại thì chất lượng phần mềm lại phụ thuộc vào hiệu quả từng phần nhỏ mà từng cá nhân thực hiện. Do đó, nếu cải tiến, nâng cao chất lượng của từng cá nhân thì chất lượng của cả nhóm, cả tổ chức cũng được nâng cao.
- Có khả năng áp dụng cho bản thân: Mặc dù trên lí thuyết tìm hiểu những qui trình Agile, CMM, TSP đều mang lại những hiệu quả rất cao. Tuy nhiên, mục tiêu cải tiến của các qui trình này là các nhóm, các tổ chức phát triển phần mềm. Do đó, nếu đem áp dụng thực tế những qui trình này thì chúng em không có điều kiện. Ngược lại, với PSP, vì đây là qui trình hỗ trợ cho cá nhân nên chúng em có thể áp dụng những điều đã nghiên cứu được trên bản thân và đánh giá thực tế kết quả đạt được.
- Có khả năng áp dụng các lĩnh vực ngoài phạm vi phần mềm: Mặc dù PSP ra đời dựa trên nhu cầu quản lý quy trình phần mềm cho cá nhân, nhưng phạm vi sử dụng của nó không giới hạn ở công việc liên quan đến phần mềm. PSP còn có thể được áp dụng cho các loại công việc hàng ngày. Do đó, chúng em có thể học cách lập kế hoạch và quản lý tốt công việc của mình.

Chương 1. Tổng quan

1.1 Qui trình PSP là gì?

PSP là một qui trình phần mềm được lập ra để hỗ trợ cho các kỹ sư phần mềm trong việc lên kế hoạch, ước lượng và thực hiện các module hay chương trình phần mềm nhỏ.

Với PSP, các kỹ sư sẽ làm việc theo một qui trình có cấu trúc được định nghĩa sẵn, trong đó các công việc sẽ được lên kế hoạch, đánh giá, theo dõi tiến độ thực hiện và quản lý chất lượng. Các thông tin trong quá trình thực hiện công việc của kỹ sư sẽ được ghi nhận lại để cơ sở cho việc đánh giá và cải thiện hiệu quả làm việc của bản thân các kỹ sư.

1.2 Lịch sử ra đời của PSP

Sau khi Watt S.Humphrey đưa ra mô hình CMM năm 1987, ông đã quyết định ứng dụng những nguyên lý của CMM vào việc viết những chương trình nhỏ. Rất nhiều người đã đặt câu hỏi làm sao để áp dụng CMM cho những tổ chức nhỏ hay công việc của những nhóm nhỏ. Mặc dù CMM có khả năng áp dụng cho những tổ chức hay nhóm như vậy nhưng nó cũng đòi hỏi phải cung cấp thêm những hướng dẫn về cách phải thực hiện như thế nào. Humphrey đã quyết định sử dụng những nguyên tắc của CMM để phát triển những chương trình được phân thành các module để xem cách tiếp cận như vậy có khả thi không và cũng để tìm ra cách để thuyết phục các kỹ sư phát triển phần mềm sử dụng phương pháp trên.

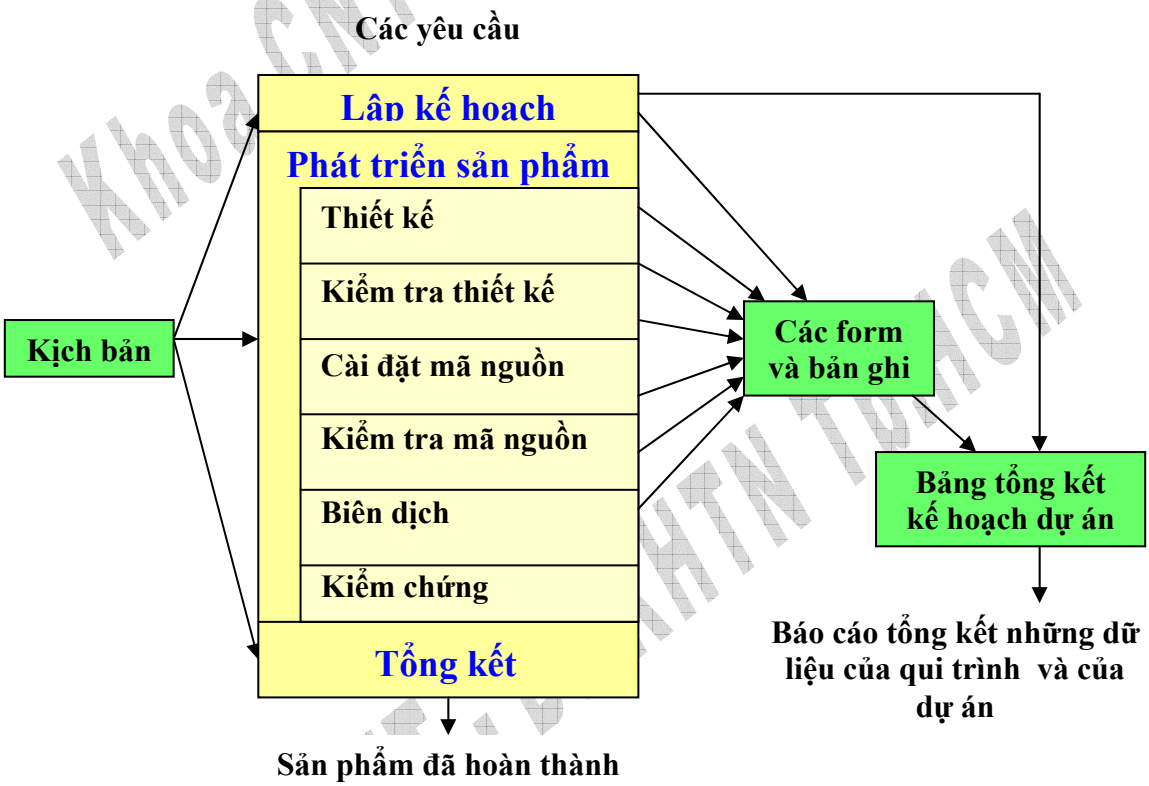
Trong khi phát triển những chương trình, Humphrey sử dụng tất cả các nguyên tắc của CMM lên tới cấp độ 5. Chỉ sau khi bắt đầu việc nghiên cứu thử nghiệm này, viện công nghệ phần mềm (SEI) đã cho phép ông sử dụng toàn bộ thời gian để nghiên cứu PSP. Trong vòng hơn 3 năm, ông đã viết tổng cộng 62 chương trình và định nghĩa khoảng 15 phiên bản của PSP. Ông đã sử dụng Pascal, Pascal hướng đối tượng và C++ để viết khoảng 25000 dòng code (LOC).

Humphrey đã viết một bản viết tay cho một vài tổ chức có ý định dạy PSP. Tháng 9/1993, Howie Dow đã mở khóa dạy PSP đầu tiên cho 4 sinh viên sau đại học tại đại học Massachusetts (Lowell). Humphrey cũng dạy PSP trong học kỳ đông của năm học 1993–1994 tại đại học Carnegie Mellon, cũng như NazimMadhavji ở đại học McGill và Soheil Khajanoori ở đại học Embry Riddle Aeronautical. Dựa trên những kinh nghiệm và dữ liệu thu thập được từ các khoá học này, Humphrey đã xem lại bản viết tay và đã xuất bản phiên

bản cuối cùng vào cuối năm 1994. Cùng khoảng thời gian đó, Jim Over và Neil Reizer của SEI và Robert Powels của Advancesd Imformation Service (AIS) phát triển khoá học đầu tiên huấn luyện các giáo viên dạy PSP trong công nghiệp.Từ đó cho đến nay, Watt Humphrey và SEI tiếp tục phát triển PSP và giới thiệu những nguyên tắc tương tự vào trong các nhóm kỹ sư phát triển phần mềm. [1]

1.3 Cấu trúc tổng quan quy trình PSP

Cũng giống như những quy trình phát triển phần mềm khác, để thực hiện một dự án phần mềm sử dụng qui trình PSP, mỗi kỹ sư đều phải trải qua 7 pha: lập kế hoạch, thiết kế, kiểm tra thiết kế, cài đặt mã nguồn, kiểm tra mã nguồn, biên dịch, kiểm chứng và tổng kết. Các pha này được phân thành 3 giai đoạn chính: giai đoạn lập kế hoạch, giai đoạn phát triển sản phẩm và giai đoạn tổng kết. Mô hình chung như sau:



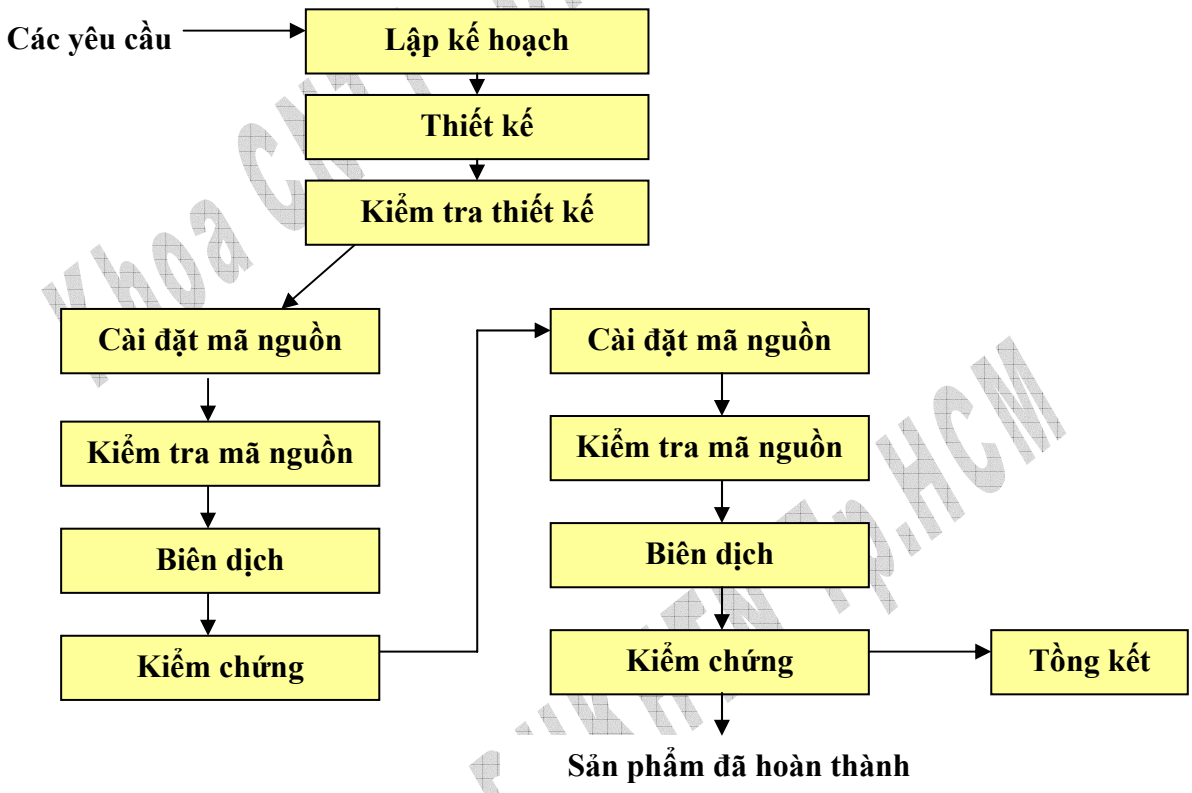
Hình 1.3.1 Dòng quy trình PSP

Trong mỗi pha, người kỹ sư được cung cấp những kịch bản kèm theo những biểu mẫu cần thiết để ghi nhận những thông tin, phục vụ cho giai đoạn tổng kết và cũng để làm dữ liệu hỗ trợ cho quá trình ước lượng về sau.

Kết thúc dự án nghĩa là khi sản phẩm đã hoàn thành, người kỹ sư sẽ tổng hợp những thông tin ghi nhận trong các biểu mẫu và bản ghi vào bản tổng kết kế hoạch dự án. Đây

chính là cơ sở để đánh giá hiệu quả làm việc trong dự án cũng như là cơ sở để thấy được sự tiến bộ sau khi sử dụng qui trình PSP.

Điểm lợi thế của PSP cũng như những qui trình khác là không bó buộc người sử dụng trong một khung qui trình chuẩn. Có những dự án mà mức độ phức tạp tập trung vào một số pha nào đó. Trong những trường hợp như vậy, người kỹ sư có thể phân dự án thành các module nhỏ để phát triển. Ví dụ như, người kỹ sư A nhận được một dự án mà độ phức tạp tập trung ở giai đoạn viết mã nguồn cho chương trình. Anh ta có thể thiết kế cho chương trình và sau đó phân thành nhiều module nhỏ để cài đặt. Cấu trúc qui trình sau hỗ trợ kỹ sư phát triển 2 module theo yêu cầu trên:



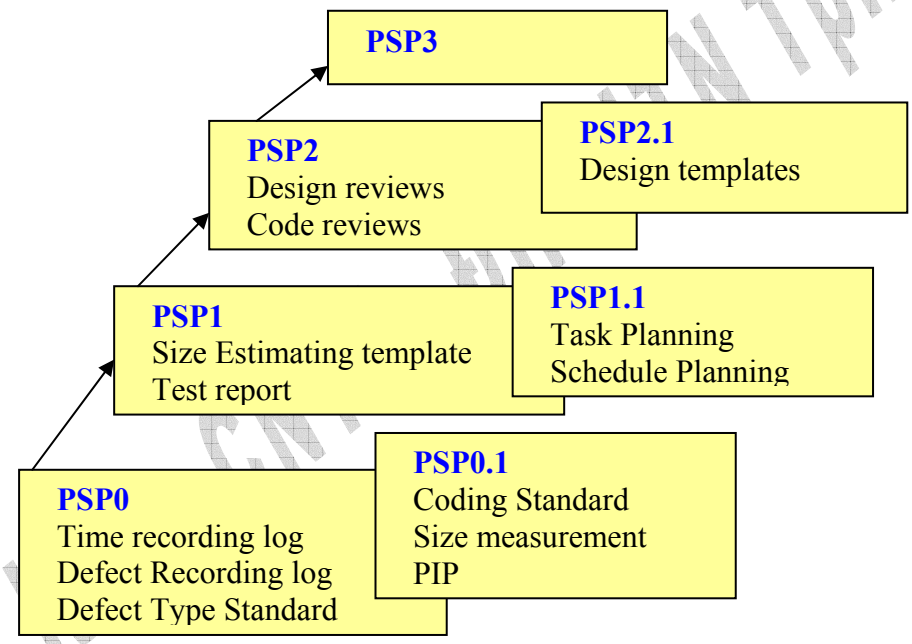
Hình 1.3.2 Ví dụ cấu trúc quy trình cho chương trình có 2 module cài đặt

1.4 Các cấp độ của PSP

Cũng giống như qui trình CMM, PSP cũng dựa trên những nguyên lý cải tiến qui trình. Tuy nhiên, trong khi CMM tập trung vào việc cải tiến khả năng của toàn bộ tổ chức thì PSP lại tập trung vào cải tiến khả năng của mỗi cá nhân phát triển phần mềm.

Mục đích chính của PSP là giúp nâng cao hiệu quả làm việc, khả năng ước lượng thời gian, sai sót, và khả năng lập kế hoạch làm việc của mỗi kỹ sư. Đó là những kỹ năng mà các kỹ sư thường ít quan tâm đến hoặc nếu có quan tâm thì cũng chưa đạt được hiệu quả

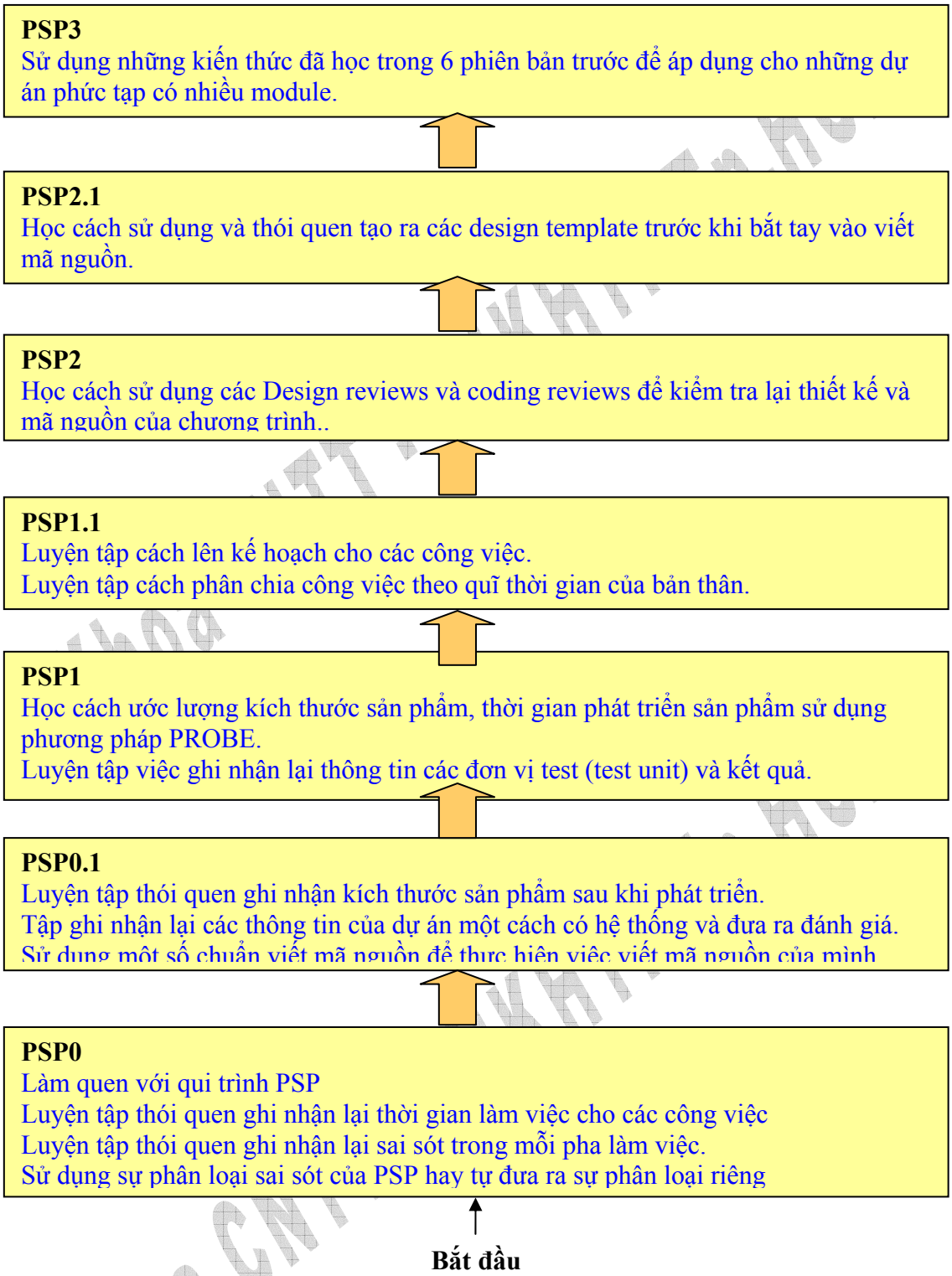
lắm. Watt S. Humphrey, sau một quá trình nghiên cứu dài, đã đưa ra 7 cấp độ chính thức. Mức độ phức tạp đi từ PSP0 đến PSP3. Ở mỗi cấp độ, người kỹ sư sẽ làm quen và luyện tập với một số kỹ năng nào đó thông qua một số biểu mẫu và template. Mô hình các cấp độ như sau:



Hình 1.4.1 Các cấp độ của quy trình PSP

Sau khi hoàn thành mỗi cấp độ, người kỹ sư sẽ biết được một số phương pháp để quản lý quy trình và hiệu quả làm việc của mình. Tuy nhiên, cần lưu ý là hiệu quả của quy trình đối với mỗi cá nhân tỉ lệ thuận với thời gian sử dụng PSP một cách liên tục. Nghĩa là, việc áp dụng PSP ban đầu có thể làm cho mỗi cá nhân cảm thấy khó khăn vì nó đòi hỏi phải thay đổi một số thói quen hàng ngày. Sau một thời gian áp dụng, mức độ thành thạo tăng dần và những công việc như vậy trở thành thói quen tốt. Đó chính là lúc chúng ta sẽ nhận ra được hiệu quả của PSP đối với việc cải tiến chất lượng làm việc của bản thân.

Mặc dù mỗi cấp độ cung cấp một số phương pháp nào đó cho người sử dụng nhưng kịch bản thực hiện cho các cấp độ đều phải tuân thủ đúng các pha trong cấu trúc quy trình PSP, nghĩa là cũng phải đi qua 3 giai đoạn: lên kế hoạch, phát triển sản phẩm và tổng kết. Mỗi giai đoạn đều phải đi theo các pha đúng với thứ tự mô tả trong cấu trúc tổng quan quy trình PSP ở trên.



Hình 1.4.2 Thứ tự thực hiện các cấp độ của PSP

Một kỹ sư hay cá nhân muốn tìm hiểu và sử dụng PSP thì nên thực hiện theo thứ tự các cấp độ trong hình 1.4.2 trên. Sau khi đã thuần thục một cấp độ thì sẽ sang cấp độ cao hơn. Bảy cấp độ này là một quá trình bổ sung, những cấp độ sau bổ sung cho cấp độ trước.

Chính vì thế sau khi hoàn thành 7 cấp độ, mỗi cá nhân sẽ có đầy đủ các kỹ năng mà PSP muốn cung cấp.

1.5 Ưu và khuyết điểm của PSP.

1.5.1 Ưu điểm

PSP cung cấp cho các kỹ sư phần mềm những phương pháp cụ thể giúp họ cải tiến chất lượng công việc và nâng cao hiệu quả qui trình làm việc cá nhân.

PSP giúp người kỹ sư nâng cao khả năng dự đoán, lập kế hoạch chính xác và theo dõi chặt chẽ tiến độ thực hiện công việc của mình. Điều này giúp cho họ chủ động hơn trong việc nhận các công việc được giao và thực hiện các công việc.

Huấn luyện các kỹ sư làm việc theo qui trình PSP để tạo nguồn lực xây dựng các nhóm phát triển phần mềm theo qui trình TSP.

PSP được chứng minh là cách tiếp cận khá hiệu quả trong việc cải tiến qui trình phát triển từ CMM cấp 3 lên CMM cấp 4. Lỗ hỗng giữa hai cấp độ này là các phương pháp đo và các phương pháp rèn luyện. PSP giúp cho các kỹ sư tin tưởng vào tầm quan trọng của các phương pháp đo và do đó giảm được sự quá tải của các công việc trong cấp 4.[2]

1.5.2 Khuyết điểm.

Việc đem áp dụng PSP vào trong môi trường sản xuất vẫn còn gặp nhiều khó khăn vì mức độ nghiêm ngặt, chi tiết của nó.

Bản thân PSP không bảo đảm đem lại kết quả tốt nhất về hiệu quả và chất lượng sản phẩm. Ngược lại, nếu PSP được kết hợp vào trong một quy trình phát triển có tính chất lặp đi lặp lại nhiều lần thì PSP sẽ hứa hẹn đem lại nhiều kết quả tốt hơn.[2]

1.6 Mối liên hệ giữa CMM, TSP và PSP [3]

CMM là chuẩn:

- Cho phép một tổ chức đánh giá cấp độ trưởng thành của tổ chức đó về quy trình phát triển phần mềm
- Chủ yếu nhằm vào các thực tiễn quản lý
- Không đưa ra sự hướng dẫn cụ thể nào cho sự thực hiện của cá nhân

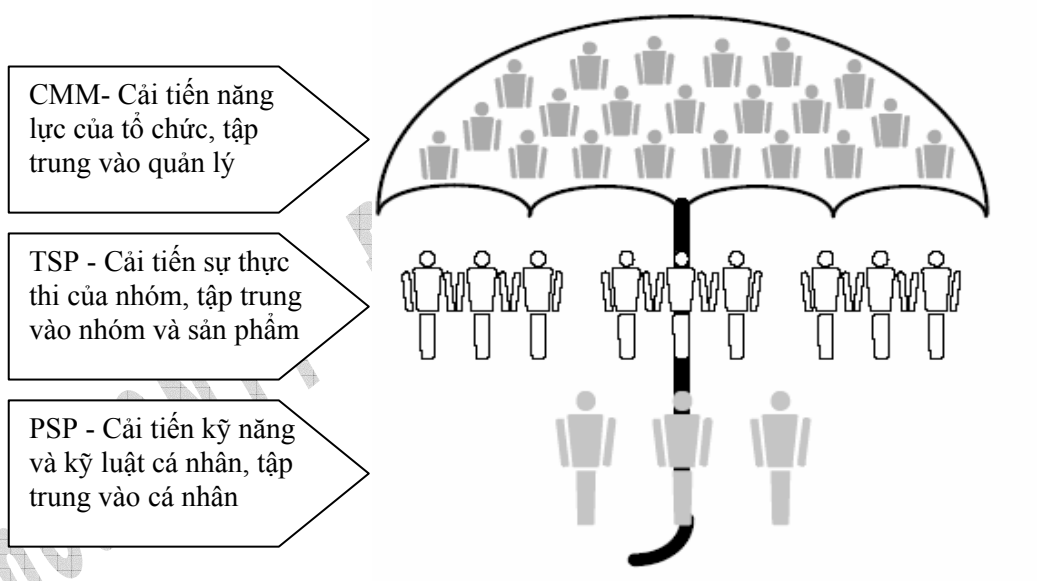
PSP là quy trình:

- hoàn toàn xác định, nghiêm ngặt cho các các nhân

TSP là quy trình:

- Chỉ ra các thực tiễn có độ trưởng thành cao cho các nhóm làm việc của các kỹ sư được huấn luyện PSP
- Là cầu nối giữa PSP và CMM

Mối liên hệ giữa CMM, TSP, PSP được thể hiện ở hình sau:



Hình 1.6.1 Mối liên hệ giữa CMM, TSP, PSP

Chương 2. Các phương pháp luận trong PSP về quy trình lập kế hoạch [4]

2.1 Nguyên lý quản lý thời gian

2.1.1 Logic của quản lý thời gian

Cơ sở logic cho quản lý thời gian như sau:

Thông thường, bạn sẽ sử dụng thời gian trong tuần này giống như cách bạn sử dụng thời gian trong tuần trước. Tuy nhiên, cũng có nhiều ngoại lệ. Ví dụ, trong suốt tuần thi cử, bạn có thể không lên lớp học, và bạn bỏ ra nhiều thời gian vào việc học bài và ít thời gian để làm bài tập hơn các tuần bình thường trước đó.

Để đưa ra được kế hoạch thực tế, bạn phải theo dõi cách sử dụng thời gian. Bạn nghĩ rằng bạn biết rõ mình đã sử dụng thời gian tuần trước ra sao, nhưng bạn sẽ phải ngạc nhiên bởi dữ liệu thực. Trí nhớ của chúng ta có xu hướng thấy thời gian dùng vào những việc ta thích trôi qua nhanh hơn. Ngược lại, những hoạt động chậm rãi, tẻ nhạt hoặc khó khăn dường như dài hơn thực tế. Chính vì vậy, để biết thời gian của bạn đã đi đâu, bạn cần phải ghi chép chính xác.

Để kiểm tra độ chính xác của việc bạn ước lượng thời gian và lập kế hoạch, bạn cần phải ghi lại sưu liệu và sau này so sánh chúng với những gì bạn thực sự làm. Lập kế hoạch là một kỹ năng mà ít người nắm được. Tuy nhiên, có những phương pháp lập kế hoạch có thể học và luyện tập được. Bước đầu tiên trong việc học lập một kế hoạch tốt là phải lập kế hoạch. Sau đó viết ra kế hoạch của bạn vì vậy bạn sẽ có cái để so sánh với số liệu thực sự sau này.

Để đưa ra được những kế hoạch chính xác hơn, hãy định rõ kế hoạch trước đó của bạn sai sót ở đâu và bạn có thể làm điều gì tốt hơn. Khi bạn thực hiện công việc có kế hoạch, hãy ghi lại thời gian bạn đã sử dụng. Những dữ liệu thời gian này sẽ hữu ích nhất nếu được ghi chép chi tiết. Ví dụ, khi thực hiện công việc của khóa học, hãy ghi chép riêng rẽ thời gian bạn sử dụng trên lớp, đọc giáo trình, viết chương trình và ôn thi. Khi viết những chương trình lớn hơn, bạn cũng sẽ thấy hữu ích khi ghi lại thời gian cho những phần khác nhau của công việc - thiết kế chương trình, viết mã, biên dịch và kiểm thử. Mức độ chi tiết như thế này không cần thiết cho những công việc quá ngắn nhưng sẽ có ích khi làm việc trên những đề án chiếm nhiều hơn một vài giờ.

Khi bạn có một bản tài liệu kế hoạch của mình và ghi lại thời gian thực sự bạn sử dụng, bạn có thể dễ dàng so sánh các kết quả thực tế với kế hoạch ban đầu. Khi đó bạn sẽ thấy kế hoạch sai sót ở đâu và tiến trình lập kế hoạch của bạn có thể cải tiến như thế nào. Bí quyết của việc lập kế hoạch chính xác là lập kế hoạch phù hợp và so sánh chúng với mỗi kết quả thật sự sau đó. Khi đó bạn sẽ thấy được cách để lập được kế hoạch tốt hơn.

Để quản lý thời gian của mình, hãy lập kế hoạch cho thời gian và làm theo kế hoạch đó. Đưa ra kế hoạch tốt đối với chúng ta là điều dễ dàng, nhưng thật ra thực hiện kế hoạch mới là việc khó khăn hơn nhiều. Thế giới có đầy những giải pháp mà không bao giờ thực hiện được. Ban đầu, làm theo kế hoạch có vẻ khó khăn. Có nhiều nguyên nhân nhưng nguyên nhân thường thấy nhất là kế hoạch không tốt lắm. Bạn sẽ không biết điều này cho đến khi bạn cố gắng để làm theo nó. Bằng cách làm việc với kế hoạch, lợi ích đầu tiên là bạn biết được kế hoạch sai sót ở chỗ nào, điều gì sẽ giúp bạn lập kế hoạch tốt hơn trong đề án tiếp sau.

Lợi ích thứ 2 của làm việc theo kế hoạch là bạn sẽ làm việc với cái cách mà bạn lập kế hoạch. Điều này dường như có vẻ không quan trọng, nhưng nó thực sự quan trọng. Nhiều vấn đề trong công nghệ phần mềm xảy ra là do đi tắt thiếu thận trọng, bất cẩn và không chú ý đến chi tiết. Trong hầu hết các trường hợp, các phương pháp tốt đều được biết và chỉ định nhưng không được làm theo. Học cách để lập ra những kế hoạch hữu dụng là quan trọng, nhưng việc học để làm việc theo kế hoạch mới đóng vai trò quyết định tuyệt đối.

Một lợi ích khác khó thấy hơn của làm việc theo kế hoạch là nó thật sự thay đổi hành vi của bạn. Với một kế hoạch, bạn ít tiêu phí thời gian vào việc quyết định phải làm gì tiếp theo. Kế hoạch còn giúp bạn tập trung vào những việc bạn đang làm, ít bị xao lãng và làm việc hiệu quả hơn.

2.1.2 Hiểu cách mình sử dụng thời gian

Để luyện tập quản lý thời gian, bước đầu tiên là hiểu cách bạn sử dụng thời gian hiện tại như thế nào. Điều này cần các bước sau:

Phân loại các hoạt động chính. Khi bạn bắt đầu theo dõi thời gian, bạn có thể nhận ra rằng bạn bỏ hầu hết thời gian vào một số tương đối ít hoạt động. Điều này là bình thường. Để đạt được cái gì đó, chúng ta phải tập trung vào một ít việc quan trọng nhất. Nếu bạn chia nhỏ thời gian vào quá nhiều loại thì bạn sẽ khó khăn để làm cho dữ liệu có ý nghĩa. Nếu sau này bạn cần chi tiết hơn, hãy chia các loại tổng quát thành các loại con.

Hãy ghi lại thời gian sử dụng cho mỗi hoạt động chính. Điều này cần nhiều kỷ luật cá nhân để ghi lại thời gian một cách nhất quán. Để ghi lại chính xác, hãy ghi lại thời gian bắt đầu và kết thúc của mỗi loại công việc chính.

Ghi lại thời gian theo một phương pháp chuẩn. Chuẩn hóa bản ghi thời gian là cần thiết vì khối lượng số liệu thời gian sẽ tăng lên nhanh chóng. Nếu bạn không quản lý dữ liệu đúng đắn, bạn sẽ không tập hợp dữ liệu lại được.

Giữ số liệu thời gian ở một nơi thuận tiện để dễ dàng cho bạn trong việc ghi chép.

2.2 Theo dõi thời gian

2.2.1 Tại sao phải theo dõi thời gian?

Phương pháp cải tiến chất lượng công việc của bạn bắt đầu bằng cách biết được hiện tại bạn đang làm gì. Điều này có nghĩa là bạn phải biết các công việc bạn làm, bạn làm chúng như thế nào và các kết quả bạn đạt được. Bước đầu tiên trong quy trình này là định nghĩa ra các công việc và tìm ra thời gian bạn sử dụng cho mỗi công việc là bao nhiêu. Để làm điều này, bạn phải đo thời gian thật. Phần này mô tả cách đo thời gian và đưa ra một biểu mẫu giúp bạn làm điều này.

2.2.2 Ghi lại số liệu thời gian

Khi ghi chép thời gian, nhớ rằng mục tiêu là thu được dữ liệu về việc bạn thật sự làm việc như thế nào. Cách thức và thủ tục được sử dụng để thu thập dữ liệu thì không quan trọng, miễn là dữ liệu chính xác và đầy đủ. Bản thân tác giả Humphrey đã sử dụng phương pháp mô tả trong chương này và đã dạy nó trong nhiều khóa học cho nhiều sinh viên cũng như cho nhiều kỹ sư đang làm việc, và họ đã đạt được những thành công đáng kể. Sau khi họ vượt qua sức ì tự nhiên của bản thân để sử dụng các biểu mẫu và các thủ tục, hầu hết các kỹ sư nhận thấy rằng các phương pháp này vừa đơn giản vừa tiện lợi. Có lẽ bạn cũng sẽ như vậy.

Khi bạn làm việc với tài liệu này, hãy sử dụng các phương pháp được mô tả để ghi chép lại thời gian. Có thể bạn tự hỏi: "Tại sao tôi nên dùng biểu mẫu này mà không tự thiết kế biểu mẫu của chính mình?" Câu trả lời là:

- Không có kinh nghiệm thu thập dữ liệu từ trước, để nghĩ ra biểu mẫu và thủ tục cá nhân khả thi sẽ khó khăn.

- Sau khi hoàn thành khóa học này, bạn sẽ có kiến thức và kinh nghiệm để sửa đổi các biểu mẫu và thủ tục cho phù hợp với bạn.

2.2.3 Đơn vị đo thời gian của bạn

Theo lệ thường khi người ta nói về công việc của mình, họ thường sử dụng đơn vị đo là giờ. Điều này cho thấy không có ích lắm vì bạn hiếm khi làm việc gì liên tục trong 1 giờ. Hơn nữa. đo công việc bằng đơn vị giờ sẽ không cung cấp chi tiết cần thiết cho kế hoạch và quản lý công việc sau này. Theo dõi thời gian bằng phút sẽ dễ hơn nhiều. Ví dụ, hãy nghĩ xem bản ghi chép thời gian ra sao nếu bạn sử dụng các phần nhỏ của giờ. Các mục nhập vào sẽ có những số như là 0,38 hay 1,27 giờ, những lượng nhỏ li ti này vừa khó khăn để tính toán và vừa khó hiểu. Thay vì là 0,38 giờ thì 23 phút sẽ dễ hiểu và dễ ghi chép hơn. Một khi bạn đã quyết định theo dõi thời gian thì theo dõi bằng phút cũng chẳng khó khăn gì hơn bằng giờ cả.

2.2.4 Sử dụng bản ghi chép thời gian (Time Recording Log)

Dưới đây là biểu mẫu sử dụng để ghi chép thời gian, các hướng dẫn để ghi chép chúng và ví dụ:

Sinh viên _____ Ngày _____
Người hướng dẫn _____ Lớp _____

Ngày	Bắt đầu	Kết thúc	Thời gian gián đoạn	Thời gian Delta	Hoạt động	Ghi chú	C	U

Bảng 2.2.1 Bản ghi ghi chép thời gian

Tổng quát	Ghi lại thời gian bằng phút, càng chính xác càng tốt.
Đầu trang	Ghi vào: - Tên bạn và ngày hiện tại. - Tên người hướng dẫn và tên hoặc số của khóa học.
Ngày	Ghi ngày khi một mục được thêm vào
Ví dụ	14/9/96
Bắt đầu	Nhập thời gian khi bạn bắt đầu một công việc.
Ví dụ	9g15
Kết thúc	Nhập thời gian khi bạn kết thúc công việc đó.
Ví dụ	11g59
Thời gian gián đoạn	Ghi lại bất cứ thời gian gián đoạn nào không được dùng cho công việc và lý do gián đoạn. Nếu bạn có một vài gián đoạn thì nhập vào tổng thời gian của chúng.
Ví dụ	5+3+22, nghỉ giải lao, điện thoại, tán gẫu
Thời gian Delta	Nhập vào thời gian trôi chảy bạn trải qua cho công việc, trừ đi thời gian gián đoạn.
Ví dụ	Từ 9g15 đến 11g59, trừ đi 30 phút hay là bằng 134 phút
Hoạt động	Nhập vào tên hoặc sự định rõ khác của công việc hoặc hoạt động đang được làm.
Ví dụ	Xem lại
Lời chú giải	Điền vào bất cứ lời chú giải thích đáng nào mà sau này có thể sẽ nhắc cho bạn nhớ về bất cứ tình huống đáng chú ý nào liên quan đến hoạt động này.
Ví dụ	Chuẩn bị cho vấn đáp
C (đã hoàn tất)	Khi một công việc được hoàn tất, đánh dấu vào ô này.
Ví dụ	Vào 7g45 ngày 9/9, bạn hoàn tất việc đọc 1 hay nhiều hơn 1 chương tài liệu, hãy đánh dấu vào ô này.
U (đơn vị)	Điền số đơn vị của công việc bạn hoàn tất.
Ví dụ	Từ 6g25 đến 7g45 ngày 9/9 bạn đọc 2 chương tài liệu, vậy điền 2 vào.
Quan trọng	Ghi lại tất cả thời gian của bạn cho khóa học này. Nếu bạn quên ghi chép lại một thời gian, lập tức điền vào ước lượng tốt nhất của bạn. Nếu bạn để quên bản ghi chép thời gian, ghi chú lại các thời gian và sao chép chúng lại trong bản ghi chép của bạn ngay khi có thể.

Bảng 2.2.2 Các hướng dẫn bản ghi ghi chép thời gian

Sinh viên Sinh viên Y **Ngày** 9/9/96

Người hướng dẫn Thầy Z **Lớp** CS1

Ngày	Bắt đầu	Kết thúc	Thời gian gián đoạn	Thời gian Delta	Hoạt động	Lời chú giải	C
9/9	9:00	9:50		50	Lớp học	Bài giảng	
	12:40	1:18		38	Chương trình	Bài tập 1	
	2:45	3:53	10	58	Chương trình	Bài tập 1	
	6:25	7:45		80	Tài liệu	Đọc sách – Ch 1&2	X
10/9	11:06	12:19	6+5	62	Chương trình	Bài tập 1, nghỉ giải lao, tán gẫu	X
11/9	9:00	9:50		50	Lớp học	Bài giảng	
	1:15	2:35	3+8	69	Chương trình	Bài tập 2, nghỉ giải lao, điện thoại	X
	4:18	5:11	25	28	Tài liệu	Tài liệu Ch3, tán gẫu với Mary	X
12/9	6:42	9:04	10+6+2	114	Chương trình	Bài tập 3	X
13/9	9:00	9:50		50	Lớp học	Bài giảng	
	12:38	1:16		38	Tài liệu	Tài liệu Ch4	
14/9	9:15	11:59	5+3+22	134	Xem lại	Chuẩn bị kiểm tra vấn đáp, nghỉ giải lao, điện thoại, tán gẫu	

Bảng 2.2.3 Ví dụ bản ghi ghi chép thời gian

2.2.5 Quản lý các gián đoạn

Một vấn đề phổ biến trong theo dõi thời gian là các gián đoạn. Chúng ta thường xuyên bị gián đoạn bởi các cú điện thoại, tán gẫu, thỉnh thoảng bị làm phiền hoặc nhu cầu muốn nghỉ giải lao. Cách quản lý các gián đoạn bằng bản ghi chép thời gian là ghi chú chúng trong cột thời gian gián đoạn. Sinh viên Y không chỉ ghi chú những thời gian này trong bản ghi chép thời gian mà còn mô tả ngắn gọn các gián đoạn trong cột lời chú giải.

Một mẹo nhỏ có ích ở đây là sử dụng đồng hồ bấm giờ cho việc theo dõi các gián đoạn. Khi bị gián đoạn thì bắt đầu bấm giờ, và khi trở lại với công việc thì bấm giờ để ngừng. Điều này thuận tiện hơn ghi lại thời gian bắt đầu lúc bị gián đoạn và cũng chính xác hơn là đoán.

Bởi vì thời gian gián đoạn không phải là thời gian có ích cho công việc, bạn phải theo dõi các gián đoạn này. Nếu lượng thời gian này cố định, cách bạn quản lý nó không khác nhau nhiều lắm. Tuy nhiên, thời gian gián đoạn lại rất hay thay đổi. Nếu bạn không dự liệu được nó, bạn sẽ phải thêm một con số ngẫu nhiên vào tất cả các dữ liệu thời gian. Điều này sẽ rất khó khăn cho sử dụng những dữ liệu này để lập kế hoạch và quản lý thời gian.

Số liệu bản ghi chép thời gian có thể được dùng để biết được công việc của bạn thường xuyên bị gián đoạn như thế nào. Gián đoạn không chỉ là thời gian lãng phí mà chúng còn phá vỡ dòng suy nghĩ của bạn, dẫn đến việc không hiệu quả và sai sót. Biết được bạn bị gián đoạn nhiều hay ít có thể sẽ giúp cải tiến chất lượng và hiệu quả công việc của bạn. Một số kỹ sư nói rằng học cách để kiểm soát số lần và khoảng thời gian gián đoạn là một trong các lợi ích quan trọng nhất của họ khi theo dõi thời gian.

2.2.6 Theo dõi các công việc đã hoàn tất

Để theo dõi việc bạn sử dụng thời gian như thế nào, bạn cũng cần phải theo dõi các kết quả đạt được. Ví dụ, khi phát triển chương trình, đọc sách hay viết báo cáo, bạn cần phải biết bao nhiêu công việc đã hoàn tất. Sau đó bạn có thể tính được năng suất làm việc như là cần bao nhiêu thời gian để đọc một chương tài liệu hay viết một chương trình. Với kiến thức này, bạn sẽ lập được kế hoạch tốt hơn cho công việc tương lai.

Cột C và U bên phải của bản ghi chép thời gian là viết tắt của *Completed* (đã hoàn tất) và *Units* (Đơn vị). Những cột này sẽ giúp cho việc xác định nhanh chóng thời gian đã bỏ ra cho các công việc khác nhau và công việc nào đã hoàn thành.

Ở đây, đơn vị là *đơn vị công việc*. Khi bạn đọc xong một chương, bạn đã hoàn thành một đơn vị công việc. Một chương trình đã hoàn tất là một đơn vị khác của công việc. Trong các phần sau sẽ thảo luận chi tiết hơn về các độ đo đơn vị công việc.

Để bản ghi chép thời gian được chính xác, quan trọng là phải điền vào cột C và U mỗi khi bạn hoàn tất một công việc mà có kết quả có thể đo được. Nếu bạn quên làm điều này, bạn có thể luôn luôn tìm thấy thông tin, nhưng sẽ dễ dàng hơn nhiều khi điền nó vào thời gian bạn hoàn tất công việc.

2.2.7 Gợi ý về việc ghi chép thời gian

Khái niệm theo dõi thời gian thì đơn giản. Tuy nhiên một vài gợi ý có thể giúp bạn thực hiện một cách nhất quán và chính xác:

- Đôi khi thỉnh thoảng bạn quên ghi lại thời gian bắt đầu, kết thúc hay khoảng thời gian gián đoạn, hãy ước lượng ngay khi bạn nhớ đến. Điều này sẽ không chính xác bằng việc bạn ghi lại thời gian chính xác nhưng đây là điều tốt nhất bạn có thể làm. Thông thường nó sẽ gần giống với thời gian thực tế.

- Bạn có thể sử dụng đồng hồ bấm giờ để theo dõi các gián đoạn. Điều này dường như chính xác quá mức, nhưng như vậy sẽ dễ hơn việc ghi lại thời gian bắt đầu và kết thúc của mỗi gián đoạn.

- Tổng kết thời gian của bạn nhanh chóng. Bạn sẽ sử dụng một bản ghi chép hoạt động hàng tuần để tổng kết thời gian hàng tuần trong khóa học này. Làm như thế nào và tại sao phải làm như vậy sẽ được thảo luận các phần kế tiếp.

Một cách tiếp cận khác là ghi chép lại số liệu thời gian lên máy vi tính. Tôi đã thử làm điều này và thấy nó tốn thời gian hơn và ít thoải mái hơn là ghi chú trên giấy. Các hệ thống máy tính là lý tưởng cho mục đích này, nhưng cần đến ứng dụng hỗ trợ thích hợp.

2.3 Lập kế hoạch sản phẩm và kế hoạch giai đoạn

Phần này mô tả việc lập kế hoạch sản phẩm và kế hoạch giai đoạn, chỉ ra mối liên hệ giữa chúng với công việc cá nhân của bạn. Bạn sẽ học cách để hoàn tất bản tóm tắt hoạt động hàng tuần từ số liệu bản ghi chép thời gian bạn đã ghi lại từ trước đến giờ.

2.3.1 Các kế hoạch sản phẩm và giai đoạn

Có 2 loại kế hoạch: Loại đầu tiên dựa trên một giai đoạn thời gian nào đó, ví dụ như một ngày, một tuần, một tháng, một năm. Kế hoạch giai đoạn thể hiện cách bạn sử dụng thời gian trong giai đoạn đó. Loại thứ hai của kế hoạch là dựa trên hoạt động như viết một chương trình hay một báo cáo. Các kết quả có thể hữu hình như là các chương trình hay báo cáo, hay vô hình như kiến thức nhận được từ việc đọc tài liệu hay dịch vụ bạn cung cấp khi làm việc trong văn phòng.

Để thấy được sự khác nhau giữa lập kế hoạch sản phẩm và lập kế hoạch giai đoạn, hãy xem xét việc đọc quyển sách này. Để lập kế hoạch công việc, đầu tiên bạn ước lượng thời gian cho toàn bộ công việc. Ví dụ, bạn có thể cho rằng cần 20 giờ để đọc 20 chương trong toàn bộ sách. Đối với kế hoạch sản phẩm, khi đó bạn nên lên lịch thời gian để đọc, ví dụ một giờ một tuần. Kế hoạch sản phẩm cho công việc này khi đó có mục tiêu là đọc các chương sách trong 20 giờ. Kế hoạch giai đoạn sẽ là cách bạn phân phối thời gian đọc liên tục một giờ hàng tuần.

Đối với bạn, các kế hoạch sản phẩm và kế hoạch giai đoạn đều quan trọng vì công việc của bạn là dựa trên sản phẩm và cuộc sống của bạn sẽ ở trong các giai đoạn. Bạn không thể lập kế hoạch hoàn chỉnh cho một kế hoạch này mà không lập kế hoạch cho cái kia. Phần tiếp theo tập trung vào việc lập kế hoạch giai đoạn và sau đó là tập trung cho lập kế hoạch sản phẩm.

2.3.2 Bản tổng kết hoạt động hàng tuần

Để lập kế hoạch giai đoạn, điều quan trọng là hiểu được cách bạn dùng thời gian như thế nào. Bước đầu tiên là theo dõi thời gian sử dụng bản ghi chép thời gian. Sau khi tập hợp số liệu thời gian của 1 hay 2 tuần, bạn sẽ bắt đầu xem bạn đã sử dụng thời gian như thế nào. Bạn cần phải tổng kết số liệu trong một biểu mẫu hữu ích hơn vì các bản ghi thời gian quá chi tiết cho việc lập kế hoạch. Bản tổng kết hoạt động hàng tuần thể hiện trong bảng sau, định dạng cho dữ liệu thời gian, vì vậy chúng thuận tiện hơn cho việc lập kế hoạch giai đoạn.

	Tên						**Ngày**			
1	Công việc									Tổng cộng
2	Ngày									
3	Chủ nhật									
4	Thứ 2									
5	Thứ 3									
6	Thứ 4									
7	Thứ 5									
8	Thứ 6									
9	Thứ 7									
10	Tổng cộng									
11	**Thời gian và tốc độ giai đoạn**						**Số tuần (số trước +1)**			
12	**Thời gian của tuần trước**									
13	Tổng cộng									
14	Trung bình									
15	Max									
16	Min									
17	**Thời gian của tuần hiện tại**									
18	Tổng cộng									
19	Trung bình									
20	Max									
21	Min									

Bảng 2.3.1 Bảng tổng kết hoạt động hàng tuần

Dòng 1 đến dòng 10 của bản tổng kết cung cấp một hồ sơ về thời gian bạn bỏ ra cho mỗi hoạt động chính trong mỗi ngày của tuần trước đó. Phía dưới, từ dòng 13 đến 16 là thời gian trung bình, tối đa và tối thiểu bạn bỏ ra cho mỗi loại công việc suốt những tuần gần đây của học kỳ. Dòng 18 đến 21 chỉ ra thời gian tổng cộng, trung bình, tối đa, tối thiểu bạn bỏ ra mỗi loại công việc cho toàn bộ học kỳ đến nay, bao gồm tuần gần đây nhất. Từ ví dụ bản ghi ghi chép thời gian ở bảng 2.2.3, chúng ta có bản tổng kết hoạt động hàng tuần như sau:

Tên Sinh viên Y **Ngày** 16/9/96

	Công việc	Lớp học	Viết chương trình	Chuẩn bị vấn đáp	Đọc tài liệu					Tổng cộng
2	Ngày									
3	Chủ nhật 8/9									
4	Thứ 2	50	96		80					226
5	Thứ 3		62							62
6	Thứ 4	50	69		28					147
7	Thứ 5		114							114
8	Thứ 6	50			38					88
9	Thứ 7			134						134
10	Tổng cộng	150	341	134	146					771

11 Thời gian và tốc độ giai đoạn _____ **Số tuần (số trước +1)** **1** _____

12 Thời gian của tuần trước

13	Tổng cộng									
14	Trung bình									
15	Max									
16	Min									

17 Thời gian của tuần hiện tại

18	Tổng cộng	150	341	134	146					771
19	Trung bình	150	341	134	146					771
20	Max	150	341	134	146					771
21	Min	150	341	134	146					771

Bảng 2.3.2 Ví dụ bản tổng kết hoạt động hàng tuần

Khi bạn muốn lập kế hoạch cho tuần kế tiếp, hãy bắt đầu với bảng tổng kết hoạt động hàng tuần gần nhất. Dựa vào thời gian trước đó bỏ ra cho mỗi công việc, bạn có thể phán đoán thời gian bạn bỏ ra cho những công việc này trong tuần tới là bao nhiêu. Cách đơn giản nhất để lập kế hoạch là giả thiết rằng trong tương lai bạn sẽ sử dụng một lượng thời gian trung bình tương tự như trong quá khứ. Một cách tiếp cận phức tạp hơn là xem xét xem công việc định hoàn thành trong tuần tới và đoán xem nó sẽ rơi vào thời gian nào

của khoảng giữa thời gian tối đa và tối thiểu của tuần trước. Điều này thường cho một kế hoạch chính xác hơn. Các phần sau mô tả cách để hoàn tất bản tổng kết hoạt động hàng tuần.

2.3.3 Tính toán khoảng thời gian và tốc độ

Một bảng tổng kết đơn giản thời gian hàng tuần của bạn có lẽ sẽ đủ nếu bạn chỉ phải bận tâm về 1 tuần, nhưng thật ra bạn quan tâm đến thời gian trung bình, tối đa và tối thiểu của các công việc này qua một học kỳ hoặc thậm chí là 1 năm. Để thấy được cách thu thập các dữ liệu này, chúng ta tiếp tục hoàn tất bản tổng kết hoạt động hàng tuần trong bảng 2.3.3. Việc này sử dụng bản tổng kết hoạt động hàng tuần của sinh viên Y trong tuần 1 (bảng 2.3.2) và bản ghi thời gian của tuần thứ 2 (không được biểu diễn ở đây). Sử dụng những dữ liệu này, phần tốc độ và khoảng thời gian trong bảng 2.3.3 được hoàn tất như sau:

	Tên	Sinh viên Y					**Ngày**	23/9/96		
1	**Công việc**	**Lớp học**	**Viết chương trình**	**Chuẩn bị vấn đáp**	**Đọc tài liệu**				**Khác**	**Tổng cộng**
2	**Ngày**									
3	Chủ nhật 15/9									
4	Thứ 2	50	93		80					223
5	Thứ 3		95							95
6	Thứ 4	50			71					121
7	Thứ 5		77							77
8	Thứ 6	50	74		40					164
9	Thứ 7				33					33
10	Tổng cộng	150	339		224					713
11	**Thời gian và tốc độ giai đoạn**					**Số tuần trước +1)**		**(số**	**2**	
12	**Thời gian của tuần trước**									
13	Tổng cộng	150	341	134	146					771
14	Trung bình	150	341	134	146					771
15	Max	150	341	134	146					771
16	Min	150	341	134	146					771
17	**Thời gian của tuần hiện tại**									
18	Tổng cộng	300	680	134	370					1484
19	Trung bình	150	340	67	185					742
20	Max	150	341	134	224					771
21	Min	150	339	134	146					713

Bảng 2.3.3 Tốc độ và thời gian giai đoạn, tuần 2

Các lời chỉ dẫn để hoàn tất bản tổng kết hoạt động hàng tuần được liệt kê trong bảng sau:

Mục đích	Biểu mẫu này dành cho việc theo dõi và phân tích thời gian sử dụng. Những số liệu này được tổng kết từ bản ghi ghi chép thời gian.
Tổng quát	Tổng kết lại dữ liệu này từ bản ghi ghi chép thời gian vào cuối mỗi tuần. Nếu loại công việc không thích hợp, hãy thay đổi nó.
Đầu trang	Nhập vào: - Tên bạn. - Ngày hiện tại.
Công việc	Nhập tên các công việc chính mà bạn đã dùng thời gian vào đó cho khóa học này.
Ví dụ	Lớp học, viết chương trình, chuẩn bị vấn đáp, đọc tài liệu, v.v...
Ngày	Cạnh chủ nhật, nhập vào ngày tháng
Ví dụ	8/9
Các cột	Cho mỗi ngày trong tuần, tìm tổng thời gian trải qua cho mỗi loại công việc từ bản ghi ghi chép thời gian. Nhập vào con số này vào cột thích hợp cho ngày đó.
Ví dụ	Với thứ 2, 9/9, thời gian viết chương trình trên bản ghi ghi chép thời gian là 38 và 58 phút. Nhập tổng cộng, hay 98 phút dưới cột viết chương trình ở dòng 4 cho thứ 2 ngày 9/9.
Tổng trong tuần (dòng 10)	Tổng các con số cho mỗi công việc cho toàn bộ tuần và nhập chúng vào dòng 10
Ví dụ	96 + 62 + 69 + 114 = 341 cho dòng 10 dưới cột viết chương trình
Các tổng	Với mỗi dòng, tính tổng thời gian công việc để lấy được tổng hàng ngày trong cột phải nhất. Nếu bạn sử dụng một vài biểu mẫu, đặt số tổng cộng cho tất cả các cột từ tất cả các biểu mẫu trong cột phải nhất của biểu mẫu đầu tiên.
Ví dụ	Tổng là 50 + 96 + 80 = 226 cho thứ 2 ngày 9/9.
Kiểm tra lần đầu	Tính tổng các số tổng cộng để biết được tổng thời gian trôi qua trong tuần và nhập nó vào cột và dòng tổng cộng cho tuần đó.
Ví dụ	226 + 62 + 147 + 114 + 88 + 134 = 771 là số tổng cộng cho tuần.
Kiểm tra lần cuối	150 + 341 + 134 + 146 = 771 cho tổng các công việc ở dòng 10. Bởi vì con số này bằng với số tổng cộng của cột bên phải nên không có sai sót.
Số tuần (dòng 11)	Đây là số tuần trong dữ liệu bản tổng kết.
Thời gian đến ngày (dòng 13-16)	Với dòng 13-16, sao chép lại theo từng cột dữ liệu từ dòng 18-21 của bản tổng kết tuần trước. Bao gồm dòng các số tổng cộng, trung bình, giá trị lớn nhất, giá trị nhỏ nhất cho mỗi cột, bao gồm cột tổng cộng.
Tổng cộng thời	Để tính giá trị cho dòng 18, cộng các số trong dòng 10 và 13.

gian hiện tại (dòng 18)	Thực hiện với mọi cột của bảng này.
Ví dụ	Trong bảng 2.3.3, giá trị cột viết chương trình cho dòng 18 được tính: 339 +341 = 680.
Trung bình thời gian hiện tại (dòng 19)	Giá trị trung bình được tính bằng cách chia mục nhập dòng 18 trong mỗi cột cho số tuần ở dòng 11.
Ví dụ	Trong bảng 2.3.3, mục nhập cho trung bình viết chương trình là 680/2 = 340
Thời gian hiện tại lớn nhất (dòng 20)	Giá trị dòng 20 được lấy bằng cách so sánh dòng 15 và dòng 10. Nhập số lớn hơn giữa 2 số. Làm tương tự cho mọi cột, bao gồm cả cột tổng số.
Ví dụ	Với viết chương trình trong bảng 2.3.3, GTLN là 341 và giá trị cho tuần này là 339 vì vậy GTLN mới vẫn là 341.
Ví dụ khác	Với công việc đọc tài liệu trong bảng 2.3.3, GTLN là 146 và giá trị của tuần này là 224 vì vậy GTLN mới là 224.
Thời gian hiện tại nhỏ nhất (dòng 21)	Giá trị dòng 21 được lấy bằng cách so sánh dòng 16 và 10, lấy số nhỏ hơn khác 0. Làm tương tự như vậy cho mỗi cột, bao gồm cột tổng số.
Ví dụ	Với cột viết chương trình trong bảng 2.3.3, GTNN là 341 và giá trị dòng 10 là 339 do đó GTNN mới là 339.
Ví dụ khác	Với cột đọc tài liệu trong bảng 2.3.3, GTNN là 146 và giá trị của dòng 10 là 224 vì vậy GTNN mới vẫn là 146.
Ví dụ khác nữa	Với cột vấn đáp trong bảng 2.3.3, GTNN là 134 và giá trị của dòng 10 là 0, vì vậy GTNN mới vẫn là 134.

Bảng 2.3.4 Các chỉ dẫn tổng kết hoạt động hàng tuần

2.3.4 Sử dụng bản tổng kết hoạt động hàng tuần

Bằng cách hoàn tất biểu mẫu này mỗi tuần, bạn sẽ có một bản đối chiếu về thời gian trung bình, lớn nhất, nhỏ nhất bạn đã trải qua trong mỗi hoạt động hàng tuần. Biểu mẫu này có thể hơi phức tạp một chút lúc đầu, nhưng sau một vài luyện tập nhỏ nó sẽ trở nên dễ làm và không rắc rối nữa. Khi đó bạn sẽ thấy đây là cách đơn giản để tổ chức và giữ lại một lượng lớn dữ liệu thời gian. Nếu bạn sử dụng trang bảng tính để làm những tính toán này, nó sẽ càng dễ hơn nữa.

Dữ liệu trong bảng tổng kết hoạt động hàng tuần sẽ giúp bạn biết được bạn sử dụng thời gian vào đâu. Ví dụ với thông tin này, bạn có thể đoán được một công việc lớn sẽ chiếm gần như hầu hết thời gian lớn nhất và công việc đơn giản gần với thời gian nhỏ nhất.

Khi đó bạn có thể sử dụng những dữ liệu này để lập kế hoạch cho các tuần tiếp theo. Cách lập kế hoạch sẽ được đề cập trong những chương tiếp theo.

Ban đầu bạn học được rất nhiều từ bản tổng kết hoạt động hàng tuần, tuy nhiên sau một vài tuần bạn sẽ thu được ít thông tin mới. Khi bạn đạt đến điểm này, hãy chuẩn bị một bản tổng kết hoạt động hàng tuần 1 lần/tháng hoặc hơn để kiểm tra việc phân phối thời gian hiện tại có thích hợp. Hơn nữa, một khi bạn bắt đầu tập hợp dữ liệu đề án ở các phần sau, bạn sẽ biết bạn dùng bao nhiêu thời gian vào đề án và có thể tổng kết dễ dàng những dữ liệu này khi bạn cần.

2.4 Lập kế hoạch sản phẩm

Phần này mô tả cách sử dụng dữ liệu bản ghi thời gian như thế nào để lập kế hoạch sản phẩm. Một biểu mẫu mới, bản ghi số công việc, được giới thiệu để giúp bạn theo dõi dữ liệu cũ và đưa ra một ví dụ chỉ cho bạn cách hoàn tất biểu mẫu.

2.4.1 Nhu cầu về các kế hoạch sản phẩm

Một vài năm trước đây khi làm việc tại IBM, tác giả Humphrey chịu trách nhiệm trong một ban phát triển phần mềm lớn với nhiều đề án. Hầu hết các đề án bị trễ nghiêm trọng và ban quản lý rất lo lắng. Công việc của ông là giải quyết tình trạng lộn xộn đó. Điều đầu tiên ông làm là xem lại các đề án chủ yếu. Thật sự ngạc nhiên, không có đề án nào có kế hoạch được báo cáo bằng tài liệu. Ngay lập tức ông đề nghị các kỹ sư lập kế hoạch sản phẩm cho tất cả các đề án của họ. Họ mất một vài tháng để làm điều đó do chưa từng chuẩn bị cho các kế hoạch đúng đắn trước đây. Họ thậm chí phải mở một lớp đặc biệt về cách lập kế hoạch đề án. Tuy nhiên, sau khi lập kế hoạch, họ đã có thể thiết lập được lịch biểu cho công việc của mình.

Việc lập kế hoạch có hiệu quả mạnh mẽ. Nhóm phát triển này trước đây chưa bao giờ bàn giao sản phẩm đúng thời hạn. Tuy nhiên, bắt đầu với các kế hoạch mới, họ đã không trễ 1 ngày trong vòng 2 năm rưỡi. Kể từ đó, họ lập kế hoạch cho công việc của mình. Lập kế hoạch là một phần quan trọng trong nghề kỹ sư phần mềm, và để trở thành một kỹ sư hiệu quả, bạn cần phải biết cách lập kế hoạch. Bí quyết là việc tập luyện, vì vậy để luyện tập ngay, hãy bắt đầu lập kế hoạch từ bây giờ và tiếp tục như vậy cho tất cả các đề án tương lai.

2.4.2 Tại sao các kế hoạch sản phẩm lại có ích

Bạn nên phát triển các kế hoạch sản phẩm cho tất cả đề án hay công việc chính của bạn: viết một chương trình, đọc một tài liệu hay chuẩn bị một báo cáo. Kế hoạch sản phẩm sẽ giúp bạn đoán được công việc sẽ chiếm bao nhiêu thời gian và khi nào bạn sẽ hoàn tất. Các kế hoạch cũng giúp bạn theo dõi tiến trình khi đang làm việc.

Khi các kỹ sư làm việc trong 1 nhóm phát triển, họ cần phải lập kế hoạch cho công việc cá nhân của mình. Lập kế hoạch cung cấp một cơ sở chắc chắn để cam kết ngày hoàn tất, và nó cho phép kỹ sư phối hợp công việc trên các sản phẩm chung. Các kế hoạch công việc cá nhân cho phép họ định ngày cho nhau cho các công việc phụ thuộc và đạt được các sự giao phó này một cách nhất quán.

Các ngành nghề dùng các kế hoạch sản phẩm cho cùng một lý do: để lập kế hoạch và quản lý công việc. Một kế hoạch được lập tốt bao gồm ước lượng chi phí đề án. Cần thiết phải ước lượng để tiến triển các hợp đồng vì khách hàng thường muốn biết trước giá cả. Ước lượng cũng cần thiết khi phát triển sản phẩm. Chi phí đề án là phần chủ yếu của giá cả sản phẩm và phải đủ thấp để cạnh tranh trên thị trường.

Các kỹ sư cũng sử dụng các kế hoạch sản phẩm để biết được tình trạng đề án. Với các kế hoạch chính xác và chi tiết hợp lý họ có thể đoán được đề án đang ở đâu trong kế hoạch. Họ có thể biết được đề án bị trễ và cần giúp đỡ hay cần phải hoãn kế hoạch làm việc. Thậm chí họ có thể đi trước lịch biểu và có thể giúp cho những người cùng nhóm bàn giao sớm. Khi kỹ sư lập kế hoạch, họ tổ chức thời gian tốt hơn và tránh sự khủng hoảng vào phút cuối. Khi đó họ ít gây ra sai sót và thường làm ra những sản phẩm tốt hơn.

Do việc lập kế hoạch quan trọng như vậy nên bạn cần biết cách lập kế hoạch chính xác. Bạn cũng cần biết cách so sánh những kế hoạch này với kết quả thực sự để học cách lập kế hoạch tốt hơn.

2.4.3 Một kế hoạch sản phẩm là gì?

Một kế hoạch sản phẩm đúng đắn bao gồm 3 điều:

- Quy mô và các đặc tính quan trọng của sản phẩm cần sản xuất.
- Ước lượng về thời gian cần thiết cho công việc.
- Dự kiến cho lịch làm việc.

Sản phẩm có thể là một chương trình, một bản thiết kế chương trình hay là một kế hoạch kiểm thử. Vì vậy kế hoạch định ra sản phẩm cần sản xuất bao gồm các ước lượng về

quy mô sản phẩm, số giờ để thực hiện công việc và lịch biểu. Những sản phẩm phức tạp hơn đòi hỏi lập kế hoạch tinh vi hơn và nhiều loại thông tin như là ủy thác trách nhiệm, kế hoạch bố trí nhân sự, các đặc tả của sản phẩm hay qui trình, sự phụ thuộc vào các nhóm khác, các kiểm thử đặc biệt hay các điều khoản chất lượng. Tuy nhiên ở đây, chúng ta sẽ chỉ giải quyết 3 yếu tố kế hoạch cơ bản: ước lượng quy mô, số giờ dự kiến và lịch làm việc.

2.4.4 Cách lập kế hoạch sản phẩm trong tài liệu này

Ta bắt đầu bằng cách lập kế hoạch cho các công việc nhỏ vì đó là cách tốt để học lập kế hoạch. Nếu bạn không thể lập kế hoạch cho một công việc nhỏ thì làm sao bạn có thể lập kế hoạch cho một đề án lớn? Các kỹ sư làm thuê thực hiện nhiều công việc nhỏ là các phần của mỗi công việc lớn. Vì vậy, nếu bạn lập kế hoạch cho mỗi công việc nhỏ này, bạn sẽ có nhiều cơ hội hơn để phát triển kỹ năng lập kế hoạch của mình. Đối với các công việc lớn hơn, bạn có thể kết hợp những kế hoạch nhỏ này thành một kế hoạch lớn cho toàn bộ công việc. Đây là cách hiệu quả nhất để lập kế hoạch chính xác cho các công việc lớn.

2.4.5 Lập kế hoạch các công việc nhỏ

Đưa ra một kế hoạch phù hợp với độ lớn và độ phức tạp của công việc cần hoàn thành là quan trọng. Ví dụ, nếu đưa ra một kế hoạch tinh vi cho một công việc mà chỉ mất 1 hay 2 giờ để thực hiện thì sẽ chẳng có ý nghĩa gì. Ngược lại, một kế hoạch lớn lao hơn sẽ xác đáng đối với một công việc có thể chiếm đến vài tuần.

Kế hoạch sản phẩm cơ bản nhất chỉ bao gồm ước lượng về thời gian cần để thực hiện một công việc. Khi bạn đã có thể ước lượng chính xác thời gian cho 1 công việc, tất cả các vấn đề về lập kế hoạch khác có thể nắm được khá dễ dàng.

Bạn có thể đoán được những công việc tương tự sẽ chiếm bao nhiêu thời gian trong tương lai bằng cách tập hợp dữ liệu về khoảng thời gian mà các công việc khác nhau trong quá khứ chiếm. Cách tiếp cận này có thể áp dụng được vào hầu hết các công việc có liên quan đến sản phẩm nào kéo dài từ một vài giờ đến một vài ngày. Dữ liệu chủ yếu cần là những công việc tương tự trước đây chiếm bao nhiêu thời gian. Ví dụ, nếu bạn lập kế hoạch đọc một chương sách, sẽ rất hữu ích để biết mất bao nhiêu thời gian để đọc các chương trước đó. Với dữ liệu thời gian đọc trung bình, lớn nhất và nhỏ nhất, bạn có thể dự đoán tốt hơn thời gian đọc 1 chương mới.

2.4.6 Bản ghi số công việc

Tên: ______________________________ Ngày: ______________________________

Công việc #	Ngày	Tiến trình	Ước lượng		Thực tế			Đến ngày				
			Thời gian	Đơn vị	Thời gian	Đơn vị	Tốc độ	Thời gian	Đơn vị	Tốc độ	GTLN	GTNN
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											

Bảng 2.4.1 Bản ghi số công việc

Tên: _____ Ngày: _____

Công việc #	Ngày	Tiến trình	Ước lượng		Thực tế			Đến ngày				
			Thời gian	Đơn vị	Thời gian	Đơn vị	Tốc độ	Thời gian	Đơn vị	Tốc độ	GTLN	GTNN
1	9/9	Chương trình	100	1	158	1	158	158	1	158	158	158
	Mô tả	Viết chương trình 1 (phút / chương trình)										
2	9/9	Tài liệu	50	2	80	2	40	80	2	40	40	40
	Mô tả	Đọc tài liệu chương 1 & 2 (phút / chương)										
3	11/9	Chương trình	158	1	69	1	69	227	2	113.5	158	69
	Mô tả	Viết chương trình 2										
4	11/9	Tài liệu	40	1	28	1	28	108	3	36	40	28
	Mô tả	Đọc tài liệu chương 3										
5	12/9	Chương trình	114	1	114	1	114	341	3	113.7	128	69
	Mô tả	Viết chương trình 3										
6	13/9	Tài liệu	60	1	118	1	118	226	4	56.5	118	28
	Mô tả	Đọc tài liệu chương 4										
7	16/9	Chương trình	114	1	93	1	93	434	4	108.5	158	69
	Mô tả	Viết chương trình 4										
8	17/9	Chương trình	109	1	95	1	95	529	5	105.8	158	69
	Mô tả	Viết chương trình 5										
9	18/9	Tài liệu	57	1	71	1	71	297	5	59.4	118	28
	Mô tả	Đọc tài liệu chương 5										
10	19/9	Chương trình	106	1	151	1	151	680	6	113.3	158	69
	Mô tả	Viết chương trình 6										
11	20/9	Tài liệu	59	1	40	1	40	337	6	26.2	118	28
	Mô tả	Đọc tài liệu chương 6										
12	21/9	Tài liệu	56	1								
	Mô tả	Đọc tài liệu chương 7										
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											
	Mô tả											

Bảng 2.4.2 Một ví dụ bản ghi số công việc

Student Sinh viên Y Ngày 9/9/96

Instructor Ông Z Lớp CS1

Ngày	Bắt đầu	Kết thúc	Thời gian gián đoạn	Thời gian Delta	Công việc #	Lời chú giải	C	U
9/9	9:00	9:50		50	Lớp học	Bài giảng		
	12:40	1:18		38	1	Bài tập 1		
	2:45	3:53	10	58	1	Bài tập 1		
	6:25	7:45		80	2	Đọc sách – Ch 1&2	X	2
10/9	11:06	12:19	6+5	62	1	Bài tập 1, nghỉ giải lao, tán gẫu	X	1
11/9	9:00	9:50		50	Lớp học	Bài giảng		
	1:15	2:35	3+8	69	3	Bài tập 2, nghỉ giải lao, điện thoại	X	1
	4:18	5:11	25	28	4	Tài liệu Ch3, tán gẫu với Mary	X	1
12/9	6:42	9:04	10+6+2	114	5	Bài tập 3	X	1
13/9	9:00	9:50		50	Lớp học	Bài giảng		
	12:38	1:16		38	6	Tài liệu Ch4		
14/9	9:15	11:59	5+3+22	134	Xem lại	Chuẩn bị kiểm tra vấn đáp, nghỉ giải lao, điện thoại, tán gẫu		
16/9	9:00	9:50		50	Lớp học	Bài giảng		
	2:10	4:06	4+19	93	7	Bài tập 4, nghỉ giải lao, điện thoại	X	1
	7:18	8:49	11	80	6	Đọc tài liệu – Ch4, tán gẫu	X	1
17/9	9:26	11:27	4+22	95	8	Bài tập 5, nghỉ giải lao, điện thoại	X	1
18/9	9:00	9:50		50	Lớp học	Bài giảng		
	4:21	5:43	11	71	9	Tài liệu Ch5, nghỉ	X	1
19/9	6:51	9:21	51+16+6	77	10	Bài tập 6		
20/9	9:00	9:50		50	Lớp học	Bài giảng		
	12:33	1:18	5	40	11	Tài liệu Ch6, nghỉ	X	1
	1:24	2:38		74	10	Bài tập 6	X	1
21/9	11:18	11:51		33	12	Tài liệu Ch7		

Bảng 2.4.3 Một ví dụ bản ghi thời gian

Tổng quát	Khi bắt đầu đề án, ghi số công việc mới trong bản ghi này. Ấn định số thứ tự bắt đầu bằng 1.
Đầu trang	Ghi tên bạn vào. Ghi vào ngày mà trang bản ghi số công việc này được bắt đầu.
Công việc #	Ghi vào số thứ tự công việc bạn đã chọn.
Ngày	Ghi vào ngày bạn bắt đầu công việc.
Tiến trình	Nhập loại công việc. Ví dụ, với bài kỹ thuật sử dụng giấy để phát triển một chương trình, sử dụng Chương trình (Prog.), v.v.…
Thời gian ước lượng	Nhập tổng thời gian bằng phút cho công việc được ước lượng. Sử dụng Tốc độ Đến Ngày, GTLN, GTNN như đã hướng dẫn. Nếu những tốc độ này có vẻ bất hợp lý, hãy sử dụng óc phán đoán của bạn.
Đơn vị ước lượng	Nhập vào đơn vị ước lượng cho công việc đã hoàn tất. Ví dụ, với việc phát triển một chương trình, ước lượng số dòng lệnh bạn nghĩ chương trình đã hoàn thành sẽ chiếm
Thời gian thực tế	Nhập vào tổng thời gian thực tế cuối cùng mà công việc chiếm.
Đơn vị thực tế	Nhập vào số thực tế cuối cùng của tổng đơn vị. Ví dụ, với một chương trình, bạn đếm số dòng lệnh trong chương trình đã hoàn thành.
Tốc độ thực tế	Nhập vào thời gian thực chia cho đơn vị thực tế.
Thời gian Đến Ngày	Tìm công việc loại này đã hoàn thành gần đây nhất. Cộng Thời gian Đến Ngày từ công việc đó với thời gian thực tế cho công việc gần đây nhất này. Nhập tổng này vào khoảng trống Thời gian Đến Ngày cho công việc mới.
Đơn vị Đến Ngày	Tìm công việc loại này đã hoàn thành gần đây nhất. Cộng Đơn vị Đến Ngày từ công việc đó với đơn vị thực tế cho công việc gần đây nhất này.
Tốc độ Đến Ngày	Chia Thời gian Đến Ngày cho Đơn vị Đến Ngày để ra được số phút/đơn vị cho tất cả các công việc đã hoàn thành đến ngày…
GTLN	Nhập tốc độ lớn nhất cho tất cả các công việc đã hoàn thành của mỗi loại.
GTNN	Nhập tốc độ nhỏ cho tất cả các công việc đã hoàn thành của mỗi loại.
Mô tả	Nhập vào mô tả cho công việc được thực hiện. Phải đủ rõ ràng để nội dung của công việc có thể dễ dàng được nhận ra. Khi dữ liệu thời gian đầu tiên về một loại công việc được ghi lại, mô tả đơn vị của phép đo.

Bảng 2.4.4 Các chỉ dẫn bản ghi số công việc

Bản ghi số công việc được thiết kế để ghi lại dữ liệu thời gian thực tế và ước lượng. Lưu ý rằng bản ghi này là một tài liệu lập kế hoạch sản phẩm vì nó làm việc với dữ liệu sản

phẩm. Ngược lại bản ghi chép thời gian và bản tổng kết hoạt động hàng tuần bao gồm dữ liệu về các giai đoạn hàng tuần. Vì vậy chúng là các tài liệu về lập kế hoạch giai đoạn.

Mục nhập trong bản ghi số công việc được hoàn tất trong bảng 2.4.2. Ví dụ này sử dụng dữ liệu từ bản ghi chép thời gian của sinh viên Y trong bảng 2.4.3. Các chỉ dẫn về bản ghi số công việc cũng được tóm tắt trong bảng 2.4.4. Chú ý rằng trong bản ghi, sinh viên Y theo dõi 2 hoạt động: lên lớp học và ôn thi. Cô cũng theo dõi 2 hoạt động gần giống dự án là viết chương trình và đọc các chương tài liệu. Các mục nhập trong bảng 2.4.2 được hoàn tất như sau:

1. Công việc #	Khi lập kế hoạch một hoạt động, chỉ định 1 số thứ tự công việc, bắt đầu bằng 1.
2. Ngày	Nhập ngày bạn bắt đầu thực hiện công việc.
3. Tiến trình	Nhập vào loại công việc, như là đọc tài liệu, viết 1 chương trình hay chuẩn bị 1 báo cáo.
4. Thời gian Ước lượng	Ước lượng thời gian mà công việc này sẽ chiếm và điền vào dưới Thời gian Ước lượng. Trong ước lượng này, khảo sát dữ liệu trong các đề án tương tự trước đây và sử dụng những dữ liệu này trong việc ước lượng.
5. Đơn vị Ước lượng	Với đơn vị đơn của công việc, nhập vào 1 dưới Đơn vị Ước lượng.
6. Thời gian Thực tế	Cuối công việc, nhập vào thời gian mà công việc chiếm. Với công việc số 10, thời gian này là 151 phút.
7. Đơn vị Thực tế	Ngoài ra, khi thực hiện xong, ghi lại các đơn vị thực tế. Ở đây, sẽ là 1 cho mỗi công việc.
8. Tốc độ Thực tế	Tốc độ Thực tế = Thời gian Thực tế / Đơn vị Thực tế. Với chương trình số 6 (công việc số 10), Tốc độ Thực tế là 151/1=151 phút / chương trình
9. Thời gian Đến Ngày	Vào cuối công việc, tính toán và nhập vào Thời gian Đến Ngày cho tất cả các công việc đã xong đến ngày này của cùng một loại tiến trình. Ví dụ, với Thời gian Thực tế là 151 phút cho chương trình 6 và Thời gian Đến Ngày cho chương trình 1 đến 5 là 529 (xem công việc 8), Thời gian Đến Ngày cho chương trình 6 là 151 + 529 = 680 phút.
10. Đơn vị Đến Ngày	Nhập Đơn vị Đến Ngày cho tất cả các công việc đã hoàn tất của mỗi loại. Ngoài ra, với chương trình 6, cộng Đơn vị Thực tế cho công việc này với Đơn vị Đến Ngày cho công việc gần nhất trước đó của loại này (công việc số 8): 5+1=6.
11. Tốc độ Đến Ngày	Tốc độ Đến Ngày = Thời gian Đến Ngày / Đơn vị Đến Ngày, hay 680/6=113.3 phút/chương trình. Đây là thời gian trung bình bạn thực hiện công việc loại này.
12. Giá trị Đến Ngày Lớn nhất	Để tìm được tốc độ lớn nhất cho bất cứ công việc nào của loại này cho đến nay, so sánh với Tốc độ Thực tế của công việc gần nhất với Giá trị Đến Ngày Lớn nhất của công việc trước đó của cùng loại và nhập vào số lớn hơn. Đối với chương

	trình 6, Giá trị Đến Ngày Lớn của chương trình 5 (công việc 8) là 158 phút/đơn vị, lớn hơn Tốc độ Thực tế là 151 phút /đơn vị của chương trình 6, vì vậy nhập vào 158 cho Giá trị Đến Ngày Lớn Nhất của chương trình 6.
13. Giá trị Đến Ngày Nhỏ nhất	Giá trị Đến Ngày Nhỏ nhất là tốc độ nhỏ nhất cho bất cứ công việc nào của loại này tính đến bây giờ. Tìm giá trị này bằng cách so sánh Tốc độ Thực tế của công việc hiện hành với Giá trị Đến Ngày Nhỏ nhất của công việc gần nhất trước đó của loại này và nhập vào số nhỏ hơn. Ví dụ, Tốc độ Thực tế là 151 phút/đơn vị lớn hơn 69 phút/đơn vị của Giá trị Tới Ngày Nhỏ nhất của chương trình 5 (công việc 8), vì vậy Giá trị Tới Ngày Nhỏ nhất của chương trình 6 vẫn là 69.

Với những dữ liệu này, bạn có thể dễ dàng tìm kiếm tốc độ trung bình cho bất cứ loại công việc nào cũng như là tốc độ lớn nhất, nhỏ nhất. Tiện lợi chủ yếu của Bản ghi số công việc là nó cung cấp một phương pháp súc tích để ghi lại và truy cập một lượng lớn dữ liệu đề án lịch sử. Vì vậy, bạn sẽ thấy đây là chìa khóa để ước lượng chính xác. Ước lượng chính xác là chìa khóa để lập kế hoạch tốt.

2.4.7 Một vài lời khuyên về cách sử dụng bản ghi số công việc

Các luyện tập sau sẽ giúp bạn sử dụng hiệu quả bản ghi số công việc:

Đối với các công việc đầu tiên của một loại đã cho, sinh viên Y không có dữ liệu trước đó để hướng dẫn cho ước lượng của mình. Vì vậy cô phải đoán. Việc đoán thời gian đầu tiên được chấp nhận cho đến khi bạn bắt đầu thu thập dữ liệu để không phải tiếp tục đoán nữa.

Thông thường, trong việc ước lượng thời gian cho một công việc mới, bạn sẽ muốn sử dụng *Tốc độ Đến Ngày* cho công việc cùng loại trước đó gần nhất. Sinh viên Y thường cũng làm như vậy. Với công việc 6, sinh viên Y đã không sử dụng *Tốc độ Đến Ngày* là 36 phút/chương. Có lẽ cô nghĩ chương 4 sẽ chiếm nhiều thời gian hơn. Quan trọng là hãy nhớ rằng dữ liệu trong bản ghi số công việc là để giúp lập kế hoạch. Tuy nhiên, hãy nghĩ về các con số, và nếu bạn nghĩ một ước lượng lớn hơn hoặc nhỏ hơn sẽ chính xác hơn, hãy sử dụng óc phán đoán của bạn.

Để nhanh chóng tìm ra tất cả các mục bản ghi chép thời gian cho một số công việc cho trước, thêm vào chỉ số công việc ở cột Hoạt động trong bản ghi chép thời gian, như trong bảng 2.4.3. Trên thực tế, bạn sẽ thấy tiện lợi hơn khi bỏ hết các loại hoạt động và chỉ tham chiếu đến hoạt động bằng mã số công việc của chúng.

Sau khi bạn học cách sử dụng bản ghi số công việc, bạn sẽ thấy tiện lợi hơn khi sử dụng trang bảng tính để thực hiện các tính toán.

2.4.8 Sử dụng dữ liệu tốc độ và thời gian sản phẩm

Giả sử bạn có dữ liệu như trong bảng 2.4.2 và muốn lập kế hoạch thời gian cho tuần sau. Bạn có thể lập kế hoạch để viết hai chương trình và đọc 4 chương tài liệu. Bạn cũng có thể biết rằng một trong hai chương trình thì phức tạp hơn những chương trình bạn đã từng viết và những thứ khác thì dường như là ở mức trung bình.

Để ước lượng thời gian viết hai chương trình này, hãy nhìn vào bảng 4.2.2 của công việc 10. Ở đây, thời gian trung bình để viết sáu chương trình đầu là 113,3 phút, hay ít hơn 2 giờ một chút. Thời gian lớn nhất là 158 phút, hay khoảng 2,5 giờ. Khi đó chương trình trung bình sẽ chiếm khoảng 2 giờ và công việc phức tạp hơn sẽ chiếm ít nhất 158 phút và có thể hơn. Để an toàn, bạn giả sử công việc phức tạp sẽ chiếm 3,5 giờ, hay 210 phút. Tổng cộng là 330 phút dành cho việc lập trình trong tuần tới.

Với các chương tài liệu, bạn nghĩ rằng hai chương trung bình và hai chương kia sẽ chiếm ít thời gian hơn. Từ dữ liệu của công việc 11, thời gian trung bình để đọc các chương là 56,2 phút. Vì vậy, bạn giả sử rằng bạn có thể đọc hai chương trung bình trong khoảng 1 giờ. Tuy nhiên, các chương phức tạp hơn, chắc sẽ gần thời gian lớn nhất là 118 phút, hay khoảng 2 giờ.

Tổng cộng bạn nghĩ là sẽ trải qua khoảng 5,5 giờ để viết chương trình và 6 giờ đọc tài liệu cho tuần tới. Với thông tin này, bạn có thể lập kế hoạch tốt hơn cho công việc.

Bạn sẽ sớm thấy rằng thời gian cho các công việc nhỏ sẽ thay đổi nhiều xung quanh ước lượng của bạn. Đây là điều bình thường. Tuy nhiên, bí quyết ước lượng tốt là học cách ước lượng cân bằng. Một tập hợp cân bằng các ước lượng có nhiều ước lượng thấp cũng như cao. Trong ví dụ của bảng 2.4.2, 5 trong 11 ước lượng thì thấp, 5 cao và công việc 5 là chính xác. Ước lượng thấp là các công việc 1, 2, 6, 9 và 10; các ước lượng khác là cao. Vì vậy sinh viên Y học các để ước lượng cân bằng. Cố thực hiện điều này bằng cách thông thường sử dụng các tốc độ trung bình trước đó khi ước lượng một công việc mới. Miễn là bạn làm điều này thích hợp và miễn là công việc của bạn tương tự một cách hợp lý, bạn sẽ sớm đạt được các ước lượng cân bằng.

Thuận lợi của việc có các ước lượng cân bằng là về trung bình công việc của bạn sẽ kéo dài trong khoảng mà bạn ước lượng. Chính vì vậy, khi một công việc kéo dài hơn, nó sẽ được bù bởi công việc khác chiếm ít thời gian hơn. Khi ước lượng các công việc lớn là tập hợp của nhiều công việc nhỏ, bạn sẽ có thể đến gần với toàn bộ kế hoạch.

2.5 Kích thước sản phẩm

Đến đây chúng ta đã xem xét các đơn vị lớn của công việc như chương trình hay chương sách. Tuy nhiên, để lập kế hoạch sản phẩm, bạn cần sử dụng những độ đo chính xác hơn. Phần này mô tả cách đo và ước lượng kích thước sản phẩm.

2.5.1 Phép đo kích thước

Vì các công việc thường khác nhau đáng kể về kích thước và độ phức tạp nên nếu có một cách so sánh các kích thước với nhau thì sẽ hữu ích. Xét việc đọc các chương trong 1 quyển sách, với dữ liệu để đọc 5 chương sách, bạn có thể ước lượng thời gian để đọc chương sách thứ 6. Một cách là lấy các thời gian trung bình đọc chương trước đó. Cách này chắc chắn tốt hơn là cách không có gì, nhưng nó không phân biệt giữa chương ngắn và chương dài. Có thể đoán chừng được sẽ mất ít thời gian để đọc chương ngắn hơn là một chương dài. Vì vậy bạn nên xem xét việc đo thời gian đọc các chương bằng phút/trang hơn là phút/chương.

Bây giờ hãy xem dữ liệu của sinh viên Y trong bảng 2.5.1 về thời gian để đọc các chương sách. Thời gian để đọc các chương lên xuống từ 28 đến 118 phút. Tỉ lệ thời gian dài nhất với ngắn nhất là hơn 4. Khi đo bằng phút/trang thì dãy thời gian là từ 2,33 phút/trang cho chương 3 và 7,38 phút/trang cho chương 4. Đây vẫn là một giới hạn rộng, nhưng nó chỉ khoảng 3 lần. Trong khi có sự khác nhau đáng kể, có vẻ như là bạn có thể ước lượng thời gian đọc các chương tài liệu tốt hơn nếu bạn ước lượng dựa trên kích thước của chương và dữ liệu lịch sử tốc độ đọc trung bình bằng phút/trang. Để tính toán tốc độ đọc trung bình, cộng tất cả các thời gian đọc và chia cho tổng số trang của chương, vì vậy tốc độ trung bình là 4,38 phút/trang.

Sinh viên	Sinh viên Y	Ngày	30/9/96
Người hướng dẫn	Thầy Z	Lớp	CS1

Chương	Thời gian đọc	Trang	Phút/trang
1&2	50	20	4,00
3	28	12	2,33
4	118	16	7,38
5	71	17	4,18
6	40	12	3,33
Tổng cộng	337	77	
Trung bình	56,17	12,83	4,38

Bảng 2.5.1 Thời gian đọc các chương của sinh viên Y

2.5.2 Một vài chú ý khi sử dụng các độ đo kích thước

Cách sử dụng các độ đo kích thước dường như khá đơn giản, nhưng cũng có vài trường hợp phức tạp. Ví dụ đọc một số tài liệu thì khó hơn nhiều so với các tài liệu khác. Do đó bạn nên xem xét cả loại công việc liên quan chứ không chỉ kích thước của nó.

Ngoài ra, có một câu hỏi nảy sinh. Ví dụ giả sử bạn đang chuẩn bị cho một kỳ thi. Thời gian để đọc lại các chương trước đây sẽ ít hơn thời gian đọc ban đầu. Tương tự như vậy, ngay cả với việc đọc lại một chương tài liệu mà bạn đã viết, thời gian đọc có thể khác nhau rất nhiều, phụ thuộc vào nội dung. Với soạn từ đầu một tài liệu, bạn có thể mất 15 hay 20 phút/trang. Tuy nhiên, đọc lướt qua một tài liệu đã hoàn tất có thể chỉ chiếm 1 phút/trang hay ít hơn.

Các vấn đề tương tự nảy sinh trong khi lên kế hoạch phát triển chương trình. Năng suất cho các loại công việc khác nhau, như tái sử dụng các chương trình đã phát triển trước đây, chỉnh sửa một chương trình đã có, hay phát triển các chương trình mới, sẽ hoàn toàn khác nhau. Để giải quyết vấn đề này, bạn nên giữ lại các kích thước và hồ sơ thời gian riêng biệt cho các công việc khác nhau mà bạn thực hiện.

2.5.3 Kích thước chương trình

Khi ước lượng thời gian cần thiết để viết một chương trình, hãy dựa vào thời gian trước đây cần có để viết một chương trình tương tự. Như trong bảng 2.5.2, thời gian của sinh viên Y viết chương trình nằm trong khoảng từ 69 phút cho đến 158 phút. Đây chỉ là một khoảng giới hạn của hai lần viết chương trình, nó sẽ mở rộng ra trong tương lai khi sinh viên Y viết các chương trình lớn hơn. Một lần nữa, đây là ý hay cho việc dựa trên ước lượng kích thước chương trình.

Sinh viên	Sinh viên Y	Ngày	30/9/96
Người hướng dẫn	Thầy Z	Lớp	CS1

Chương trình	Thời gian phát triển	LOC	Phút/LOC
1	158	20	7,90
2	69	11	6,27
3	114	14	8,14
4	93	10	9,30
5	95	14	6,79
6	151	18	8,39
Tổng cộng	680	87	
Trung bình	110,0	14,5	7,82

Bảng 2.5.2 Thời gian phát triển chương trình của sinh viên Y

Độ đo ta sử dụng cho kích thước chương trình là số dòng văn bản trong chương trình nguồn. Nếu chương trình chiếm 16 dòng in, chương trình có 16 dòng lệnh (LOC). Quy ước không đếm dòng trống hoặc dòng lời chú giải. Dòng có mã lệnh và lời chú giải cũng được tính là 1 dòng lệnh. Dù bạn có thể chọn hầu như bất kỳ chuẩn nhất định nào, trong tài liệu này chúng ta sẽ đếm dòng lệnh mà không đếm dòng trống hay dòng chỉ có lời chú giải. Vì vậy, đoạn chương trình Ada sau có 5 dòng lệnh:

Ví dụ 1

```
-- lời bình giải thích chức năng chương trình
      If (X_Average >=100) then
                  Size := X_Average;
            else
            Size := X_Average/2;
            end if
```

Tương tự, nếu bạn viết đoạn chương trình này mà không có lời chú giải và trong một kiểu gọn hơn, nó sẽ chỉ có 3 dòng lệnh:

Ví dụ 2

```
If (X_Average >=100) then
                  Size := X_Average;
            else Size := X_Average/2; end if
```

Trong khi đây là những chương trình giống hệt nhau và thời gian phát triển chúng sẽ giống nhau, kích thước của chúng khác nhau bởi cách tính này. Chỉ cần bạn nhất quán trong cách viết chương trình, những sự khác nhau này không quan trọng. Tuy nhiên để đảm bảo rằng tổng số kích thước nhất quán, bạn nên sử dụng một định dạng chuẩn để viết chương trình.

Độ đo dòng lệnh thường có thể áp dụng được cho hầu hết các ngôn ngữ lập trình. Ví dụ, đoạn chương trình tương tự trong C++ sẽ trông như sau:

Ví dụ 3

```
       if (X_Average >=100)
            Size := X_Average;
      else
            Size := X_Average/2;
```

Khi được viết bằng cách này, đoạn mã C++ này sẽ có 4 dòng lệnh.

Sử dụng cách đếm dòng lệnh này, kích thước 6 chương trình của sinh viên Y được thể hiện trong bảng 2.5.2. Giới hạn phút/dòng lệnh đi từ 6,27 với chương trình 2 đến 9,3 cho chương trình 4. Vì thời gian bằng phút/dòng lệnh sẽ thay đổi đáng kể với kinh nghiệm, bạn nên theo dõi tốc độ phút/dòng lệnh và ước lượng dựa trên tốc độ đạt được của bạn trong 5 đến 10 chương trình gần đây nhất của bạn.

2.5.4 Các độ đo kích thước khác

Mặc dù chúng ta không sử dụng chúng ở đây, có rất nhiều các độ đo kích thước phần mềm có tiềm năng khác. Ví dụ, phát triển phần mềm công nghiệp thường liên quan đến báo cáo tài liệu mà có thể đo bằng trang. Ngay cả cho chương trình, độ đo dòng lệnh cũng không bao gồm hết tất cả các trường hợp. Các ví dụ của loại sản phẩm mà số dòng lệnh thường không thích hợp là thực đơn, tập tin, các trang báo cáo hay màn hình. Trừ khi bạn nghĩ ra một độ đo kích thước phù hợp, bạn nên sử dụng các tốc độ và đếm đơn vị như đã làm trong ở phần trước.

Ngoài ra, nếu bạn đang sử dụng một công cụ trợ giúp phát triển chương trình phát sinh nhiều loại màn hình, cửa sổ hay các yếu tố chương trình chuẩn khác, việc đếm có thể đòi hỏi một chút tinh vi hơn. Ở đây, điều quan trọng là chỉ đếm số dòng lệnh bạn phát triển và không phải là dòng lệnh phát sinh bởi các công cụ trợ giúp lập trình.

Bất chấp các độ đo được sử dụng, mục tiêu chính là để ước lượng công việc phát triển. Đối với việc này, bạn muốn các độ đo kích thước liên quan đến công việc đòi hỏi để phát triển sản phẩm. Chính vì vậy đối với việc phát triển sản phẩm tốn nhiều thời gian, ta nên sử dụng các độ đo kích thước lớn hơn tương ứng.

2.5.5 Ước lượng kích thước chương trình

Dường như bây giờ bạn có thể ước lượng thời gian để viết chương trình, nhưng nó không đơn giản như vậy. Với việc đọc các chương tài liệu, bạn có thể đếm số trang cần đọc và sử dụng dữ liệu trung bình cũ số phút/trang để tính toán thời gian đọc. Ngay cả việc bạn có thể sắp xếp các chương bởi độ khó và cho phép một cái gì đó lớn hơn tốc độ phút/trang cho các chương mà dường như phức tạp hơn hay gồm tài liệu ít liên quan. Bằng cách liếc sơ chương mới, bạn đánh giá độ khó tương ứng và lấy tốc độ phút/trang dựa trên kinh nghiệm quá khứ.

Tuy nhiên đối với việc viết chương trình, không có chương trình để đếm đến khi bạn phát triển xong nó. Vì vậy, để ước lượng thời gian viết chương trình, đầu tiên bạn phải ước lượng chương trình cần bao nhiêu dòng lệnh và sau đó ước lượng số phút/dòng lệnh. Sau đó bạn có thể tính tổng thời gian ước lượng.

Trong khi có nhiều cách để ước lượng kích thước chương trình trước khi phát triển chúng, tất cả các phương pháp ước lượng kích thước liên quan đến rất nhiều sự phán đoán. Đầu tiên, xem xét các yêu cầu của chương trình cần phát triển. Sau đó, xếp hạng kích

thước tương đối của chương trình mới so với các chương trình mà bạn đã viết. Cuối cùng, dựa vào ý kiến của bạn về việc chương trình mới sẽ rơi vào điểm nào trong dãy kích thước lịch sử này, ước lượng nó giống như số dòng lệnh.

Một ví dụ của thủ tục này được thể hiện trong bảng 2.5.3. Đây là danh sách các chương trình phát triển trước đây của sinh viên Y, thời gian phát triển tính bằng phút, sắp xếp theo thứ tự kích thước. Danh sách chỉ ra kích thước chương trình tính bằng dòng lệnh, thời gian phát triển bằng phút, tốc độ bằng phút/dòng lệnh và mô tả ngắn gọn các chức năng của chương trình. Bằng việc xem xét các dữ liệu này trong chương trình của bạn, và cân nhắc những gì bạn biết gì về chương trình định viết, bạn có thể đoán được chương trình mới sẽ rơi vào chỗ nào trong bảng giới hạn kích thước. Điều này giúp bạn ước lượng giới hạn kích thước của chương trình mới. Dựa vào dữ liệu lịch sử phút/dòng lệnh, bạn có thể ước lượng thời gian để phát triển chương trình mới.

Sinh viên	Sinh viên Y	Ngày	30/9/96
Người hướng dẫn	Thầy Z	Lớp	CS1

Chương trình	Thời gian	Số dòng lệnh	Phút/dòng lệnh	Chức năng
4	93	10	9,30	Dòng lặp while đơn giản
2	69	11	6,27	Câu lệnh case nhỏ
3	114	14	8,14	Câu lệnh case lớn
5	95	14	6,79	Repeat-until trung bình
6	151	18	8,39	Danh sách liên kết nhỏ
1	158	20	7,90	Tính toán nhỏ

Bảng 2.5.3 Dãy các kích thước chương của sinh viên Y

Ví dụ, giả sử sinh viên Y lên kế hoạch viết một chương trình mới bao gồm một dòng lặp while phức tạp vừa phải. Từ bảng 2.5.3, cô sẽ ước lượng nó lớn hơn 14 dòng lệnh cho chương trình 5 và có thể nhỏ hơn 20 dòng lệnh cho chương trình 1. Vì vậy cô có thể đặt giá trị là trung bình của chúng là (14 + 20)/2 = 17 dòng lệnh.

2.5.6 Ước lượng một kích thước lớn hơn

Trong khi cách tiếp cận trong bảng 2.5.3 làm tốt cho các chương trình nhỏ của loại mà bạn đã viết trước đây, nó lại không hoạt động tốt đối với chương trình lớn hơn hay các chương trình mới. Lý do là ngay cả những chương trình khá nhỏ cũng thường chứa một tập hợp các hàm và thủ tục. Vì vậy với các chương trình lớn hơn, bạn sẽ càng gặp nhiều rắc rối khi liên hệ chương trình mới với các chương trình đã phát triển trước đây.

Một cách tiếp cận cho vấn đề này là sử dụng biểu mẫu như trong bảng 2.5.4. Ở đây, các chương trình hay hàm và thủ tục khác nhau được liệt kê ra. Với dữ liệu của một số các chương trình, bạn có thể chia một danh sách thành các loại như sinh viên Y làm trong bảng 2.5.5. Mục đích là để xây dựng một hồ sơ lịch sử các thành phần được viết trước đây với các dữ liệu là số dòng lệnh mà chúng chứa đựng. Khi xem xét các hàm trong một chương trình mới, bạn có thể ước lượng kích thước của mỗi hàm và cộng tất cả những ước lượng hàm này để lấy được ước lượng tổng cộng cho chương trình.

Với các chương trình nhỏ, liệt kê mỗi chương trình như bảng 2.5.5. Khi chương trình trở nên lớn hơn, chúng sẽ bao gồm nhiều chức năng và hàm. Bằng cách liệt kê mỗi hàm và thủ tục riêng rẽ nhau, bạn có thể nhanh chóng xây dựng được một cơ sở cho dữ liệu ước lượng lịch sử. Với nhiều dữ liệu hơn, để giữ một biểu mẫu riêng rẽ giống bảng 2.5.4 cho mỗi loại chương trình sẽ rất có ích. Bạn có thể có một biểu mẫu cho tính toán và một cái khác cho các hàm quản lý văn bản, một cho điều khiển, và v..v… Trên mỗi biểu mẫu, liệt kê các hàm chương trình theo trật tự kích thước và bao gồm tên hoặc số của chương trình chứa đựng nó.

Sinh viên _____ Ngày _____

Người hướng dẫn _____ Lớp _____

Chương trình	**Số dòng lệnh**	**Các chức năng trước đây**	**Hàm ước lượng**	**Nhỏ nhất**	**Trung bình**	**Lớn nhất**
Ước lượng						

Lời chú giải:

Bảng 2.5.4 Biểu mẫu ước lượng kích thước chương trình

Sinh viên Sinh viên Y Ngày 30/9/96
Người hướng dẫn Ông Z Lớp CS1

Chương trình	Số dòng lệnh	Các chức năng trước đây	Hàm ước lượng	Nhỏ nhất	Trung bình	Lớn nhất
Vòng lặp						
4	10	Vòng lặp while đơn giản				
5	14	Repeat-until trung bình	Repeat-until	7	11	4
Case						
2	11	Câu trường hợp nhỏ	trường hợp	5	8	11
3	14	Câu trường hợp lớn hơn				
Dữ liệu						
6	18	Danh sách liên kết nhỏ				
Tính toán						
1	20	Tính toán nhỏ	Tính toán	10	15	20
Ước lượng				22	34	45

Lời chú giải: Chương trình này có một câu trường hợp đơn giản, 1 vòng lặp, và 1 tính toán. Giả sử rằng, tối đa nó sẽ chiếm tổng các kích thước tiêu biểu này, hay 11+14+20=45 dòng lệnh

Tối thiểu, giả sử rằng những hàm này có thể được phối hợp hiệu quả hơn khi chúng được viết riêng rẽ. Vì vậy, lấy 22 dòng lệnh là tối thiểu. 34 dòng lệnh là điểm giữa của chúng.

Bảng 2.5.5 Các ước lượng kích thước chương trình của sinh viên Y

Một ví dụ về cách ước lượng theo cách này được thể hiện trong bảng 2.5.5. Ở đây, sinh viên Y đã nhóm dữ liệu trong một số chương trình thành các loại khác nhau. Tuy nhiên vì các chương trình này khá nhỏ, cô chỉ liệt kê các chương trình hoàn chỉnh. Để ước

lượng số dòng lệnh của một chương trình mới, cô phán đoán chương trình cần bao nhiêu dòng lệnh cho mỗi loại hàm.

Sinh viên Y làm điều này bằng cách ban đầu kiểm tra các yêu cầu cho chương trình mới và thiết lập một chiến lược chung về cách xây dựng nó. Cô nghĩ rằng sẽ sử dụng một vòng lặp repeat-until, một câu case đơn giản và một tính toán khá đơn giản. Tuy nhiên, trong việc ước lượng này, cô không hoàn toàn chắc chắn mỗi phần của chương trình sẽ lớn bao nhiêu, vì vậy cô ước lượng một kích thước nhỏ nhất, kích thước lớn nhất và sau đó là kích thước trung bình cho mỗi chức năng. Tuy cô hay sử dụng các kích thước trung bình trong ước lượng của mình, luyện tập về cách nghĩ về kích thước nhỏ nhất và lớn nhất sẽ giúp tránh đưa ra các ước lượng quá nhỏ hoặc quá lớn.

Tuy đây không phải là một ước lượng kích thước hết sức rõ ràng nhưng vẫn tốt hơn là đoán. Khi bạn viết các chương trình bạn cũng có thể thêm danh sách này vào các tiến trình. Việc lớn lên của dữ liệu lịch sử khi đó sẽ giúp bạn đưa ra được những ước lượng phù hợp hơn.

Không có phương pháp nào cam đoan cho việc ước lượng kích thước tốt. Ước lượng kích thước là một kỹ năng. Bí quyết là có một lượng đáng kể các dữ liệu lịch sử, đưa ra nhiều ước lượng kích thước, thường xuyên so sánh các kết quả của bạn với các ước lượng.

2.5.7 Sử dụng các đơn vị đo kích thước trong bản ghi số công việc

Khi bạn sử dụng các đơn vị đo kích thước thay vì đếm đơn vị, bạn nên để các dữ liệu kích thước trong bản ghi chép thời gian và số công việc như trong bảng 2.5.6 và 2.5.7. Những ví dụ này sử dụng dữ liệu kích thước trong bảng 2.5.1 và 2.5.2. Ở đây, thay vì chỉ chú ý rằng một hay hai chương sách hay chương trình đã hoàn tất, nhập kích thước của chúng trong cột đơn vị. Như trong bảng 2.5.6, sinh viên Y hoàn thành một chương trình 20 dòng lệnh vào ngày 10 tháng 9, vì vậy cô nhập 20 vào cột U. Tương tự vào ngày 11 tháng 9, cô đọc chương 3, vì chương 12 trang này chiếm mất 28 phút, cô nhập 12 vào cột đơn vị của bảng 2.5.6 và 28 dưới cột thời gian Delta.

Vì bạn sẽ có dữ liệu kích thước trong bản ghi số công việc, có thể không cần thiết nhập chúng trong bản ghi thời gian nữa. Tuy nhiên nếu bạn thường hoàn tất bản ghi số công việc một hay hai lần một tuần, khi đó bạn sẽ phải tìm kiếm thông tin về dữ liệu kích thước. Tuy nhiên lúc bạn làm công việc, thường không mất thời gian để nhập dữ liệu kích thước vào bản ghi thời gian.

Student	Sinh viên Y	Ngày	9/9/96
Instructor	Ông Z	Lớp	CS1

Ngày	Bắt đầu	Kết thúc	Thời gian gián đoạn	Thời gian Delta	Công việc #	Lời chú giải	C	U
9/9	9:00	9:50		50	Lớp học	Bài giảng		
	12:40	1:18		38	1	Bài tập 1		
	2:45	3:53	10	58	1	Bài tập 1		
	6:25	7:45		80	2	Đọc sách – Ch 1&2	X	20
10/9	11:06	12:19	6+5	62	1	Bài tập 1, nghỉ giải lao, tán gẫu	X	20
11/9	9:00	9:50		50	Lớp học	Bài giảng		
	1:15	2:35	3+8	69	3	Bài tập 2, nghỉ giải lao, điện thoại	X	11
	4:18	5:11	25	28	4	Tài liệu Ch3, tán gẫu với Mary	X	12
12/9	6:42	9:04	10+6+2	114	5	Bài tập 3	X	14
13/9	9:00	9:50		50	Lớp học	Bài giảng		
	12:38	1:16		38	6	Tài liệu Ch4		
14/9	9:15	11:59	5+3+22	134	Xem lại	Chuẩn bị kiểm tra vấn đáp, nghỉ giải lao, điện thoại, tán gẫu		
16/9	9:00	9:50		50	Lớp học	Bài giảng		
	2:10	4:06	4+19	93	7	Bài tập 4, nghỉ giải lao, điện thoại	X	10
	7:18	8:49	11	80	6	Đọc tài liệu – Ch4, tán gẫu	X	16
17/9	9:26	11:27	4+22	95	8	Bài tập 5, nghỉ giải lao, điện thoại	X	14
18/9	9:00	9:50		50	Lớp học	Bài giảng		
	4:21	5:43	11	71	9	Tài liệu Ch5, nghỉ	X	17
19/9	6:51	9:21	51+16+6	77	10	Bài tập 6		
20/9	9:00	9:50		50	Lớp học	Bài giảng		
	12:33	1:18	5	40	11	Tài liệu Ch6, nghỉ	X	12
	1:24	2:38		74	10	Bài tập 6	X	18
21/9	11:18	11:51		33	12	Tài liệu Ch7		

Bảng 2.5.6 Một bản ghi thời gian với dữ liệu kích thước

Trong bảng 2.5.7 sinh viên Y sử dụng các dữ liệu kích thước này để tính toán tốc độ sản phẩm. Cô thực hiện bằng cách sử dụng các chỉ dẫn bản ghi số công việc trong bảng 2.5.8. Ví dụ, với công việc 1 cô không có ước lượng kích thước, vì vậy cô để trống *Đơn vị Ước lượng*. Thời gian thực tế vẫn là 158 phút, nhưng đơn vị thực tế là 20 dòng lệnh. Điều

này có nghĩa *Đơn vị Đến Ngày* cũng là 20 dòng lệnh và *Tốc độ Đến Ngày* là 158/20=7,90 phút/dòng lệnh. Vì đây là chương trình đầu tiên, giá trị lớn nhất và nhỏ nhất vẫn là 7,90.

Tên: Sinh viên Y Ngày: 9/9/96

Công việc #	Ngày	Tiến trình	Ước lượng		Thực tế			Đến ngày				
			Thời gian	Đơn vị	Thời gian	Đơn vị	Tốc độ	Thời gian	Đơn vị	Tốc độ	GTLN	GTNN
1	9/9	Chương trình	100		158	20	7,90	158	20	7,90	7,90	7,90
	Mô tả	Viết chương trình 1 (phút / chương trình)										
2	9/9	Tài liệu	50	20	80	20	4,00	80	20	4,00	4,00	4,00
	Mô tả	Đọc tài liệu chương 1 & 2 (phút / chương)										
3	11/9	Chương trình	158		69	11	6,27	227	31	7,32	7,90	6,27
	Mô tả	Viết chương trình 2										
4	11/9	Tài liệu	40	12	28	12	2,33	108	32	3,38	4,00	2,33
	Mô tả	Đọc tài liệu chương 3										
5	12/9	Chương trình	114		114	14	8,14	341	45	7,58	8,14	6,27
	Mô tả	Viết chương trình 3										
6	13/9	Tài liệu	60	16	118	16	7,38	226	48	4,71	7,38	2,33
	Mô tả	Đọc tài liệu chương 4										
7	16/9	Chương trình	114		93	10	9,30	434	55	7,89	9,30	6,27
	Mô tả	Viết chương trình 4										
8	17/9	Chương trình	109		95	14	6,79	529	69	7,67	9,30	6,27
	Mô tả	Viết chương trình 5										
9	18/9	Tài liệu	57	17	71	17	4,18	297	65	4,57	7,38	2,33
	Mô tả	Đọc tài liệu chương 5										
10	19/9	Chương trình	106		151	18	8,39	680	87	7,82	9,30	6,27
	Mô tả	Viết chương trình 6										
11	20/9	Tài liệu	59	12	40	12	3,33	337	77	4,38	7,38	2,33
	Mô tả	Đọc tài liệu chương 6										
12	21/9	Tài liệu	56									
	Mô tả	Đọc tài liệu chương 7										

Bảng 2.5.7 Một bản ghi số công việc với dữ liệu kích thước

Với chương trình thứ 2, công việc 3, sinh viên Y không ước lượng kích thước, vì vậy Đơn vị Ước lượng được để trống. Thời gian thực tế cho chương trình 2 là 69 phút và đơn vị thực tế cho chương trình 2 là 11 dòng lệnh. Bây giờ Thời gian Đến Ngày là 227 phút như trước nhưng Đơn vị Thời gian Đến Ngày là 20+11=31 dòng lệnh. Sinh viên Y thu được con số này bằng cách nhìn vào Đơn vị Đến Ngày cho công việc loại này (công việc 1) gần nhất trước đó và cộng 20 dòng lệnh với 11 dòng lệnh Đơn vị thực tế cho công việc

này. Bây giờ với 31 dòng lệnh *Đến Ngày* và 227 *Thời gian Đến Ngày*, *Tốc độ Đến Ngày* là 227/31 = 7,32 phút/dòng lệnh. Tốc độ tối đa cho công việc 1 là 7,90, lớn hơn 6,27 phút/dòng lênh của công việc 3, vì vậy tốc độ tối đa vẫn là 7,90. Tốc độ tối thiểu cho công việc 1 cũng là 7,90, nhưng 6,27 phút/dòng lệnh của công việc 3 thì nhỏ hơn 7,90 nên GTNN được thay thế là 6,27.

Tiến hành một chuỗi công việc tương tự cho mỗi chương trình tiếp theo cho đến khi bạn nhập tất cả các dữ liệu số công việc cho các chương trình được viết đến nay. Trong tương lai, bạn hãy ước lượng và nhập số dòng lệnh cho mỗi chương trình mới trước khi bạn viết nó. Tuy điều này tốn công một chút, thủ tục được mô tả trong bảng 2.5.6 và 2.5.7 thì khá dễ để làm theo miễn là bạn có dữ liệu lịch sử. Với việc đọc tài liệu, thủ tục thì giống nhau ngoại trừ bạn lấy Đơn vị Ước lượng bằng cách đếm số trang của tài liệu định đọc.

Sử dụng những dữ liệu này, bạn có thể biết được cần bao nhiêu thời gian để viết chương trình và để đọc các chương tài liệu, có thể gồm cả các công việc khác mà bạn có đơn vị đo kích thước và thời gian.

2.6 Quản lý thời gian

2.6.1 Các yếu tố trong quản lý thời gian

Trong các phần trước, bạn đã đo lường cách bạn sử dụng thời gian của mình. Bạn đã có một số thông tin về việc bạn phải mất bao nhiêu thời gian cho các nhiệm vụ khác nhau. Trong phần này, bạn sẽ sử dụng những thông tin này để quản lý thời gian của mình như sau:

1. Quyết định việc bạn muốn sử dụng thời gian của mình như thế nào.
2. Tạo một quỹ thời gian
3. Theo vết cách bạn sử dụng thời gian trong quỹ thời gian của mình
4. Quyết định mình cần phải thay đổi những gì giúp cho các hoạt động của mình hoàn thành trong quỹ thời gian đó

Những nội dung này sẽ được thảo luận trong những phần sau.

2.6.2 Phân loại các hoạt động của bạn

Hãy xem xét lại các mục thời gian của bạn có bao phủ hết các hoạt động chính yếu của bạn hay không. Nếu không hãy thêm hoặc sửa đổi lại chúng.

Bạn nên xem xét những mục hoạt động này xem chúng có quá chung chung hay quá chi tiết hay không. Một phân bố không cân đối có thể có một mục chiếm khoảng hoặc hơn 50% thời gian trong khi một số khác thì chỉ chiếm có 5% hoặc ít hơn.

Để quản lý thời gian, bạn cần phải tập trung vào một số ít các mục mà chiếm nhiều thời gian nhất. Bạn sẽ cần phải biết một ít chi tiết vào việc bạn làm. Nếu bạn tốn 25% thời gian trên một mục gán nhãn là *Các việc khác*, bạn nên phân nhỏ nó thành một số các hoạt động tốt hơn. Nếu như không có sẵn những thông tin về các hoạt động trong mục *Khác* này, bạn sẽ gặp khó khăn trong việc quản lý việc bạn đã tốn bao nhiêu thời gian cho chúng.

Mặc dù hiện tại bạn đã có ý thức tốt về việc mình sử dụng thời gian cho các nhiệm vụ trong tuần, nhưng bạn sẽ không có được nhiều thông tin trên những nhiệm vụ mà đôi khi xảy ra trong một tháng, một học kỳ, hoặc một năm học, ví dụ như việc học ôn thi hoặc việc làm trong những dự án ngắn hạn. Dĩ nhiên bạn không thể thu thập thông tin trên các nhiệm vụ này cho đến khi bạn thực hiện, nhưng hãy thử xác định những mục này bây giờ, điều này sẽ giúp bạn có một số dự phòng về chúng trong các kế hoạch của mình.

2.6.3 Đánh giá việc phân bổ thời gian của bạn

Bây giờ bạn đã có thể nhìn thấy việc bạn sử dụng thời gian của mình vào đâu, hãy tự hỏi mình xem bạn có sử dụng thời gian theo cách mà mình muốn hay không. Quyết định những hoạt động nào là quan trọng nhất và xem xét xem liệu bạn có cho chúng đủ thời gian chưa. Có nhiệm vụ nào tốn nhiều thời gian đến nỗi bạn không bao giờ có thể làm việc khác quan trọng hơn không? Bạn có đủ thời gian để đọc sách không? Bạn có việc làm không? Cam kết cá nhân của bạn là gì? Bạn có bắt đầu công việc được giao đúng lúc để hoàn thành chúng không, hoặc bạn có thường cuống quít vào phút chót không? Không có chỉ dẫn chung nào về cách bạn sử dụng thời gian của mình. Quyết định của chính cá nhân bạn phải cân bằng giữa việc học, việc làm, giải trí và cuộc sống xã hội. Một số trong số này là những vấn đề cá nhân liên quan đến nhiều yếu tố phức tạp, đặc biệt là khi bạn có những trách nhiệm trong công việc và trong gia đình.

2.6.4 Tạo quỹ thời gian

Quỹ thời gian là kế hoạch của bạn cho việc bạn sử dụng thời gian như thế nào. Bắt đầu từ những thông tin về cách mà bạn sử dụng thời gian trước đó, bạn có thể xác định khối lượng thời gian bạn muốn bỏ ra cho mỗi mục trong tương lai. Quỹ thời gian chính của một sinh viên Y được mô tả trong bảng dưới.

Sinh viên	Sinh viên Y	**Ngày**	9/23/96
Người hướng dẫn	Thầy Z	**Lớp**	CS1

Loại hoạt động	**Quỹ thời gian tối thiểu**	**Thời gian thực sự (phút)**
Lên lớp học	150	
Viết chương trình	360	
Đọc tài liệu	180	
Ôn thi	120	
Hoạt động khác	30	
Tổng cộng	840	

Bảng 2.6.1 Ví dụ một quỹ thời gian hằng tuần

Tên Sinh viên Y **Ngày** 23/9/96

Công việc / **Ngày**	**Tham dự lớp học**	**Viết chương trình**	**Chuẩn bị vấn đáp**	**Đọc tài liệu**				**Khác**	**Tổng cộng**
Chủ nhật 15/9									
Thứ 2	50	93		80					223
Thứ 3		95							95
Thứ 4	50			71					121
Thứ 5		77							77
Thứ 6	50	74		40					164
Thứ 7				33					33
Tổng cộng	150	339		224					713

Thời gian và tốc độ giai đoạn Số tuần (số trước +1) 2

Thời gian của tuần trước

Tổng cộng	150	341	134	146					771
Trung bình	150	341	134	146					771
Max	150	341	134	146					771
Min	150	341	134	146					771

Thời gian của tuần hiện tại

Tổng cộng	300	680	134	370					1484
Trung bình	150	340	67	185					742
Max	150	341	134	224					771
Min	150	339	134	146					713

Bảng 2.6.2 Tóm tắt hoạt động hằng tuần của sinh viên Y

Bảng 2.6.2 là một bản sao *Tóm tắt hoạt động trong tuần* của sinh viên Y từ Bảng 2.3.3. Hãy xem dòng 19, bạn có thể so sánh quỹ thời gian trong bảng 2.6.1 với cách sinh viên Y thật sự sử dụng thời gian của cô ấy. Điều này dẫn đến một vài kết luận sau:

- Tổng thời gian của sinh viên Y cho khóa học này trung bình là mất 742 phút mỗi tuần, nghĩa là 12,37 giờ. Với quỹ thời gian mới, cô ấy lên kế hoạch sử dụng 840

phút, hay 14 giờ mỗi tuần – thêm vào 1,5 giờ. Đây là một thay đổi lớn hợp lý và cô ấy có thể gặp khó khăn trong việc nhận thấy rằng có khá nhiều thời gian thêm vào cho mỗi tuần.

- Sinh viên Y tạo quỹ có cùng số lượng thời gian cho việc có mặt trên lớp học như cô ấy đã sử dụng, một kế hoạch hợp lý.
- Cô ấy lên kế hoạch sử dụng khoảng hơn 20 phút mỗi tuần cho việc viết chương trình. Điều này cũng hoàn toàn hợp lý.
- Cô ấy lên kế hoạch sử dụng cùng lượng thời gian với thời gian trung bình cho việc đọc sách. Điều này có vẻ hợp lý.
- Việc thêm nhiều thời gian hơn cả là trong các mục của việc ôn thi và *Khác*. Hai mục này được thêm tổng cộng là 150 phút mỗi tuần, hay 2,5 giờ. Đây là một thay đổi lớn không thực tế. Mặc dù nó có thể có ý nghĩa đối với sinh viên Y về việc tăng thời gian tổng cộng trong kế hoạch của cô ấy lên tới tối đa 771 phút mà cô ấy đã sử dụng trong chừng mực của khóa học, việc yêu cầu quá nhiều thời gian như vậy có thể không thực tế.

Điều chính yếu trong quản lý thời gian là cân bằng lại dần dần cách bạn sử dụng thời gian. Mặc dù bạn có thể hy vọng sử dụng nhiều thời gian hơn cho một số nhiệm vụ trong tương lai, nếu như bạn không xác định được những chỗ khác để giảm bớt thời gian thì đó thường là mơ tưởng hảo huyền. Một bước quan trọng là bảo đảm rằng thời gian mà hiện bạn đang sử dụng là thời gian được sử dụng hiệu quả nhất. Người ta thường lãng phí thời gian cho việc quyết định phải làm gì tiếp theo. Việc lập kế hoạch cá nhân và một quỹ thời gian sẽ giúp bạn biết được phải làm gì tiếp theo. Đáng ngạc nhiên là điều này sẽ lập tức cải thiện hiệu năng làm việc của bạn.

Bạn cũng có thể cần nhiều thời gian hơn để làm tất cả những việc mà bạn muốn. Tuy nhiên một lần nữa bạn cần phải thực tế. Bạn có thể nhận thấy rằng một số thời gian phải làm thêm khi cấp bách, nhưng không ai muốn sống trong cảm giác cấp bách cả. Vì thế xác định khoảng thời gian hợp lý cho việc giữ gìn sức khỏe, cho gia đình, bạn bè, sở thích, ngủ và ăn. Trong khi bạn có thể có một số điều chỉnh trong các phần này, bạn nên thực hiện chúng một cách từ từ. Bạn có thể cắt bỏ một số phần khi cấp bách, nhưng bạn không thể làm như vậy trong một khoảng thời gian dài. Nếu bạn phải sử dụng nhiều thời gian hơn cho một số hoạt động, bạn cần phải lấy thời gian đó từ những việc khác. Xem xét lại ưu

tiên của mình và quyết định cái gì là quan trọng nhất đối với bạn. Sau đó thiết lập quỹ thời gian của mình dựa vào đó.

2.6.5 Thiết lập các qui tắc cơ bản

Để quản lý thời gian hiệu quả, bạn cũng cần phải có quy tắc. Tuy nhiên hiện tại bạn phải tự thiết lập chúng. Thật sự đó chính là quỹ quản lý thời gian: các quy tắc bạn thiết lập cho việc quản lý chính thời gian của bạn.

Tên _____ **Ngày** _____

Quỹ Tuần #1

Nhiệm vụ									**Tổng cộng**
Ngày									
Chủ nhật									
Thứ hai									
Thứ ba									
Thứ tư									
Thứ năm									
Thứ sáu									
Thứ bảy									
Tổng cộng									

Quỹ Tuần #2

Nhiệm vụ									**Tổng cộng**
Ngày									
Chủ nhật									
Thứ hai									
Thứ ba									
Thứ tư									
Thứ năm									
Thứ sáu									
Thứ bảy									
Tổng cộng									

Quỹ Tuần #3

Nhiệm vụ									**Tổng cộng**
Ngày									
Chủ nhật									
Thứ hai									
Thứ ba									
Thứ tư									
Thứ năm									
Thứ sáu									
Thứ bảy									
Tổng cộng									

Bảng 2.6.3 Quỹ hoạt động hằng tuần

Để cung cấp một chỉ dẫn hữu ích cho việc ứng xử hằng ngày, bạn sẽ cần một quỹ thời gian hằng ngày. Quỹ của bạn cho những ngày và những tuần khác nhau có thể khác nhau. Vì vậy định dạng như trên sẽ là cần thiết.

Bây giờ bạn có thể thấy rằng tại sao quỹ trong bảng 2.6.1 không thật hữu dụng. Ví dụ sẽ có ý nghĩa nếu như bỏ ra thời gian mỗi tuần ôn thi. Bạn sẽ không muốn đợi đến ngày trước kỳ thi cũng như không muốn ôn quá sớm đến nỗi bạn sẽ quên hết kiến thức vào ngày thi.

Cũng như vậy, mục *Khác* trong bảng 2.6.1 là dành cho các hoạt động không lên kế hoạch trước. Mặc dù đó là một ý kiến tốt cho việc dành thời gian cho một số các sự kiện không lường trước, rõ ràng là không có cách nào để lập kế hoạch chi tiết cho những điều chưa biết cả. Một cách để xử lý các việc thường xảy ra không đoán trước là dành cho chúng một ít thời gian nhiều hơn thời gian trung bình cho mỗi hoạt động. Số dư tích lũy sẽ cho phép bạn xử lý các lỗi lên kế hoạch và các sự kiện không đoán trước.

Một tập các quỹ thời gian thực tế hơn cho sinh viên Y là như trong bảng sau:

Tên Sinh viên Y **Ngày** 23/9/96

Quỹ Tuần #1 Quỹ hằng tuần

Nhiệm vụ	Lớp	Viết	Thi	Đọc	Dự trữ			Khác	Tổng cộng
Ngày		Chương trình	Chuẩn bị	Sách					
Chủ nhật									
Thứ hai	50			40					90
Thứ ba		120		40					160
Thứ tư	50			40					90
Thứ năm		120		40					160
Thứ sáu	50			40					90
Thứ bảy		120							120
Tổng cộng	150	360		200					710

Quỹ Tuần #2 Tuần trước khi thi

Nhiệm vụ	Lớp	Viết	Thi	Đọc	Dự trữ			Khác	Tổng cộng
Ngày		Chương trình	Chuẩn bị	Sách					
Chủ nhật									
Thứ hai	50		40						90
Thứ ba		120	40						160
Thứ tư	50		40						90
Thứ năm		120	40						160
Thứ sáu	50		120						160
Thứ bảy			150						120
Tổng cộng	150	240	430						820

Quỹ Tuần #3 Tuần thi

Nhiệm vụ	Lớp	Viết	Thi	Đọc	Dự trữ			Khác	Tổng cộng
Ngày		Chương trình	Chuẩn bị	Sách					
Chủ nhật									
Thứ hai			300						300
Thứ ba			450						450
Thứ tư	170		150						320
Thứ năm	50								50
Thứ sáu									
Thứ bảy									
Tổng cộng	220		900						1120

Bảng 2.6.4 Quỹ hoạt động hằng tuần của sinh viên Y

2.6.6 Đặt độ ưu tiên cho thời gian của bạn

Một bước cần thiết trong việc quản lý thời gian là thiết lập độ ưu tiên. Một số thời gian được thiết lập cứng, giống như khi bạn lên lớp học hoặc khi bạn làm việc bán thời gian. Bạn có thể gọi đó là những thời gian cố định. Mọi thứ khác là các hoạt động có thời gian thay đổi mà bạn làm khi bạn có thời gian. Hoạt động thay đổi có hai loại:

- Hoạt động được yêu cầu làm (làm bài tập về nhà, đọc sách, hoặc học ôn thi…). Chúng thay đổi vì bạn có thể làm chúng bất cứ khi nào bạn có thời gian và bạn sẽ tốn khối lượng thời gian khác nhau cho chúng mỗi tuần.

- Nhu cầu cơ bản (tất cả những việc khác như ăn, ngủ, hoạt động xã hội, giải trí…)

Lập lịch cho các việc cố định thì không có vấn đề, vấn đề chung nhất là xác định thời gian thay đổi. Ví dụ nếu như bạn dường như không có đủ thời gian để làm bài tập về nhà đúng hạn, chỉ có một cách duy nhất là lấy thời gian từ các nhu cầu cơ bản. Điều này gợi ý cho ta xem xét kỹ hơn tới thời gian cố định, được yêu cầu làm và nhu cầu cơ bản để cho thấy chỗ nào bạn có thể điều chỉnh.

Bản tóm tắt thời gian chung như trong bảng sau giúp bạn xem xét kỹ hơn việc phân bố thời gian của mình. Để hoàn thành một bản tóm tắt, bạn sẽ phải theo vết đại khái bao nhiêu thời gian mà bạn thường dùng cho mỗi việc. Khởi đầu, chỉ cần ghi chú vào cuối mỗi ngày rằng bạn tốn thời gian bao nhiêu và ở đâu cho mỗi mục. Nếu bạn chắc chắn rằng tổng thời gian cho mỗi ngày mà bạn dùng là 24 giờ và tổng thời gian cho mỗi tuần là 168 giờ hoặc 10.080 phút, thì tổng thời gian của bạn có thể là khá tốt. Nếu bạn không tạo được bản ghi đầy đủ theo cách này, lúc này bạn hãy sử dụng đến bản ghi thời gian.

Tên	Sinh viên Y	**Ngày**	9/23/1996
Người hướng dẫn	Thầy Z	**Lớp**	CS1

Hoạt động	Tin học	Vật lý	Toán	Anh văn	Ăn/Nghỉ	Khác	Tổng cộng
Cố định							
Lớp	150	150	100	100			500
Được yêu cầu							
Bài tập về nhà	360	240	240	360			1200
Đọc sách	240	240	180	60			720
Nhu cầu cơ bản							
Ăn					1260		1260
Ngủ					3150		3150
Thể thao						600	600
Giải trí						360	360
Nghỉ ngơi						2290	2290
Tổng cộng	750	630	520	520	4410	3250	10080

Bảng 2.6.5 Bản tóm tắt thời gian hằng tuần

Với bản tóm tắt tổng hợp như trên, bạn có được thông tin cơ sở cho việc quyết định nơi nào cần thêm thời gian. Tuy nhiên một lần nữa bạn cần phải có cái nhìn thực tế. Cắt giảm thời gian ngủ và ăn, có thể gây nguy hại đến sức khỏe của bạn và làm giảm hiệu năng làm việc. Bỏ bữa ăn và thức thâu đêm có thể là một cách giải quyết trong lúc cấp bách, nhưng nói chung đó không phải là cách tốt cho việc lập kế hoạch cho việc cắt giảm chính trong các phần này.

Thực tế dẫn đến một vấn đề khác: sự sẵn lòng của bạn trong việc thực sự làm việc như trong kế hoạch đã thiết lập. Một lịch biểu mà không có thời gian cho các hoạt động xã hội, nghỉ ngơi, hoặc tập thể thao có thể gây buồn chán, điều này có thể dẫn đến việc bạn nổi dậy chống lại lại toàn bộ ý định quản lý thời gian. Vì việc theo dõi và lập quỹ thời gian là quan trọng, hãy chắc chắn rằng bạn thực sự muốn sống với một quỹ thời gian như vậy.

2.6.7 Quản lý quỹ thời gian của bạn

Để lập các quy tắc cơ bản cho việc quản lý thời gian, hãy dùng hướng tiếp cận mô tả ở dưới, liệt kê các thời gian cho mỗi mục trong mỗi ngày.

Tên Sinh viên Y **Ngày** 9/23/1996

Quỹ Tuần #1 Quỹ hằng tuần

Nhiệm vụ / **Ngày**	Lớp	Viết chương trình	Chuẩn bị thi	Đọc Sách	Dự trữ		Khác	Tổng cộng
Chủ nhật								
Thứ hai	9:00 – 9:50			10:20 – 11:00				90
Thứ ba		8:30 – 10:30		10:20 – 11:00				160
Thứ tư	9:00 – 9:50			10:20 – 11:00				90
Thứ năm		8:30 – 10:30		10:20 – 11:00				160
Thứ sáu	9:00 – 9:50			10:20 – 11:00				90
Thứ bảy		8:30 – 10:30		10:20 – 11:00				160
Tổng cộng	150	360		240				750

Bảng 2.6.6 Ví dụ về Quỹ và lịch biểu thời gian

Khả năng làm việc đúng theo quỹ thời gian phụ thuộc rất nhiều vào kỷ luật cá nhân nhưng nó cũng phụ thuộc vào số lượng và độ ưu tiên của các công việc mà bạn định làm.

Các sự cố không được mong đợi là một phần bình thường của cuộc sống, đặc biệt là trong công nghệ phần mềm. Khủng hoảng sẽ thường xuyên phá hủy kế hoạch của bạn và bạn sẽ phải thực hiện các điều chỉnh. Ví dụ để kịp một thời hạn quan trọng, bạn phải thức khuya làm việc hoặc bỏ bớt những hoạt động xã hội, giải trí và gia đình đã định trước.

Bạn có thể sẽ thấy lần đầu tiên sử dụng quỹ thời gian không mang lại ích lợi lắm. Đừng từ bỏ quy trình lập quỹ thời gian chỉ vì nó không hoạt động tốt trong lần đầu tiên, thay vào đó hãy suy nghĩ về những gì đã xảy ra. Có sự kiện bất thường nào xảy ra hoặc bạn đã tiêu phí thời gian vào một việc bình thường nào đó mà bạn không biết hay không? Nếu có thì bạn không cần phải thực hiện các thay đổi cơ bản trong quỹ. Hãy thử sử dụng nó thêm một tuần nữa và sau đó kiểm tra lại kết quả. Nếu quỹ vẫn bị phá vỡ do một số sự kiện ngày thường thì hãy xét đến việc điều chỉnh quỹ thời gian để bao gồm tất cả những sự kiện này trong tương lai.

2.6.8 Mục tiêu quản lý thời gian

Sau khi bạn quản lý thời gian theo cách này được một vài tuần, hãy xem xét đến việc đơn giản hóa cách thu thập dữ liệu bằng cách nhóm các loại công việc lại thành một số ít loại hơn. Mục tiêu của bạn tại thời điểm này là có được một bức tranh toàn cảnh của việc sử dụng thời gian của mình chứ không phải là các chi tiết.

Hãy nhớ rằng mục đích của việc thu thập dữ liệu thời gian là giúp chính mình quản lý thời gian. Nếu dữ liệu mà bạn thu thập không chứng minh được tính hữu dụng của nó thì hãy xét lại cách mà bạn thu thập dữ liệu. Tuy nhiên hãy làm như thế chỉ khi nào bạn đã thực hành lập quỹ thời gian của mình trong thực tế. Thậm chí sau đó nếu do một số lý do nào đó mà việc sử dụng thời gian của bạn thay đổi đáng kể, hãy thu thập thêm dữ liệu cho đến khi bạn hiểu được mình đang sử dụng thời gian như thế nào.

2.7 Quản lý cam kết

Phần này tập trung vào cam kết, thảo luận xem đó là cái gì, tại sao nó lại quan trọng, và làm sao quản lý được nó.

2.7.1 Định nghĩa

Được cam kết là một trạng thái ý nghĩ. Đó là, bạn phải nhận một nhiệm vụ vì một lý do nào đó, và bạn cảm thấy mình nên làm điều này. Tuy nhiên, sự cam kết không chỉ là bạn định một việc gì đó, mà còn là có người khác mong muốn bạn làm việc này. Đây mới thật sự là điểm mấu chốt của vấn đề: bạn sẽ cam kết với ai? Hiểu theo nghĩa thường, bạn cam kết với ai đó: với thầy, với sếp, hay với ông chủ của bạn. Tuy nhiên, quan trọng hơn là những cam kết với chính bản thân mình.

Vấn đề thường gặp với nhiều lịch biểu và kế hoạch phần mềm là nhà quản lý xem chúng như những cam kết hợp đồng, trong khi kỹ sư phần mềm lại không xem chúng là những cam kết cá nhân. Chúng ta sẽ thấy rằng sự khác biệt thể hiện nhiều ở cách đưa ra những cam kết này. Trong phần này, các bạn sẽ nghiên cứu cách thức đưa ra cam kết hợp đồng vì chúng cũng là cam kết cá nhân.

Đặc tính chủ yếu của cam kết cá nhân là chúng mang tính tự nguyện. Để cam kết trở thành sự thật, bạn phải xem xét kỹ càng các phương án và quyết định rằng đây là việc có thể làm được và bạn sẽ làm nó.

Sự đồng thuận thật sự là đặc điểm quan trọng nhất trong lời cam kết cá nhân. Các bên phải đồng ý về những việc cần phải làm, thời điểm hoàn thành, và ngược lại sẽ nhận được cái gì.

Một cam kết rõ ràng vừa mang tính cá nhân vừa mang tính hợp đồng. Nó yêu cầu một thoả thuận rõ ràng và tự nguyện giữa hai hay nhiều bên về các điểm:

- Cái gì sẽ được làm.
- Các tiêu chuẩn hoàn thành công việc.

- Ai sẽ làm.
- Khi nào thì hoàn tất
- Các điều khoản đền bù, thanh toán sẽ được nhận.
- Ai sẽ đền bù hay cung cấp các khoản thanh toán.

2.7.2 Các lời cam kết được thực hiện hợp lý

Cùng với các đặc điểm đã được giới thiệu bên trên, các lời cam kết phải được đưa ra hợp lý và quản lý đúng đắn. Bạn có thể đảm bảo các lời cam kết của bạn là hợp lý và được quản lý tốt theo những cách sau.

Phân tích công việc trước khi đồng ý bằng cam kết. Các bên có liên quan phải tin tưởng nhau. Bạn cam kết với cá nhân và thật sự muốn làm công việc này và ngược lại, bên còn lại cũng muốn thanh toán thích hợp. Tuy nhiên, câu hỏi là cả hai bên đạt được sự thoả thuận ở mức độ nào. Ví dụ, bạn có xem xét công việc ở mức chi tiết có thể để xem bạn có thể thực hiện nó được hay không? Tương tự, bên còn lại có khả năng đáp ứng thanh toán cho bạn không?

Hỗ trợ lời cam kết bằng một kế hoạch. Với công việc có kích cỡ lớn đến mức nào, cách để thực hiện cam kết hợp lý là đầu tiên tạo ra kế hoạch cho công việc.

Lập tài liệu cho thoả thuận. Trao đổi bằng lời thường gây hiểu lầm. Thậm chí sau khi hai người thoả thuận miệng xong, họ còn gặp rắc rối với việc đồng ý ở các văn bản thoả thuận. Lý do chính của hầu hết các hợp đồng giấy là thỏa thuận về công việc mà các bên phải làm và phòng khi xảy ra sự cố.

Nếu không thể thực hiện đúng cam kết, hãy trao đổi với bên còn lại và cố gắng giảm thiểu hậu quả đối với bên đó. Một khi bạn đã học xong cách quản lý cam kết, hầu như bạn sẽ luôn thực hiện đúng cam kết. Song, thậm chí với các kế hoạch tốt nhất, công việc thường phức tạp hơn bạn tưởng và sẽ có nhiều thứ chẳng mong muốn sẽ xảy ra.

2.7.3 Ví dụ về một lời cam kết

Sinh viên Y đồng ý làm việc bán thời gian với thời lượng 10 giờ 1 tuần cho văn phòng tuyển sinh của một trường đại học. Các bước sau liên quan đến việc tạo ra một cam kết đúng đắn dựa theo trường hợp của sinh viên đó.

1. Sinh viên Y đến văn phòng gặp người trưởng phòng. Người này sẽ giải thích công việc mà sinh viên Y sẽ làm. Đây là giai đoạn yêu cầu của quá trình tạo cam kết. Trong giai đoạn này, sinh viên Y sẽ biết chính xác về loại công việc. Đối với công việc văn

phòng, kết quả của giai đoạn này là một danh sách các nhiệm vụ cần thực hiện, có thể là một bảng mô tả công việc.

2. Sinh viên Y xem xét các nhiệm vụ và kết luận rằng đó là công việc mà cô đã sẵn sàng làm và có khả năng thực hiện.

Trả lời xong câu hỏi thứ nhất, **bạn có thể làm công việc này không?**, đến câu hỏi thứ hai, **bạn có thể làm công việc vào thời gian và với tài nguyên được yêu cầu hay không?** Để trả lời được câu hỏi này, sinh viên Y phải làm như sau:

3. Cô ấy kiểm tra các cam kết thời gian cá nhân của mình và kết luận rằng cô ấy có thể có được 10 giờ (hay 600 phút) một tuần để làm công việc. Bạn có thể thấy điều này bằng cách so sánh bảng 8.1 với bảng 7.5, cô ta có được khoảng thời gian này nhờ vào việc giảm tổng thời gian nghỉ ngơi 600 phút.

4. Cô ấy xem xét lại các cam kết của bản thân, như được liệt kê trong bảng 8.2, và kết luận rằng khoảng thời gian cần cho công việc, từ 4g đến 6g chiều từ thứ hai đến thứ sáu, còn trống.

5. Bây giờ, cô ấy đã có thể làm công việc, cô ấy sẽ thoả thuận tiền lương và ngày mà vị trưởng phòng muốn cô bắt đầu công việc.

6. Với các vấn đề đặt ra, sinh viên Y và trưởng phòng đồng ý trên tất cả các khoản được thảo luận và người trưởng phòng đồng ý cung cấp bản tổng kết các điểm quan trọng.

Tên: Sinh viên Y. **Ngày**: 9/30/96
Người hướng dẫn: Thầy Z. **Lớp** CS1

Hoạt động	**Tin học**	**Vật lý**	**Toán**	**Tiếng Anh**	**Ăn/Nghỉ**	**Khác**	**Tổng cộng**
Cố định							
Lớp	150	150	100	100			500
Công việc						600	600
Cần							
Bài tập về nhà	360	240	240	360			1220
Đọc tài liệu	240	240	180	60			720
Tuỳ ý							
Ăn					1260		1260
Ngủ					3150		3150
Thể thao						600	600
Giải trí						360	360
Nghỉ ngơi						1690	1690
Tổng cộng	**750**	**630**	**520**	**520**	**4410**	**3250**	**10.080**

Bảng 2.7.1 Bảng tổng kết thời gian hàng tuần của sinh viên Y

Tên: Sinh viên Y. **Ngày:** 9/30/96

Nhiệm vụ	Tin học	Vật lý	Toán	Tiếng Anh	Công việc				Tổng cộng
Thứ									
H	9:00-9:50	10:00-10:50			4:00-6:00				220
B			9:00-9:50	10:00-10:50	4:00-6:00				220
T	9:00-9:50	10:00-10:50			4:00-6:00				220
N			9:00-9:50	10:00-10:50	4:00-6:00				220
S	9:00-9:50	10:00-10:50			4:00-6:00				220
B									
C									
Tổng cộng	**150**	**150**	**100**	**100**	**600**				

Bảng 2.7.2 Các cam kết cố định hàng tuần của sinh viên Y

Đây là một cam kết hợp lý. Sinh viên Y mất một khoảng thời gian để hiểu công việc và khả năng thực hiện. Cô ta tin rằng có thể hoàn thành công việc và có thời gian rảnh để thực hiện. Cô cũng đồng ý với mức lương và các điều kiện công việc khác mà trưởng phòng đưa ra.

2.7.4 Giải quyết các cam kết bị bỏ lỡ

Sau khi bạn đã lập kế hoạch thuần thục, bạn có thể không còn bỏ lỡ các thời hạn thường xuyên. Tuy nhiên, không có gì đảm bảo bạn sẽ hoàn toàn không bỏ lỡ cái nào. Các thứ mà các kỹ sư phần mềm phải thực hiện thường là mới mẻ và sáng tạo, do đó có một rủi ro đáng kể. Nếu bạn theo tất cả các rủi ro có thể, các ước tính của bạn sẽ vượt lên rất cao, vì vậy một thời gian biểu không thường xuyên và mất chi phí là không thể tránh được.

Khi bạn phải bỏ lỡ một cam kết, nhanh chóng báo cho đối tác và những người cùng làm việc biết để cùng giải quyết các vấn đề có thể xảy ra. Ví dụ, bạn có thể đồng ý trên khung thời gian dài hơn hoặc phạm vi công việc giảm đi. Với phần mềm, một chiến lược thường gặp là phân phối các phiên bản với chức năng tối thiểu tại hoặc gần thời gian dự kiến ban đầu, sau đó cung cấp phiên bản với các chức năng cải tiến, nâng cao. Bạn có thể làm tối thiểu hoá các nguy cơ đổ vỡ hợp đồng với khách hàng bằng cách xác định đúng các trình tự và thời gian cho các chức năng sau đó.

Có một chú ý quan trọng: đừng từ bỏ các nỗ lực cố gắng đáp ứng cam kết. Bạn đã kiểm tra với một chuyên gia độc lập xem có cách nào tốt hơn chưa? Có thể thêm các tài nguyên vào làm tăng tốc công việc không? Có thiết kế nào khôn ngoan hơn không? Xem xét cặn kẽ các chọn lựa và sau đó, nếu không có cách khác thì đừng trì hoãn đối mặt vấn đề.

Bỏ lỡ các cam kết thường dẫn đến sự bất tiện và không vui. Đó là lý do tại sao người ta thường hoãn việc đối mặt với các sự cố như vậy cho đến thời điểm cuối cùng. Họ trì hoãn sự khó chịu với hy vọng sự cố sẽ kết thúc. Tuy nhiên, càng làm như vậy sẽ càng lún sâu hơn, làm gia tăng nguy cơ gẫy đổ, giảm các chọn lựa của khách hàng.

2.7.5 Hậu quả của việc không quản lý cam kết

Công việc được yêu cầu đã vượt thời gian cho phép. Bạn thường có nhiều việc để làm hơn là bạn có thể chịu được. Nếu bạn không giữ danh sách các cam kết của bạn, có thể bạn phải chịu một cam kết mới tại thời điểm bạn không nên có.

Không đạt được cam kết. Các công việc phát triển phần mềm thường phức tạp hơn người ta nghĩ. Khi bạn không có một cách để đưa ra cam kết một cách trật tự, gần như bạn sẽ giả định rằng công việc thì đơn giản hơn thực tế. Bạn thường cam kết quá khả năng vào lúc bắt đầu thực hiện công việc.

Độ ưu tiên không đúng chỗ. Khi cam kết quá khả năng, người ta thường đặt các độ ưu tiên dựa vào cái cần hoàn tất trước hơn là dựa vào cái nào quan trọng nhất. Khi bạn bị cam kết quá khả năng nghiêm trọng, bạn cần phải sắp xếp lại tất cả các cam kết để phù hợp với những cái bạn có thể làm. Bạn có thể thấy được các công việc quan trọng hơn đến sau bằng cách hoãn hoặc bỏ một vài nhiệm vụ trước mắt.

Công việc chất lượng thấp. Dưới áp lực của thời gian biểu, các kỹ sư phần mềm thường bất cẩn và gây ra các lỗi ngớ ngẩn trong khi nhất thiết phải chú ý đến chất lượng. Khi thời gian còn ít, các kỹ sư nên đặc biệt chú ý để tránh lỗi. Kinh nghiệm cho thấy đây là lúc các kỹ sư cũng như những người quản lý ít theo trình tự xem xét, thanh tra, hay kiểm tra toàn bộ.

Mất niềm tin. Nếu bạn thường xuyên bỏ lỡ cam kết, người ta sẽ chú ý và hiểu ra rằng nếu bạn cam kết việc gì đó, bạn thường không thực hiện đúng. Những lần lặp lại như vậy rất khó sửa chữa và ảnh hưởng đến xếp loại, đến đánh giá công việc, đến tiền lương, và thậm chí đến việc làm của bạn.

Các ý kiến của bạn không tôn được tôn trọng. Khi không còn tin vào những gì bạn nói, người ta ít khi hỏi đến quan điểm của bạn và thường cho rằng bạn không làm việc theo thời gian biểu hợp lý.

Tài sản quý nhất của một kỹ sư phần mềm là hoàn tất được các cam kết thường xuyên. Để người ta tin bạn, bạn phải nói kế hoạch bạn sẽ làm và làm những gì bạn nói.

Một trong những mục đích chính của tài liệu này là cung cấp các công cụ giúp bạn tạo ra các cam kết thực tế có thể hoàn tất được.

2.7.6 Cách quản lý cam kết

Quản lý cam kết đúng đắn bắt đầu bằng việc tạo ra một danh sách các cam kết bạn có. Lưu ý đến thời hạn mỗi cam kết và tổng số thời gian cho mỗi cam kết.

Tên Sinh viên Y **Ngày:** 30/9/96
Giảng viên: Thầy Z. **Lớp :** CS1

Ngày tới hạn	Cam kết	Đến ai?	Giờ	Ngày làm	Sẽ được
Hàng tuần					
2-4-6	Học lớp CS1	Giáo viên	1.5		Điểm
2-4-6	Nộp bài CS1	Giáo viên	6.0		Điểm
3-5	Đọc tài liệu	Giáo viên	4.0		Điểm
2->6	Việc làm thêm, 4 – 6 chiều	Giám sát	10.0	1/9	Trả tiền
Khác					
11/28	Kiểm tra học kỳ	Giáo viên	24	11/9	Điểm

Bảng 2.7.3 Danh sách các cam kết của sinh viên Y

Đối với quản lý cam kết trong công nghệ phần mềm, cần phải nhớ các yếu tố của cuộc sống ảnh hưởng đến nghiệp vụ phần mềm:

- Nếu bạn rớt lại đằng sau, thời gian biểu của bạn sẽ tiếp tục trượt lại sau trừ phi bạn làm một cái gì khác đi.

- Chỉ cố gắng làm việc chăm chỉ hơn thì chưa đủ. Nên nhớ, bạn đã cố gắng làm việc chăm chỉ rồi.

- Nếu bạn không biết được chính xác bạn đang ở đâu trong dự án và công việc của bạn còn bao nhiêu thì bạn hầu như chắc chắn đang gặp sự cố.

- Khi bạn cần phải có may mắn để hoàn tất cam kết thì bạn sẽ không đạt được cam kết đó đâu.

- Khi bạn ước lượng sai, giá trị đó thường là quá thấp.

- Hầu như tất cả thay đổi đều sẽ làm tăng công việc của bạn.

Làm việc tháo vát để đạt được cam kết là rất quan trọng nhưng nếu bạn không thể hoàn tất công việc trong một vài giờ nỗ lực thêm thì nên đối mặt với sự cố và giải quyết nó hợp lý.

2.8 Quản lý thời gian biểu

Phần này sẽ trình bày cách sử dụng và khai thác thời gian biểu để theo dõi quá trình làm việc của bạn. Bạn sẽ học được cách sử dụng các điểm mốc để theo dõi tiến trình làm việc đối chiếu thời gian biểu của bạn.

2.8.1 Sự cần thiết của thời gian biểu

Bạn lập ra thời gian biểu để giải quyết các công việc khi có nhiều việc làm cùng một lúc. Với những công việc hoặc dự án nhỏ, thông thường bạn bắt đầu một việc mới khi bạn đã hoàn tất việc trước đó. Cách làm này chỉ khả thi đối với một số việc ngắn và đơn giản hoặc là chúng ta có đủ thời gian để hoàn thành. Khi công việc trở nên lớn hơn, phức tạp hơn hoặc là chúng ta thiếu thời gian thì cách làm trên sẽ không mang lại hiệu quả. Do đó bạn cần phải có một cách quản lý thời gian tinh vi hơn và hiệu quả hơn, phải uyển chuyển giữa các công việc, làm việc này một lúc rồi sau đó chuyển qua làm việc khác.

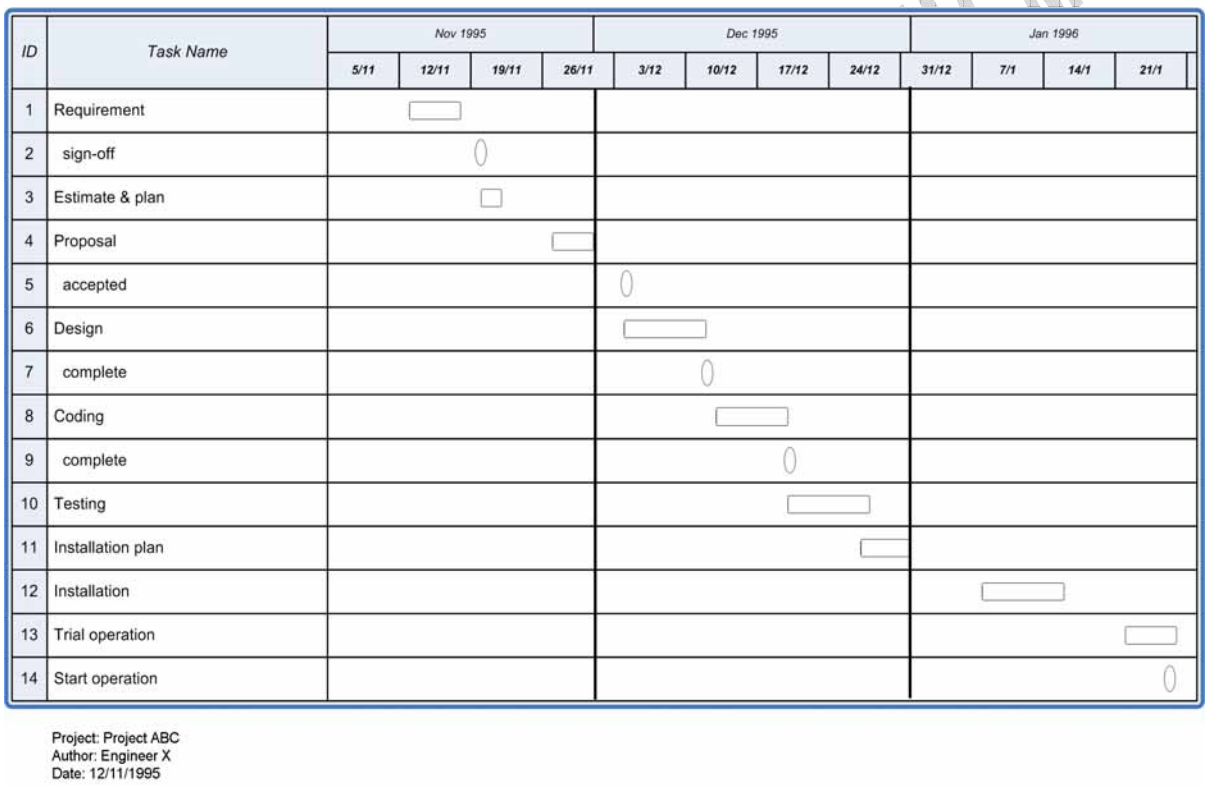
Khi làm việc với những dự án càng lớn thì việc xếp thời gian biểu của bạn càng quan trọng. Một kỹ sư phần mềm thông thường phải kiểm thử nhiều mô đun của dự án đang phát triển đồng thời phải làm việc với những người viết báo cáo cho dự án. Để giải quyết những câu hỏi phát sinh của chương trình thường có những cuộc họp giữa hai bên. Những kỹ sư không biết làm sao để điều phối các công việc hàng ngày của mình sao cho không quên hoặc xem nhẹ những việc đó. Và họ cũng phải thường xuyên phối hợp với nhiều người khác. Để cho mọi công việc thật sự trôi chảy và hiệu quả, họ cần phải tuân theo thời gian biểu cá nhân của mình.

2.8.2 Biểu đồ Gantt

Một thời gian biểu là một danh sách được sắp xếp thời gian của những việc có kế hoạch. Hình dưới là một thời gian biểu. Đây là một biểu đồ Gantt được mô tả như sau:

- Dòng đầu tiên đi từ trái sang phải là lịch các ngày làm việc. Tùy vào mức độ chi tiết mà bạn có thể phân thời gian theo ngày, tuần, tháng, năm và thậm chí có thể là giờ.
- Cột bên trái đánh số cho từng công việc của bạn.
- Ở cột thứ hai, từ trên xuống dưới là tên các công việc của cần thực hiện.
- Bên trong biểu đồ, các thanh ngang (ô hình chữ nhật) thể hiện thời điểm bắt đầu và kết thúc một công việc.
- Ở dưới bảng ghi tên dự án, người đề ra thời gian biểu và ngày tạo thời gian biểu.

- Các điểm mốc khác nhau được thể hiện bằng các hình bầu dục nhỏ. Điểm mốc sẽ được bàn luận rõ hơn trong phần sau.



Hình 2.8.1 Ví dụ về biểu đồ Gantt

Như đã thấy, biểu đồ Gantt là cách tốt nhất để thể hiện "luồng" công việc trong dự án của bạn. Các biểu đồ này sẽ giúp cho bạn phối hợp các công việc hiệu quả hơn. Biểu đồ Gantt còn giúp cho bạn theo dõi tiến trình của dự án tốt hơn đối chiếu với kế hoạch và thời gian biểu. Điều này sẽ được giải thích cặn kẽ trong phần sau.

2.8.3 Lập thời gian biểu

Bước đầu tiên để lập thời gian biểu là phải phân tích công việc lớn chi tiết ra thành các công việc nhỏ, sau đó ước lượng khối lượng của từng việc nhỏ này và xác định tổng số công việc mà bạn sẽ phải làm. Cuối cùng, liệt kê từng công việc đó trên biểu đồ Gantt với thanh thời gian để biết được khi nào việc này bắt đầu và kết thúc.

Khi lập thời gian biểu cho những việc có liên quan đến nhiều người khác, bạn cần thực hiện thêm vài bước sau:

- Phải chắc rằng mỗi người biết được những việc mà mình có nhiệm vụ phải làm.
- Định được ngày cam kết phải làm xong cho mỗi công việc.

- Xác định sự liên thuộc giữa các công việc. Mỗi người phải nhận được gì trước khi tiến hành công việc của mình và nhận từ ai.

- Viết báo cáo cho những sự liên thuộc này.

- Xem lại thời gian biểu đề nghị và những liên thuộc với tất cả những người có liên quan để đảm bảo không có sự xung đột, bất đồng hoặc hiểu nhầm.

- Xem lại thời gian biểu một lần nữa để đảm bảo thời gian biểu này bao gồm hết tất cả công việc nhỏ cần thiết để hoàn thành hết công việc lớn ban đầu đã định.

Đôi khi bạn cần thêm một số bước để lập thời gian biểu cho những dự án phần mềm, nhưng với những nguyên tắc cơ bản này sẽ giúp bạn lập ra những kế hoạch cho việc cá nhân của bạn hoặc cho những nhóm nhỏ của bạn.

2.8.4 Điểm mốc

Mỗi khi một mục trong thời gian biểu được thực hiện xong, bạn đánh dấu đã thêm được một mức xử lý. Điểm đánh dấu như thế được gọi là điểm mốc hay cột mốc. Điểm mốc là một phần quan trọng trong việc lập kế hoạch cho một dự án và quản lý dự án.

Điểm mốc là điểm xác định mang tính khách quan trong một dự án, nhìn vào đó bạn có thể dễ dàng nhận thấy bạn đang làm đúng theo thời gian biểu hay là đã chậm trễ.

Điểm mốc chỉ thực sự hữu ích khi nó rõ ràng và không mơ hồ. Điểm mốc phải là một hành động rõ ràng mà có thể được thực hiện hoặc không. Sau đây là một vài điểm mốc chính xác:

- Bạn đã hoàn thành kỳ thi cuối khóa.

- Bạn đã lập và báo cáo kế hoạch để viết chương trình, dùng theo mẫu báo cáo chuẩn.

- Bạn đã xem xét kế hoạch phát triển cùng với trợ lý và đề nghị một vài sự thay đổi.

- Bạn đã hoàn thành và báo cáo bản thảo chương trình, dùng dạng bản thảo cụ thể.

- Bạn đã cài đặt, biên dịch, và sửa lỗi chương trình để chương trình biên dịch tốt mà không mang lỗi.

Có rất nhiều ví dụ về điểm mốc tương ứng cho mục đích lập kế hoạch. Yêu cầu chính là sự hoàn thành của mỗi điểm mốc phải rõ ràng, xác thực.

Những phát biểu chung chung không mang tính xác thực, rõ ràng thì không được xem như là những điểm mốc. Chẳng hạn những điểm mốc không hợp lý, quá chung chung và mập mờ như sau:

- Bạn đã lên một kế hoạch để viết chương trình. (Làm sao bạn biết được liệu bản kế hoạch có chứa những thông tin cần thiết hay không? Bản kế hoạch này đã hoàn chỉnh hay chưa?)

- Bạn đã thiết kế một chương trình. (Từ câu phát biểu này, làm sao bạn biết được cái gì tạo thành một bản thiết kế hoàn chỉnh? Nếu bạn thêm vào những cái mà bản thiết kế đã được báo cáo trong một mẫu định sẵn, thì nó được xem như điểm mốc.)

- Việc lập trình đã hoàn thành hết 90%. (Câu phát biểu này sẽ làm cho bạn lo lắng vì tính mập mờ của nó. Tuy nhiên, nếu ai đó nói với bạn rằng họ đã lập trình xong, đã xem lại, đã sửa lỗi của 7 mô đun trong tổng số 9 mô đun và đã bắt đầu mô đun thứ 8, lúc đó bạn có thể yên tâm vì người đó biết được họ nói gì)

Những đề xuất trong việc thiết lập điểm mốc

Khi lập kế hoạch, việc thiết lập các điểm mốc cho các dự án sẽ tốn nhiều giờ. Bạn nên chọn số lượng điểm mốc phù hợp với số lượng công việc liên quan. Không nên vượt quá 1 điểm mốc cho một công việc được làm trong 3 đến 5 giờ. Tốt nhất là thiết lập điểm mốc cho công việc sau mỗi 5 giờ hoặc nhiều hơn. Nếu bạn có nhiều điểm mốc hơn thì bạn sẽ tốn nhiều thời gian cho việc theo dõi, đôi khi sẽ không cần thiết.

Đối với những việc phải tốn hàng tuần, bạn phải thiết lập ít nhất một điểm mốc cho mỗi tuần cho dù bạn chỉ tốn có ½ giờ hoặc hơn cho công việc đó trong tuần. Điểm mốc cần thiết này dùng để nhắc bạn làm việc theo kế hoạch đã định. Nếu như không được nhắc nhở thì bạn dễ dàng bỏ qua những công việc nhỏ.

Những dự án lớn hơn thường tốn nhiều tuần hoặc nhiều tháng và được thực hiện bởi nhiều kỹ sư phần mềm. Bởi vì các việc trong dự án thì có liên quan với nhau và có liên quan đến dự án khác, do đó bạn không thể hoàn thành công việc của mình cho đến khi các kỹ sư khác hoàn thành việc của họ. Điều quan trọng đối với một kỹ sư làm việc trong một dự án là phải có một vài điểm mốc trung gian để từ đó mỗi người đều có thể biết được tình trạng của tất cả các công việc khác. Nếu như có một vài kỹ sư gặp khó khăn thì những người khác có thể giúp đỡ họ hoặc là sắp xếp lại công việc của họ nhằm ngăn chặn sư trì hoãn trên toàn dự án.

2.8.5 Theo dõi các kế hoạch của dự án

Theo dõi kế hoạch dự án cho phép bạn xác định được dự án đang theo kịp, đi trước, hay ở phía sau thời gian biểu của bạn. Các điểm mốc chính xác và một kế hoạch chi tiết có thể cho bạn biết được chính xác những phần nào của dự án đang gặp khó khăn và cần được

giúp đỡ để bắt kịp với tiến độ của công việc. Những điều này sẽ trở nên quan trọng khi kích thước dự án tăng lên. Và càng trở nên quan trọng hơn khi sự cam kết bị bỏ qua gây ra những hậu quả nghiêm trọng. Việc báo cáo tình hình công việc chính xác rất cần thiết khi thực hiện những dự án cho khách hàng.

Lý do khác để thực hiện việc theo dõi dự án là để có thể hành động kịp thời khi xảy ra sự cố. Với việc theo dõi hiệu quả, bạn sẽ sớm nhận ra được vấn đề, biết cách khắc phục chúng, và thường là có thể khắc phục được. Trên thực tế, một hệ thống theo dõi tốt có thể giúp bạn dự đoán được sự cố trước khi chúng trở nên nghiêm trọng đủ để đe dọa sự thành công của dự án.

Một ví dụ

Một biểu đồ Gantt có thể được sử dụng để mô tả kế hoạch dự án và để công bố tiến trình theo thời gian biểu. Ví dụ, trong hình dưới, kỹ sư X đưa ra tình hình của ông vào ngày 13/12. Chú ý các điểm sau đây:

- Ngày cập nhật tình hình được biểu hiện bởi đường thẳng đứng đôi xuyên suốt lịch biểu ở ngày 13/12.

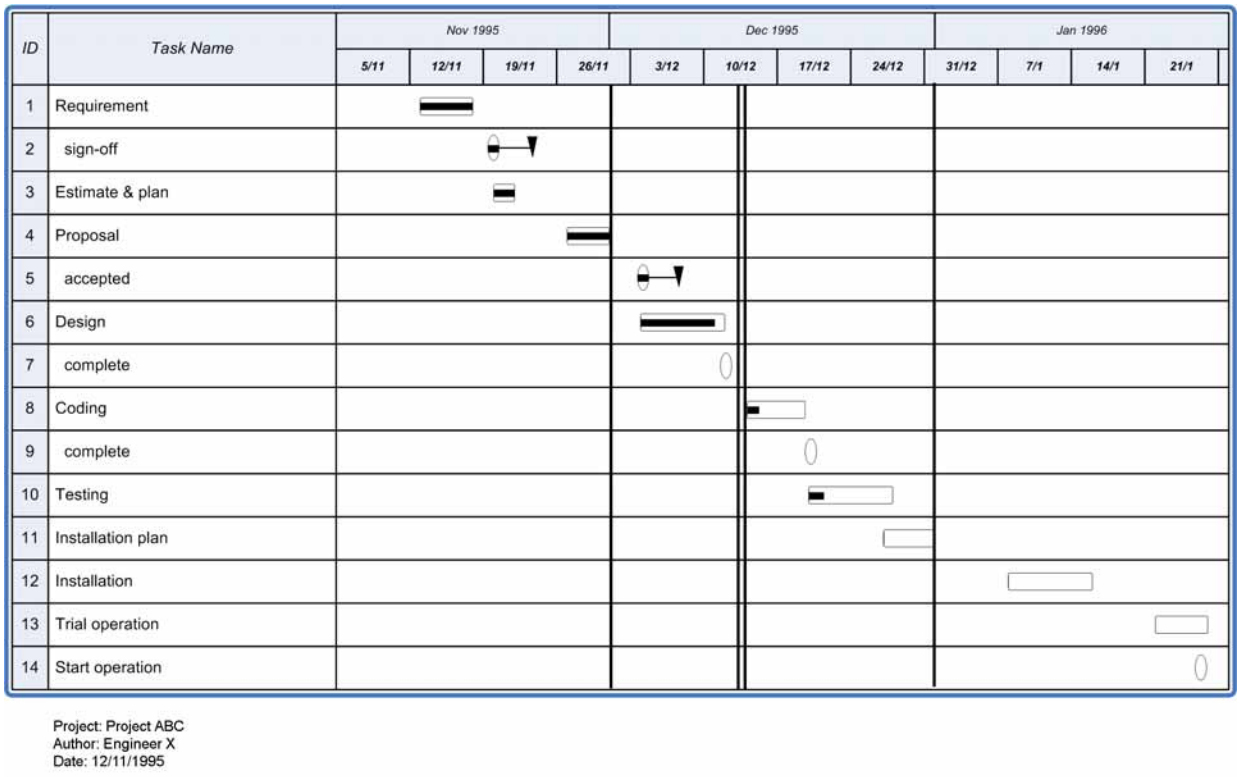
- Ô hình chữ nhật có một đường thẳng xuyên ngang từ đầu đến cuối ô biểu hiện công việc đã hoàn tất. Ví dụ, công việc lấy yêu cầu (ID1) được xem là đã hoàn tất.

- Ô hình chữ nhật có một đường thẳng xuyên ngang nhưng chỉ một phần không đến cuối ô biểu hiện công việc đã hoàn thành một phần. Ví dụ, việc thiết kế (ID6) dự kiến sẽ hoàn tất vào ngày 12/12 nhưng chỉ mới hoàn thành được 80% công việc mà thôi. Dĩ nhiên, con số 80% chỉ là 1 sự phỏng đoán.

- Ô hình chữ nhật có một đường thẳng xuyên ngang và nằm sau đường thẳng đứng đôi (đường biểu hiện ngày cập nhật tình hình) biểu hiện công việc được làm sớm hơn dự định. Ví dụ, kỹ sư X đã bắt đầu sớm một số việc lập trình và kiểm thử (ID 8 và 10).

- Các điểm mốc đã hoàn thành được điền với 1 mũi tên chỉ đến ngày chúng thật sự hoàn tất (ID2 và 5).

Với thông tin được cập nhật này, bạn có thể thấy rằng kỹ sư X trễ hạn so với lịch biểu đối với việc thiết kế (design), nhưng sớm hạn hơn trong việc lập trình (coding) và kiểm thử (testing). Tuy nhiên, cột "Design complete" vẫn chưa được hoàn tất, do đó dự án có thể được coi là chậm tiến độ so với thời gian biểu.



Hình 2.8.2 Biểu đồ Gantt của tình hình

Một số đề xuất trong việc theo dõi thời gian biểu

Rủi ro chính trong việc theo dõi thời gian biểu là bạn có thể dễ dàng đánh lừa chính mình. Bạn có thể có những nhận định lạc quan sai lệch bởi những điểm mốc không rõ ràng hoặc bởi việc thay đổi thời gian biểu thường xuyên. Một số bước giúp bạn tránh được việc tự lừa dối mình là:

- Đảm bảo rằng điểm mốc phải được định nghĩa và được viết ra rõ ràng.

- Không thay đổi thời gian biểu cho đến khi bạn lập 1 kế hoạch hoàn toàn mới.

- Khi nhận thấy tình hình công việc không theo đúng kế hoạch, không được thay đổi kế hoạch.

- Nếu bạn muốn đưa ra ngày hoàn thành mới, đặt thời gian biểu cũ xuống và ghi chú ngày mới vào với những đường đứt nét.

- Giữ lại bản thời khóa gian gốc và tất cả bản cập nhật.

Điều quan trọng cần nhớ là bạn đang đưa ra những tình hình công việc đối chiếu với thời gian biểu ban đầu khi mà thời gian biểu này chưa được chỉnh sửa. Đó là kế hoạch, và bạn đang đo lường những công việc của bạn dựa vào kế hoạch này. Nếu bạn thay đổi kế hoạch ban đầu, bạn sẽ không có gì để dựa vào đó mà đo lường, mà đánh giá. Đó là lý do tại

sao khi xem lại tình hình công việc, việc kiểm tra ngày mà thời gian biểu gốc đưa ra luôn là một ý kiến hay. Nếu nó quá gần với thời gian bạn xem xét, thì việc cập nhật không đưa ra được một sự đánh giá hiệu quả nào cho tiến trình dự án.

2.9 Lập kế hoạch cho dự án

2.9.1 Sự cần thiết phải lập kế hoạch cho dự án

Bản kế hoạch dự án mô tả các công việc cần làm và cách thực hiện các công việc này. Bản kế hoạch này định nghĩa các công việc chính yếu cần làm, gồm các ước lượng về thời gian, tài nguyên sử dụng, và đưa ra 1 "bộ khung" giúp ta quản lý và điều khiển công việc. Việc lập kế hoạch dự án cũng giúp ta nâng cao trình độ rất nhiều, cụ thể là nếu ghi nhận đúng, ta có thể sử dụng chúng để so sánh với hiệu suất làm việc thực sự, và từ đó nhìn ra được các lỗi ước lượng, giúp phát triển khả năng lập kế hoạch.

Việc lập kế hoạch vì vậy là một phần rất cần thiết của dự án. Dự án càng lớn, tầm quan trọng càng cao. Khi có nhiều kinh nghiệm, việc lập kế hoạch sẽ nhanh chóng và chính xác. Còn bây giờ, việc lập kế hoạch sẽ chỉ bao gồm ước lượng độ lớn của sản phẩm và thời gian để hoàn thành công việc. Sau khi thực hiện xong, ta sẽ ghi nhận lại độ lớn và thời gian hoàn thành thực tế để phân tích so sánh. Từ đó, bạn sẽ có thể ước lượng tốt hơn cho các công việc trong tương lai. Ví dụ, nếu ước lượng quá thấp, ta sẽ chỉnh cao lên, và ngược lại.

2.9.2 Bản tổng kết kế hoạch

Sinh viên		Ngày	
Chương trình		Chương trình #	
Người hướng dẫn		Ngôn ngữ	

Tóm tắt	**Kế hoạch**	**Thực tế**	***Đến ngày***
Phút/LOC			
LOC/Giờ			
Sai sót/KLOC			
Hiệu suất			
A/FR			
Kích thước chương trình (LOC)			
Tổng mới và thay đổi			
Kích thước tối đa			
Kích thước tối thiểu			

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	***Đến ngày***	***Đến ngày %***
Lên kế hoạch				
Thiết kế				
Cài đặt				

Xem lại mã				
Biên dịch				
Kiểm thử				
Tổng kết				
Tổng cộng				
Kích thước tối đa				
Kích thước tối thiểu				

Sai sót mắc phải	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Sai sót loại bỏ	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Bảng 2.9.1 Bản tổng kết kế hoạch đề án theo quy trình phần mềm cá nhân

Bảng trên đây là mẫu tóm tắt nên sử dụng để ghi lại các thông tin ước lượng và thực tế của dự án. Ta sẽ thảo luận chi tiết hơn về bảng này ở các phần sau. Hiện tại, bạn chỉ nên quan tâm đến phần không tô đậm.

Điền thông tin vào phần **Kế hoạch** trước khi bắt đầu dự án, và khi hoàn thành, điền vào phần **Thực tế**. Phần tiếp theo sẽ giải thích cách điền thông tin vào bản tóm tắt kế hoạch như thế nào.

2.9.2.1 Phần Tóm tắt

Phần tóm tắt trong mẫu thể hiện các tỉ số dùng để lập kế hoạch. Phần này cũng thể hiện các tỉ số thực tế sau khi hoàn thành công việc. Dòng trên cùng trong phần này là Phút/LOC (số phút cho mỗi dòng lệnh). Trong cột Kế hoạch, nhập vào tỉ số phút/LOC mà ta ước lượng trước khi thực hiện bằng cách sử dụng các tư liệu từ các chương trình đã làm trướcđó, hoặc từ bản ghi số công việc (Job Number Log). Tất nhiên, nếu không có tư liệu, thì bạn sẽ phải tự đưa ra 1 dự đoán nào đó.

Sinh viên	Sinh viên X	Ngày	7/10/96
Chương trình		Chương trình #	8
Người hướng dẫn	Thầy Z	Ngôn ngữ	Ada

Tóm tắt	**Kế hoạch**	**Thực tế**	***Đến ngày***
Phút/LOC	7.82	7.21	
LOC/Giờ	7.67	8.32	
Sai sót/KLOC			
Hiệu suất			
A/FR			

Kích thước chương trình (LOC)			
Tổng mới và thay đổi	26	19	
Kích thước tối đa	36		
Kích thước tối thiểu	18		

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	***Đến ngày***	***Đến ngày %***
Lên kế hoạch				
Thiết kế				
Cài đặt				
Xem lại mã				
Biên dịch				
Kiểm thử				
Tổng kết				
Tổng cộng	203	137		
Kích thước tối đa	282			
Kích thước tối thiểu	141			

Sai sót mắc phải	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Sai sót loại bỏ	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Bảng 2.9.2 Một ví dụ về lập kế hoạch dự án

Trừ khi bạn có một vài lí do nào đó, còn nếu không, hãy sử dụng tỉ số trung bình thu được từ chương trình gần nhất mà bạn đã lập trình (được lưu lại trong Job Number

Log). Bạn cũng có thể đưa ra các dự đoán khác nếu chương trình mới phức tạp hơn (đòi hỏi nhiều thời gian hơn) hoặc dễ hơn (hoặc tương tự các chương trình đã từng làm).

Dòng kế tiếp là LOC/Giờ (số dòng lệnh trên 1 giờ). Số LOC/Giờ được tính bằng cách lấy 60 chia cho Phút/LOC. Chỉ số LOC/Giờ này thường được các kĩ sư sử dụng để phân tích hiệu suất của việc phát triển sản phẩm.

2.9.2.2 Kích thước chương trình

Sinh viên: Sinh viên X Ngày: 10/7/96
Hướng dẫn: Thầy Z Lớp: CS1

Chương trình	LOC	Chức năng trước đó	Chức năng ước lượng	Nhỏ nhất	Trung bình	Lớn nhất
Case						
2	11	Biểu thức case đơn giản				
4	18	Biểu thức case trung bình	Biểu thức case phức tạp	12	17	24
Loops						
1	5	Vòng lặp do while đơn giản	Vòng lặp nhỏ	3	4	5
6	11	Vòng lặp repeat until trung bình				
Tính toán						
3	23	Tính toán nhiều				
Văn bản						
5	7	Chuỗi văn bản nhỏ	Chuỗi văn bản đơn giản	3	5	7
7	24	Chuỗi văn bản trung bình				
Ước lượng				18	26	36

Ghi chú: Chương trình này có 1 câu lệnh case khá lớn để kiểm tra các điều kiện đầu vào của 1 chuỗi văn bản. Câu lệnh case này vì vậy có kích thước xấp xỉ với kích thước câu lệnh case trong chương trình 4 (có thể lớn hơn). Sử dụng một vòng lặp đơn giản trong xử lý điều kiện đối với mỗi chuỗi đầu vào, và 1 bộ phân tích chuỗi đơn giản để xuất kết quả. Giả định rằng kích thước tối đa đối với vòng lặp và các hàm xử lý chuỗi xấp xỉ với kích thước của các chương trình trước đó, và câu lệnh case có thể lớn hơn. Ta cũng giả định kích thước tối thiểu bằng 1 nửa kích thước tối đa

Bảng 2.9.3 Ước lượng về kích thước chương trình của Sinh viên X

Phần *Kích thước chương trình* (LOC) chứa các thông tin ước lượng và thực tế về kích thước và "khoảng kích thước" (size range) của chương trình. Sử dụng các phương pháp ở phần 2.5 (kích thước sản phẩm), ước lượng kích thước của chương trình và nhập vào dòng *Tổng mới và Thay đổi* trong phần Kế hoạch. Để hoàn tất kế hoạch cho chương trình 9, sinh viên X đã dự đoán về kích thước như trong bảng trên.

Nguyên nhân đặt tên cho dòng này là Tổng mới và Thay đổi là do bạn sẽ chỉ ghi nhận số lượng dòng lệnh (LOC) mà bạn thật sự viết. Ban đầu, mỗi chương trình bạn viết nói chung đều hoàn toàn mới. Tuy nhiên, sau một vài chương trình, bạn sẽ bắt đầu sử dụng thư viện, hoặc dùng lại, hoặc sửa đổi một số dòng code từ các chương trình đã viết trước đó. Từ số lượng dòng lệnh mà bạn thực sự viết này, bạn có thể tính được tỉ số về hiệu suất Phút/LOC, từ đó có thể biết được thời gian để phát triển chương trình bằng cách nhân LOC Mới & Thay đổi với Phút/LOC.

Ví dụ, nếu bạn sao chép 25 LOC từ một chương trình cũ, đồng thời viết thêm 30 LOC mới, thì ta cũng sẽ chỉ ghi nhận 30 LOC vào phần Tổng Mới và Thay đổi. Nhưng, nếu bạn sao chép 25 LOC, và sửa chữa 7 LOC trong số này, thì giá trị Tổng Mới & Thay đổi sẽ là 37: 7 LOC được thay đổi và 30 LOC mới hoàn toàn.

Nguyên nhân của việc chỉ đếm LOC Mới & Thay đổi theo cách này là do số lượng dòng code bạn dùng từ các thư viện, hoặc từ các chương trình trước đó sẽ biến động rất nhiều. Đồng thời, thời gian bỏ ra (tính theo phút/LOC) để thêm các dòng code này cũng thường không đáng kể so với thời gian viết code mới, hoặc thay đổi các đoạn code đã có sẵn. Vì vậy, nếu bạn bỏ qua các LOC được sử dụng lại này, bạn có thể bỏ qua một phần nhỏ trong việc ước lượng, và ước lượng của bạn sẽ có thể chính xác hơn.

Kích thước tối đa và tối thiểu

Các giá trị kích thước tối đa và tối thiểu trong phần Kích thước chương trình (LOC) được tính toán theo phương pháp đã đề cập trong phần 2.5.

Các giá trị độ lớn tối đa và tối thiểu này rất có ích khi đánh giá khoảng thời gian bỏ ra khi để ước lượng việc xây dựng chương trình. Ví dụn, nếu bạn cần hoàn thành chương trình trước một ngày quy định nào đó, bạn hãy xem độ lớn tối đa với ý nghĩa là độ lớn mà chương trình có thể đạt đến. Nếu bạn ước lượng độ lớn có khả năng nhất là 26 LOC và độ lớn tối đa là 36 LOC, bạn có thể sẽ cho phép xảy ra chậm trễ trong kế hoạch trong trường hợp chương trình đã gần đạt đến giá trị tối đa.

Độ lớn tối thiểu được tính nhằm khuyến khích bạn nghĩ về độ lớn của chương trình dự định viết. Tuy nhiên, nói chung, các kế hoạch cơ bản cũng như thời hạn thường dựa vào các giá trị trung bình và giá trị tối đa.

2.9.2.3 Thời gian trong pha

Phần tiếp theo trong mẫu được gọi là thời gian trong pha (thời gian bỏ ra trong từng pha) vì nó sẽ được sử dụng làm tư liệu cho các giai đoạn trong quy trình phát triển phần mềm. Tuy nhiên, trong phần này, ta sẽ chỉ dùng khái niệm tổng thời gian phát triển chương trình.

Để ước lượng tổng thời gian phát triển cho một chương trình mới, ta cần ước lượng độ lớn của chương trình theo LOC và sau đó nhân với tỉ số Phút/LOC ở trên.

Ta cũng tính số thời gian tối thiểu và tối đa và nhập vào cột Kế hoạch bằng cách tương tự: lấy độ lớn tối thiểu và tối đa lần lượt nhân với Phút/LOC.

Tính toán và nhập các số liệu dự kiến này vào mẫu trước khi phát triển, và sau đó hoàn tất phần Thực tế khi đã làm xong. Cụ thể là ghi nhận thời gian làm việc, khi làm xong, nhập vào dòng Tổng, cột Thực tế. Sau đó, đếm số lượng dòng lệnh mới và có thay đổi trong chương trình đã hoàn chỉnh, nhập vào dòng LOC Mới & Thay đổi, đồng thời tính các giá trị thực tế đã bỏ ra Phút/LOC, LOC/Giờ.

2.9.3 Đánh giá độ chính xác

Mặc dù lúc nào ta cũng muốn các ước lượng là chính xác tuyệt đối, nhưng điều đó là không thể. Các dao động trong việc ước lượng chỉ có thể giảm dần dần, nhưng điều ta có thể làm ngay là ước lượng một cách khách quan, công bằng, nghĩa là, số lượng ước lượng thừa xấp xỉ số lượng ước lượng thiếu.

Ghi nhận lại các ước lượng này, nghiền ngẫm và rút ra bài học cho mình. Điều này sẽ giúp ta ước lượng tốt hơn. Khi đánh giá độ chính xác của các ước lượng, bạn sẽ biết được chúng ta được phép làm bao nhiêu ước lượng thừa, bao nhiêu ước lượng thiếu. Từ đó, tránh được rủi ro đưa ra các dự tính không thể đáp ứng được trong thực tế.

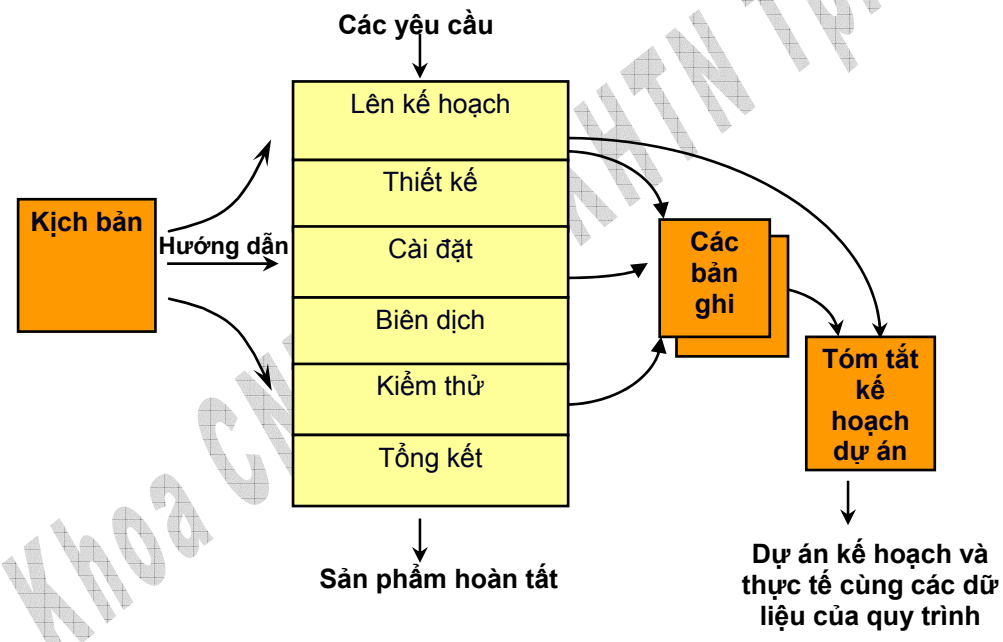
Chương 3. Các phương pháp luận trong PSP về quy trình quản lý sai sót [4]

3.1 Quy trình phát triển phần mềm

3.1.1 Tại sao chúng ta sử dụng quy trình

Một quy trình là một tập các bước đã được định nghĩa để thực hiện một công việc. Mỗi bước hay mỗi pha trong quy trình đều có những tiêu chuẩn đầu vào phải thoả trước khi bắt đầu pha. Tương tự như vậy, mỗi pha cũng có những tiêu chuẩn đầu ra cần được thỏa mãn trước khi hoàn tất pha. Các bước trong quy trình khi đó định nghĩa các nhiệm vụ và cách thực hiện chúng. Thiết kế và quản lý quy trình thì quan trọng trong công nghệ phần mềm bởi vì chất lượng một qui trình của kỹ sư sẽ phần lớn xác định được chất lượng và tính hiệu quả công việc của kỹ sư này. Mục đích của quy trình cá nhân được định nghĩa ở đây là để giúp bạn là một kỹ sư phần mềm hiệu quả hơn.

PSP là một "bộ khung" giúp cho các kỹ sư phần mềm đánh giá và cải thiện cách thức họ làm việc. Hai mục tiêu của PSP là giúp cho bạn phát triển các chương trình và chỉ ra cho bạn việc sử dụng quy trình có thể cải thiện cách làm việc của bạn như thế nào. Về sau, khi bạn phát triển các chương trình lớn hơn và phức tạp hơn, bạn có thể mở rộng PSP để giúp bạn thực hiện các công việc đó.



Hình 3.1.1 Dòng quy trình PSP

Khi một quy trình được mô tả đầy đủ thì nó được gọi là quy trình được định nghĩa. Một quy trình được định nghĩa thường bao gồm các kịch bản (script), biểu mẫu (form), khuôn dạng (template) và các chuẩn (standard). Một kịch bản quy trình là một tập hợp các bước được viết để người sử dụng đi theo khi sử dụng quy trình. Các biểu mẫu khác nhau, như là các bản ghi hay bản tóm tắt, được sử dụng để ghi lại hay lưu giữ những dữ liệu của dự án. Các yếu tố của PSP được thể hiện ở hình trên.

3.1.2 Kịch bản quy trình

Kịch bản ban đầu được thể hiện trong bảng dưới đây. Đây là cấp độ đầu tiên trong một số các cấp độ kich bản PSP. Trong các phần tiếp theo, kịch bản này được cải tiến thêm vào một số bước. Các pha của qui trình PSP được mô tả dưới đây và được tóm tắt trong bảng 3.1.1.

Lên kế hoạch: Đầu tiên nắm bắt các yêu cầu của dự án và sau đó hoàn chỉnh những phần Kế hoạch chưa được điền trong bản tóm tắt kế hoạch dự án. Cuối cùng, điền vào thời gian bạn bỏ ra để thực hiện việc lên kế hoạch này trong bản ghi thời gian.

Thiết kế: Vì một thiết kế không cần tỉ mỉ nên hãy suy nghĩ về logic chương trình trước khi bắt tay vào viết mã. Ghi nhận lại thiết kế trong biểu đồ, mã giả hay theo bất cứ một định dạng được định rõ nào. Cuối pha thiết kế, ghi nhận lại thời gian thiết kế trong bản ghi thời gian.

Viết mã (code): Thực thi thiết kế bằng cách cài đặt với ngôn ngữ chương trình đã chọn. Sử dụng một định dạng cài đặt nhất quán và theo những tiêu chuẩn được đưa ra. Cuối pha coding, ghi nhận lại thời gian cài đặt trong bản ghi thời gian.

Biên dịch: Biên dịch chương trình và sửa tất cả các lỗi tìm thấy. Tiếp tục biên dịch và chỉnh sửa lỗi cho đến khi chương trình không còn lỗi nữa. Tất cả thời gian trong pha này được tính là thời gian biên dịch, ngay cả khi sửa code cho đúng hay thay đổi thiết kế. Cuối pha biên dịch, ghi nhận lại thời gian biên dịch trong Bản ghi thời gian.

Kiểm thử: Chạy đủ test để đảm bảo rằng chương trình đáp ứng tất cả các yêu cầu và chạy tất cả các trường hợp test đều không gặp lỗi. Tất cả thời gian dùng trong pha này được tính là thời gian kiểm thử, bao gồm việc sửa mã cho đúng, thay đổi thiết kế và biên dịch lại. Cuối pha, ghi nhận lại thời gian kiểm thử trong Bản ghi thời gian.

Tổng kết (Postmortem): Hoàn tất các mục thực tế trong bản tóm tắt kế hoạch dự án. Bởi vì bạn cần phải ghi nhận lại thời gian tổng kết trước khi bạn thật sự kết thúc pha tổng kết, hãy hoàn tất càng nhiều công việc mà bạn có thể và khi đó, cho phép một vài phút

để thực hiện tính toán cuối cùng, điền vào thời gian tổng kết ước lượng. Dùng thời gian tổng kết ước lượng này để tính toán thời gian phát triển và tất cả các tính toán khác.

Mục đích	Hướng dẫn bạn trong việc phát triển những chương trình nhỏ
Tiêu chuẩn đầu vào	- Mô tả vấn đề - Bản tóm tắt kế hoạch dự án PSP - Dữ liệu về thời gian và kích thước thật sự của những chương trình trước - Bản ghi thời gian
1. Lên kế hoạch	- Ghi nhận những mô tả về chức năng của chương trình - Ước tính tổng số, tối đa, tối thiểu dòng lệnh cần thiết. - Xác định Phút/LOC - Xác định giá trị lớn nhất, nhỏ nhất và tổng cộng thời gian phát triển - Ghi nhận những dữ liệu kế hoạch trong bản tóm tắt kế hoạch dự án. - Ghi lại thời gian lên kế hoạch trong bản ghi thời gian.
2. Thiết kế	- Thiết kế chương trình - Ghi nhận lại thiết kế theo một định dạng chuẩn. - Ghi nhận lại thời gian thiết kế trong bản ghi thời gian
3. Cài đặt	- Thực thi thiết kế - Sử dụng dạng chuẩn để viết code. - Ghi nhận lại thời gian viết code trong bản ghi thời gian.
4. Biên dịch	- Biên dịch chương trình - Sửa tất cả các lỗi tìm thấy. - Ghi nhận lại thời gian biên dịch trong bản ghi thời gian.
5. Kiểm thử	- Kiểm thử chương trình. - Sửa tất cả các lỗi tìm thấy. - Ghi nhận lại thời gian kiểm thử trong bản ghi thời gian.
6. Tổng kết	- Hoàn tất bản tóm tắt kế hoạch dự án với thời gian và kích thước thực tế - Ghi nhận thời gian tổng kết trong bản ghi thời gian.
Tiêu chuẩn đầu ra	- Một chương trình đã được kiểm thử kỹ càng. - Một thiết kế đã được sưu liệu một cách chính xác. - Danh sách các chương trình hoàn tất. - Bản tóm tắt kế hoạch dự án đã hoàn tất. - Bản ghi thời gian đã hoàn tất.

Bảng 3.1.1 Kịch bản quy trình PSP

3.1.3 Điểm mốc và pha

Bằng việc định nghĩa ra các điểm mốc dự án có thể nhận ra một cách rõ ràng, bạn có thể lên kế hoạch tốt hơn. Lý do mà các dự án này tốt hơn là vì các điểm mốc cung cấp các điểm tham chiếu chính xác để đánh giá tình trạng của dự án khi bạn đang làm việc.

Qui trình phát triển phần mềm mở rộng ý tưởng điểm mốc từ một vài điểm cho đến toàn bộ các pha của qui trình. Với một qui trình được định nghĩa, mỗi pha đưa ra một kết quả và do đó việc hoàn tất một pha là một điểm mốc có thể đo được. Bằng cách sử dụng

một qui trình đã được định nghĩa, bạn sẽ có nhiều điểm mốc để giúp cho việc lên kế hoạch và theo dõi công việc.

3.1.4 Bản tổng kết các kế hoạch dự án cập nhật

Sinh viên		Ngày	
Chương trình		Chương trình #	
Người hướng dẫn		Ngôn ngữ	

Tóm tắt	**Kế hoạch**	**Thực tế**	***Đến ngày***
Phút/LOC			
LOC/Giờ			
Sai sót/KLOC			
Hiệu suất			
A/FR			
Kích thước chương trình (LOC)			
Tổng mới và thay đổi			
Kích thước tối đa			
Kích thước tối thiểu			

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	***Đến ngày***	***Đến ngày %***
Lên kế hoạch				
Thiết kế				
Cài đặt				
Xem lại mã				
Biên dịch				
Kiểm thử				
Tổng kết				
Tổng cộng				
Kích thước tối đa				
Kích thước tối thiểu				

Sai sót mắc phải	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Sai sót loại bỏ	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Bảng 3.1.2 Bản tổng kết kế hoạch đề án theo quy trình phần mềm cá nhân

Bản tổng kết kế hoạch dự án là một trong những biểu mẫu của qui trình PSP. Như trước đây, một số phần của bản tổng kết kế hoạch dự án được tô đậm. Các phần này lúc này chúng ta có thể lờ đi vì chưa sử dụng chúng. Để nhận ra sự thay đổi từ một cấp độ quy trình đến cấp độ kế tiếp, các phần được thêm vào được in ***nghiêng đậm***.

Mục đích	Mẫu này ghi nhận các thông tin ước lượng và thực tế của đề án
Đầu trang	Nhập các thông tin: - Tên và ngày hiện tại - Tên và mã số chương trình - Tên người hướng dẫn - Ngôn ngữ sử dụng để lập trình
Tóm tắt	
Phút/LOC	Trước khi phát triển: - Nhập giá trị Phút/LOC dự kiến cho đề án. Sử dụng tốc độ Đến ngày từ chương trình gần nhất trong bản ghi công việc hay bản tổng kết kế hoạch dự án. Sau khi phát triển: - Chia tổng thời gian phát triển cho độ lớn chương trình thực tế để có chỉ số Phút/LOC thực tế - Ví dụ, nếu dự án phát triển mất 196 phút và gồm 29 LOC, chỉ số Phút/LOC sẽ là 196/29=6.76
LOC/Giờ	Trước khi phát triển: - Tính LOC/Giờ dự kiến bằng cách lấy 60 chia cho Phút/LOC dự kiến Sau khi phát triển: - Để tính LOC/Giờ thực tế, lấy 60 chia cho Phút/LOC thực tế Ví dụ: với chỉ số Phút/LOC thực tế là 6.76, chỉ số LOC/Giờ thực tế là 60/6.76=8.88
Độ lớn chương trình (LOC)	Trước khi phát triển: - Nhập giá trị Tổng cộng, Tối đa và tối thiểu của LOC Mới & Thay đổi Sau khi phát triển: - Đếm và nhập giá trị LOC Mới & Thay đổi thực tế. - Với Đến ngày, cộng thêm LOC Mới và Thay đổi thực sự với LOC mới và Thay đổi Đến ngày của chương trình trước đó.
Thời gian bỏ ra ở từng giai đoạn	
Kế hoạch	Đối với Tổng thời gian phát triển (Total Development time), nhân LOC Mới & Thay đổi với Phút/LOC Đối với Thời gian tối đa, nhân độ lớn tối đa (Maximum size) với Phút/LOC. Đối với Thời gian tối thiểu, nhân độ lớn tối thiểu (Minimum size) với Phút/LOC. Từ bản tổng kết kế hoạch dự án của chương trình gần nhất, tìm giá trị Đến ngày % cho mỗi pha. Sử dụng Đến ngày % từ chương trình trước đó, tính toán thời gian kế hoạch cho mỗi pha.
Thực tế	Sau khi hoàn tất, nhập thời gian thực tế tính theo phút trong mỗi pha phát triển. Lấy dữ liệu này từ Bản ghi nhận thời gian
Đến ngày	Với mỗi pha, điền vào tổng thời gian thực tế và thời gian Đến ngày từ chương trình gần nhất.
Đến ngày %	Với mỗi pha, điền vào (thời gian Đến ngày * 100) / Tổng thời gian Đến ngày.

Bảng 3.1.3 Chỉ dẫn cho bản tổng kết kế hoạch

Phần Thời gian trong Pha của bản tổng kết kế hoạch dự án mới có một dòng cho mỗi pha của quy trình. Dòng này chứa thời gian kế hoạch và thực tế cho mỗi pha. Trong pha lên kế hoạch, điền vào tất cả các dữ liệu kế hoạch trong biểu mẫu này. Trong pha tổng kết, điền vào thời gian thực tế. Khi ghi nhận lại thời gian trong bản ghi thời gian, ghi chú vào phần chú thích bạn đang ở pha quy trình nào. Sau đó, trong khi tổng kết, điền các thời gian này vào Thời gian Thực tế trong cột Pha cho mỗi pha.

Trước khi bắt đầu một dự án, hoàn tất phần Kế hoạch của biểu mẫu tổng kết kế hoạch dự án như ở phần 2.9.2. Điều khác biệt duy nhất bây giờ là bạn cần phải ước lượng thời gian bỏ ra trong mỗi pha. Cách làm là phân phối tổng thời gian phát triển cho mỗi pha theo tỉ lệ mà bạn đã bỏ ra trong các dự án trước đây. Lần đầu bạn sử dụng cấp độ PSP này, bạn sẽ không có dữ liệu thực tế để làm điều này nên bạn phải đoán. Tuy nhiên, với các dự án tiếp theo, bạn có thể sử dụng dữ liệu từ các dự án trước này để ước lượng thời gian mỗi pha cho dự án mới. Đây là lý do sử dụng giá trị Đến ngày % trong bản tổng kết kế hoạch dự án.

Cột Đến ngày và Đến ngày % trong bản tổng kết kế hoạch dự án đưa ra một cách đơn giản để tính phần trăm phân phối thời gian phát triển cho mỗi pha. Cột Đến ngày chứa tổng thời gian bỏ ra trong mỗi pha cho tất cả các chương trình đã hoàn tất. Cột Đến ngày % chứa phần trăm phân phối của thời gian ở cột Đến ngày (Ví dụ trong phần 3.1.6 sẽ chỉ ra cách tính toán mục Đến ngày và Đến ngày %).

3.1.5 Một ví dụ về lên kế hoạch

Bảng 3.1.4 cho thấy sinh viên X hoàn tất một phần của biểu mẫu tổng kết kế hoạch cho chương trình 9. Sinh viên này sử dụng dữ liệu từ bản tổng kết kế hoạch dự án cho chương trình 8 ở bảng 3.1.5. Các mục kế hoạch trong bảng 3.1.4 được điền như sau:

- *Phút/LOC*. Khi lên kế hoạch cho chương trình 9, hãy xem Phút/LOC thật sự của chương trình trước, nghĩa là từ chương trình 8 ở bảng 3.1.5, và ta có được tốc độ là 7.21 Phút/LOC. Sau này, bạn sẽ không sử dụng những dữ liệu này nữa mà bạn sẽ sử dụng tốc độ trung bình của tất cả các chương trình được phát triển cho tới hiện tại (hay còn gọi là tốc độ Đến ngày, sẽ được thêm vào ở các phần sau).
- *LOC/giờ*. Sinh viên X tính ra là 60/7.21 = 8.32
- *Kích thước chương trình*. Sinh viên X ước lượng tổng LOC Mới và Thay đổi của chương trình (N), LOC tối đa và tối thiểu. Trong ví dụ của bảng 3.1.4, các kích thước này lần lượt là 23, 31 và 15 LOC.

Sinh viên	Sinh viên X	Ngày	21/10/96
Chương trình		Chương trình #	9
Người hướng dẫn	Thầy Z	Ngôn ngữ	Ada

Tóm tắt	**Kế hoạch**	**Thực tế**	***Đến ngày***
Phút/LOC	7.21		
LOC/Giờ	8.32		
Sai sót/KLOC			
Hiệu suất			
A/FR			

Kích thước chương trình (LOC)			
Tổng mới và thay đổi	23		
Kích thước tối đa	31		
Kích thước tối thiểu	15		

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	***Đến ngày***	***Đến ngày %***
Lên kế hoạch	5			
Thiết kế	0			
Cài đặt	74			
Xem lại mã				
Biên dịch	25			
Kiểm thử	52			
Tổng kết	10			
Tổng cộng	166			
Kích thước tối đa	224			
Kích thước tối thiểu	108			

Sai sót mắc phải	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Sai sót loại bỏ	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Bảng 3.1.4 Bản tổng kết kế hoạch đề án chương trình 9

- *Thời gian trong Pha - Tổng cộng*. Sử dụng kích thước ước lượng là 23 LOC và tốc độ 7.21 phút/LOC, thời gian phát triển sẽ là 7.21*23=166 phút.

- *Thời gian tối đa* = Phút/LOC * Kích thước tối đa = 7.21*31=224 phút.

- *Thời gian tối thiểu* = Phút/LOC * Kích thước tối thiểu = 7.21*15=108 phút.

Sinh viên	Sinh viên X	Ngày	7/10/96
Chương trình		Chương trình #	8
Người hướng dẫn	Thầy Z	Ngôn ngữ	Ada

Tóm tắt	**Kế hoạch**	**Thực tế**	***Đến ngày***
Phút/LOC	7.82	7.21	7.21
LOC/Giờ	7.67	8.32	8.32
Sai sót/KLOC			
Hiệu suất			
A/FR			
Kích thước chương trình (LOC)			
Tổng mới và thay đổi	26	19	19
Kích thước tối đa	36		
Kích thước tối thiểu	18		

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	***Đến ngày***	***Đến ngày %***
Lên kế hoạch	10	4	4	2.9
Thiết kế	19	0	0	0
Cài đặt	118	61	61	44.6
Xem lại mã				
Biên dịch	12	21	21	15.3
Kiểm thử	29	43	43	31.4
Tổng kết	15	8	8	5.8
Tổng cộng	203	137	137	100.0
Kích thước tối đa	282			
Kích thước tối thiểu	141			

Sai sót mắc phải	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Sai sót loại bỏ	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Bảng 3.1.5 Bản tổng kết kế hoạch đề án của chương trình 8

Với tổng cộng thời gian phát triển ước lượng là 166 phút, sử dụng Tốc độ Đến ngày % ở bảng 3.1.5 để ước lượng thời gian phát triển cho mỗi pha của chương trình 9. Sinh viên X tính toán các giá trị này như sau:

- *Lên kế hoạch*. Thời gian lên kế hoạch ước lượng là 2.9*166/100=4.48, hay khoảng 5 phút.

- *Thiết kế*. Thời gian thiết kế kế hoạch là 0*166=0.

- *Cài đặt*. Thời gian cài đặt kế hoạch là 44.6*166/100=74.04, hay 74 phút.

- *Biên dịch*. Thời gian biên dịch kế hoạch là 15.3*166/100=25.40, hay 25 phút.

- *Kiểm thử*. Thời gian kiểm thử kế hoạch là 31.4*166=52.12, hay 52 phút.

- *Tổng kết*. Thời gian tổng kết kế hoạch là 5.8*166=9.63, hay 10 phút.

Nhớ rằng với chương trình đầu tiên được phát triển với PSP, bạn sẽ không có dữ liệu trước đó để sử dụng như là một định hướng để ước lượng. Vì sinh viên X không có dữ liệu lịch sử trước đó khi lên kế hoạch chương trình 8 nên phải đoán. Bạn có thể thấy bằng cách so sánh thời gian kế hoạch và thực tế trong bảng 3.1.5, sự phân phối thời gian thực tế của sinh viên này hoàn toàn khác với dự đoán. Một ít dữ liệu có thể làm thay đổi nhiều trong việc lên kế hoạch. Với chương trình 9, thời gian kế hoạch của sinh viên X đã gần hơn nhiều với thời gian thực tế.

3.1.6 Một ví dụ về tính toán giá trị Đến ngày

Để hoàn tất cột Đến ngày ở bảng 3.1.6, sinh viên X cộng thời gian thực tế trên biểu mẫu này với thời gian Đến ngày của chương trình 8 ở bảng 3.1.5. Cậu cũng tính con số Đến ngày % trong bảng 3.1.6 bằng cách chia con số Đến ngày ở mỗi pha cho Tổng thời gian Đến ngày là 333 phút và nhân với 100. Các con số này được cậu tính toán như sau:

- LOC Mới và Thay đổi	LOC Mới và Thay đổi Đến ngày = 19+29=48
- Lên kế hoạch	Thời gian lên kế hoạch Đến ngày = 4+11=15
	Đến ngày %, lên kế hoạch = 100*15/333=4.5
- Thiết kế	Thời gian thiết kế Đến ngày = 0+12=12
	Đến ngày %, thiết kế = 100*12/333=3.6
- Cài đặt	Thời gian cài đặt Đến ngày = 61+85=146
	Đến ngày %, cài đặt = 100*146/333=43.9
- Biên dịch	Thời gian biên dịch Đến ngày = 21+28=49
	Đến ngày %, biên dịch = 100*49/333=14.7
- Kiểm thử	Thời gian kiểm thử Đến ngày = 43+49=92
	Đến ngày %, kiểm thử = 100*92/333=27.6
- Tổng kết	Thời gian tổng kết Đến ngày = 8+11=19
	Đến ngày %, tổng kết = 100*19/333=5.7

- Tổng cộng Tổng thời gian phát triển Đến ngày = 137+196=333

Sinh viên	Sinh viên X	Ngày	21/10/96
Chương trình		Chương trình #	9
Người hướng dẫn	Thầy Z	Ngôn ngữ	Ada

Tóm tắt	**Kế hoạch**	**Thực tế**	***Đến ngày***
Phút/LOC	7.21	6.76	
LOC/Giờ	8.32	8.88	
Sai sót/KLOC			
Hiệu suất			
A/FR			
Kích thước chương trình (LOC)			
Tổng mới và thay đổi	23	29	48
Kích thước tối đa	31		
Kích thước tối thiểu	15		

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	***Đến ngày***	***Đến ngày %***
Lên kế hoạch	5	11	15	4.5
Thiết kế	0	12	12	3.6
Cài đặt	74	85	146	43.9
Xem lại mã				
Biên dịch	25	28	49	14.7
Kiểm thử	52	49	92	27.6
Tổng kết	10	11	19	5.7
Tổng cộng	166	196	333	100.0
Kích thước tối đa	224			
Kích thước tối thiểu	108			

Sai sót mắc phải	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Sai sót loại bỏ	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Bảng 3.1.6 Bản tổng kết kế hoạch đề án của chương trình 9

Với dữ liệu từ chương trình 9, sinh viên X bây giờ có thể ước lượng thời gian bỏ ra trong mỗi pha của dự án kế tiếp. Tuy nhiên, để hiệu quả nhất, cậu nên lấy trung bình các

thời gian này qua một vài dự án. Cột Đến ngày và Đến ngày % là để thực hiện điều đó. Với chương trình 8 ở bảng 2.10.5, cột Đến ngày giữ thời gian thực tế mà sinh viên X bỏ ra cho dự án đó. Tuy nhiên, khi cậu phát triển chương trình 9, cậu cộng thời gian thực tế của chương trình 8 và 9 để lấy được thời gian Đến ngày mới. Giá trị Đến ngày % mới vì vậy đưa ra thời gian phân phối trung bình cho chương trình 8 và 9. Tương tự, khi bản tổng kết kế hoạch dự án của chương trình 10 được hoàn tất, nó sẽ chứa thời gian phân phối trung bình cho chương trình 8, 9, 10 và cứ như vậy.

Dần dần qua mỗi dự án, tổng thời gian phát triển sẽ biến đổi nhưng việc phân phối thời gian giữa các pha sẽ trở nên ổn định hơn. Điều này dĩ nhiên phụ thuộc vào chất lượng quy trình của bạn. Ví dụ, khi bạn bỏ ra quá nhiều thời gian để biên dịch và kiểm thử, thời gian pha được lên kế hoạch sẽ trở nên ít chính xác hơn vì lượng thời gian lớn và không dự đoán trước được bỏ ra để sửa chữa sai sót. Tuy nhiên khi lấy bình quân thời gian một vài chương trình, lượng thời gian trung bình bỏ ra trong biên dịch và kiểm thử sẽ không thay đổi nhiều lắm. Điều này có nghĩa là nó sẽ không thay đổi cho đến khi bạn thay đổi quy trình. Ban đầu khi sử dụng PSP, bạn sẽ bỏ ra tử 1/3 đến 1/2 thời gian để tìm và sửa lỗi trong biên dịch và kiểm thử. Khi bạn sử dụng các phương pháp PSP ở các phần sau, bạn sẽ giảm số sai sót tìm thấy trong biên dịch và kiểm thử, vì vậy sẽ giảm thời gian biên dịch và kiểm thử. Việc này sẽ tiết kiệm được thời gian phát triển, cải tiến tính có thể dự đoán của quy trình, các kế hoạch chính xác hơn và tạo ra được các chương trình tốt hơn.

3.2 Sai sót (defects)

Phần này đưa ra vấn đề về các sai sót phần mềm. Sai sót có thể gây ra các vấn đề nghiêm trọng cho người dùng sản phẩm phần mềm và việc tìm và sửa chúng thì lại đắt đỏ. Vì sai sót gây ra bởi lỗi của nhà phát triển, các kỹ sư cần phải hiểu sai sót họ mắc phải và học cách để quản lý chúng. Bước đầu tiên trong quản lý sai sót là tập hợp dữ liệu về sai sót mà bạn mắc phải trong chương trình. Với các dữ liệu này, bạn có thể nghĩ ra cách tốt hơn để tìm và sửa chúng.

Cho đến bây giờ, chúng ta chỉ nói về các phương pháp để quản lý chi phí và lịch biểu. Tuy nhiên đây mới chỉ là một nửa đoạn đường. Bắt đầu phần này sẽ chỉ ra nhu cầu cần phải sản xuất ra các sản phẩm phần mềm có chất lượng. Đầu tiên, chúng ta cần phải định nghĩa chất lượng là gì.

3.2.1 Chất lượng phần mềm là gì?

Chất lượng phần mềm ảnh hưởng rất lớn đến chi phí phát triển, kế hoạch chuyển giao sản phẩm và sự thoả mãn của khách hàng về sản phẩm. Vì chất lượng phần mềm quan trọng như vậy, đầu tiên chúng ta cần thảo luận chúng ta nói gì với từ *Chất lượng*. Chất lượng của sản phẩm phần mềm phải được định nghĩa bằng những thuật ngữ có ý nghĩa đối với người dùng sản phẩm. Một sản phẩm cung cấp được các khả năng quan trọng nhất đối với người dùng thì là một sản phẩm có chất lượng. Nhu cầu của người dùng thường được chỉ ra trong các tài liệu về yêu cầu. Cần phải nhớ rằng bạn không thể phát triển một chương trình có chất lượng cho đến khi bạn có các yêu cầu rõ ràng. Bạn có thể không bắt đầu với các yêu cầu rõ ràng nhưng bạn phải hiểu được yêu cầu trước khi bạn có thể hoàn tất.

3.2.2 Sai sót và chất lượng

Công việc của kỹ sư phần mềm là sản xuất ra các sản phẩm có chất lượng với chi phí mong đợi và đúng kế hoạch. Sản phẩm phần mềm phải đáp ứng được nhu cầu về chức năng của người dùng và thực hiện công việc của người dùng một cách đáng tin cậy và nhất quán. Thực hiện công việc là điểm mấu chốt. Các chức năng phần mềm thì quan trọng nhất đối với người dùng và các chức năng này sẽ không dùng được trừ khi phần mềm chạy. Để phần mềm chạy thì bạn phải loại bỏ các sai sót của nó. Vì vậy, có rất nhiều khía cạnh của chất lượng phần mềm nhưng mối quan tâm chất lượng đầu tiên của bạn phải là sai sót của nó. Điều này không có nghĩa sai sót là mối quan tâm duy nhất của bạn hay chúng là quan trọng nhất, mà chỉ khi bạn giải quyết được hầu hết các sai sót thì bạn mới có thể thỏa mãn được bất cứ mục tiêu nào khác của chương trình. Ngay cả khi bạn làm cho chương trình hoạt động được mà chúng vẫn còn một số lỗi thì chúng cũng sẽ không hoạt động trong các hệ thống lớn và sẽ không ai sử dụng chúng, bất chấp các chất lượng khác của chúng.

Lý do mà sai sót lại quan trọng đến thế là do con người vẫn hay phạm rất nhiều lỗi. Trong thực tế, ngay cả những lập trình viên kinh nghiệm vẫn thường mắc 1 sai sót cho mỗi 7 đến 10 dòng lệnh mà họ viết. Có thể họ sẽ tìm ra và chỉnh sửa hầu hết các lỗi trong lúc biên dịch và kiểm tra chương trình, nhưng thường vẫn có nhiều sai sót còn tồn tại ở sản phẩm cuối. Rõ ràng lúc này, nhiệm vụ ưu tiên hàng đầu của bạn là hiểu được các sai sót bạn mắc phải và tránh được càng nhiều càng tốt. Để làm được việc này, bạn cần thành thạo ngôn ngữ bạn sử dụng, cần hiểu sâu sắc về hệ thống hỗ trợ phát triển và phải nắm vững các

ứng dụng bạn sẽ phát triển. Những bước này và nhiều hơn nữa được yêu cầu để làm giảm bớt sai sót mà bạn mắc phải.

3.2.3 Sai sót là gì?

Thuật ngữ *sai sót* liên quan đến một cái gì đó sai trong chương trình, như lỗi cú pháp, viết sai chính tả, lỗi dấu câu hay một phát biểu chương trình không đúng. Sai sót có thể xuất hiện trong chương trình, thiết kế hay thậm chí trong yêu cầu, đặc tả hay trong các tài liệu khác. Sai sót có thề là một câu lệnh thừa, một câu lệnh không chính xác hay những phần chương trình bị lược bỏ. Nói tóm lại, sai sót là bất cứ thứ gì làm giảm đi khả năng của chương trình để đáp ứng hoàn toàn và có hiệu quả nhu cầu của người sử dụng. Đó là một thứ mà bạn có thể xác định, mô tả và đếm.

Những lỗi cài đặt đơn giản có thể sản sinh ra những sai sót nguy hiểm và khó tìm thấy. Ngược lại, có nhiều sai sót thiết kế phức tạp nhưng lại dễ dàng nhận thấy. Vì vậy độ phức tạp của sai sót thiết kế và ảnh hưởng của sai sót phần lớn là độc lập nhau. Ngay cả những lỗi thực thi tầm thường cũng có thể gây ra những vấn đề nghiêm trọng cho hệ thống.

Quan trọng là cần tách biệt vấn đề tìm hay nhận diện sai sót khỏi việc xác định nguyên nhân của chúng. Việc đếm hay ghi nhận các sai sót một cách đơn giản trong sản phẩm phần mềm thì không xác định được nguyên nhân hay đưa ra trách nhiệm. Tuy nhiên, sai sót lại có nguyên nhân. Có thể bạn đã viết sai chính tả một tham số, bỏ sót một dấu chấm câu hay gọi một hàm không đúng. Tất cả những lỗi này đều gây ra sai sót. Tóm lại, tất cả sai sót bắt nguồn từ lỗi của con người và nhiều lỗi mà kỹ sư phần mềm gây ra thì gọi là sai sót chương trình.

Lỗi là những gì không đúng mà con người gây ra và bất chấp việc khi nào hay ai tạo ra chúng, sai sót là yếu tố khuyết điểm của chương trình. Vì vậy, con người *tạo ra* lỗi còn chương trình thì *có* sai sót. Điều này có nghĩa là để giảm sai sót bạn mắc phải trong sản phẩm, bạn phải thay đổi những gì bạn làm. Tuy nhiên, để loại bỏ sai sót khỏi sản phẩm của bạn, thường thì bạn chỉ cần tìm chúng. Loại bỏ sai sót vì vậy là một quy trình dễ hơn là ngăn ngừa sai sót. Ngăn ngừa sai sót là một đề tài chính yếu và quan trọng đòi hỏi việc nghiên cứu toàn diện về toàn bộ quy trình phát triển phần mềm.

Vấn đề đặt ra là cần nhiều thời gian và tiền bạc để tìm và sửa sai sót phần mềm. Để ít mắc lỗi hơn, bạn phải cố gắng học hỏi từ những sai sót đã mắc phải trước đó, nhận diện được các lỗi tạo ra chúng, và học cách tránh lặp lại sai sót tương tự trong tương lai. Vì các sản phẩm khuyết điểm có thể rất đắt đỏ để kiểm thử, khó sửa chữa và có thể nguy hiểm khi

sử dụng, điều quan trọng là bạn phải học cách giảm thiểu sai sót bạn để lại trong sản phẩm. Tài liệu này sẽ giúp bạn làm điều đó.

Một số phần trăm của sai sót trong một chương trình có thể có các hậu quả không lường trước được. Nếu chúng ta có thể biết được chúng là những sai sót nào thì khi đó chúng ta chỉ cần sửa chúng và không còn lo lắng gì nữa. Không may là không có cách nào để làm điều này và bất cứ sai sót bị bỏ sót nào đều có thể gây nên hậu quả nghiêm trọng. Bây giờ có thể sai sót không là vấn đề nghiêm trọng với bạn, nhưng không sớm thì muộn. Vì vậy quan trọng là bạn phải học cách quản lý sai sót ngay từ bây giờ để bạn có thể sẵn sàng khi thật sự cần phải sản xuất ra các chương trình chất lượng cao.

Kỹ sư phần mềm - những người viết chương trình - là những người có thể tìm và vá lỗi tốt nhất. Vì vậy quan trọng là kỹ sư phần mềm cần phải có trách nhiệm cá nhân cho chất lượng của chương trình mà họ viết. Tuy nhiên, việc học cách để viết các chương trình không-lỗi là một thách thức lớn. Đó không phải là điều mà bất cứ ai cũng có thể thực hiện nhanh chóng và dễ dàng. Cần phải có dữ liệu, các kỹ thuật hiệu quả và kỹ năng. Sử dụng các phương pháp được mô tả ở đây bạn có thể phát triển và mài dũa khả năng sản xuất ra các chương trình chất lượng cao. Sau cùng, nếu bạn không cố gắng để thực hiện một công việc không có lỗi thì sẽ chẳng bao giờ bạn có thể làm được cả.

3.2.4 Các loại sai sót

Để phân tích các sai sót thì cách hữu ích là nên phân loại chúng. Tài liệu này phân sai sót ra thành 10 loại chung. Bằng việc phân loại này, bạn sẽ nhanh chóng tìm ra loại nào gây ra vấn đề nhất để tập trung vào việc ngăn chặn và loại bỏ nó. Tất nhiên, đây là điều then chốt của quản lý sai sót. Tập trung giải quyết một số ít sai sót gây rắc rối nhất. Một khi đã kiểm soát được chúng, hãy tiếp tục với các sai sót tiếp theo, và cứ như vậy.

Trong bảng sau là danh sách 10 loại sai sót, xuất phát từ thành quả công việc của Chillarege cùng các đồng sự khi nghiên cứu ở IBM. Ông đã nghiên cứu về các sai sót trên một diện rộng các sản phẩm IBM và nhận ra các loại chính của chúng. Các loại này cũng được nhận thấy là có ích cho PSP.

Loại số	Tên loại	Mô tả
10	Sưu liệu (documentation)	Lời bình, thông điệp
20	Cú pháp	Lỗi chính tả, lỗi chấm câu, lỗi do gõ phím, các định dạng chỉ thị, lệnh
30	Xây dựng, đóng gói	Quản lý thay đổi, thư viện, điều khiển các phiên bản
40	Chỉ định	Khai báo, trùng tên, phạm vi, giới hạn
50	Giao diện (Interface)	Lời gọi hàm, các tham chiếu, nhập/xuất (I/O), định dạng người dùng
60	Kiểm tra (Checking)	Thông điệp báo lỗi, kiểm tra không thích hợp
70	Dữ liệu	cấu trúc, nội dung
80	Chức năng	Logic, con trỏ, vòng lặp, đệ qui, tính toán, sai sót chức năng
90	Hệ thống	Cấu hình, sự định giờ, bộ nhớ
100	Môi trường	Thiết kế, biên dịch, kiểm tra, những vấn đề hệ thống hỗ trợ khác

Bảng 3.2.1 Chuẩn các loại sai sót

Trong bảng trên, các loại sai sót dựa vào các loại vấn đề có đặc điểm chung. Ví dụ, loại 20 – sai sót cú pháp đề cập đến tất cả những gì được tạo ra nhưng không đúng đặc tả của ngôn ngữ lập trình. Có thể đó là thiếu ";", câu lệnh *if* không đúng, lỗi chính tả hay khai báo sai. Sai sót loại 20 là bất cứ sai sót nào dẫn đến cú pháp chương trình không đúng, bất chấp việc sai sót đó bị gây ra hay được tìm thấy như thế nào. Một loại sai sót khác như là 80 – chức năng và ví dụ cho loại này như là một vòng lặp *do-while* với logic sai trong điều kiện while.

Các loại sai sót trong bảng được sắp xếp theo độ phức tạp của nguyên nhân gây ra có thể có. Ví dụ, loại 10, 20, 30 thường bắt nguồn từ các lỗi hay sơ suất đơn giản. Tuy nhiên, loại 80, 90, 100 lại thường liên quan đến vấn đề thiết kế phức tạp hơn hay là các vấn đề về hệ thống.

Thay vì cải tiến từng 10 loại sai sót PSP này thành các loại con, hãy chờ cho đến khi bạn đã tập hợp được dữ liệu ít nhất 100 sai sót của một số chương trình. Khi đó, bạn có thể thấy được chỗ nào cần chi tiết thì tốt hơn và cần thêm thông tin cụ thể nào. Ví dụ, bạn có thể chia loại 20 – sai sót cú pháp thành các loại con như loại 21 cho sai sót về ";", 22 cho các sai sót dấu câu khác, 23 cho vấn đề về biểu thức Boolean, 24 cho các dạng chỉ thị lệnh sai, v.v…

3.2.5 Hiểu được các sai sót

Bước đầu tiên để quản lý được các sai sót là phải hiểu được chúng. Để làm được điều này, bạn cần tập hợp các dữ liệu sai sót. Khi đó bạn có thể hiểu được các lỗi này và

luận ra được cách tránh chúng. Ngoài ra bạn có thể luận ra được cách tốt nhất để tìm, sửa hoặc thậm chí là ngăn chặn các sai sót mà bạn vẫn còn mắc phải.

Để tập hợp được dữ liệu sai sót trong chương trình, cần làm theo những bước sau:

- Giữ lại ghi nhận về mỗi sai sót bạn tìm thấy trong chương trình.
- Ghi nhận lại thông tin đầy đủ cho mỗi sai sót để sau này bạn có thể hiểu nó.
- Phân tích những dữ liệu này để thấy được loại sai sót nào gây ra hầu hết các vấn đề.
- Nghĩ ra cách để tìm ra sai sót và chỉnh sửa những sai sót này.

Những sai sót bạn mắc phải và tìm thấy trong chính chương trình của mình chỉ là một phần của vấn đề. Một lúc nào đó bạn sẽ cần phải tìm hiểu về các sai sót mà người khác tìm thấy trong chương trình của bạn. Vì các sai sót này đã thoát khỏi sự ngăn ngừa sai sót và nỗ lực tìm kiếm của bạn nên chúng rất quan trọng trong việc hiểu và chỉ ra các điểm yếu trong quy trình cá nhân của bạn. Khi quy trình cá nhân của bạn được cải thiện các sai sót bị bỏ sót này cuối cùng sẽ là nguồn dữ liệu chính cho việc cải tiến cá nhân của bạn.

3.2.6 Bản ghi ghi chép sai sót (Defect Recording Log)

Loại sai sót	
10 Sưu liệu	60 Kiểm tra
20 Cú pháp	70 Dữ liệu
30 Xây dựng, đóng gói	80 Chức năng
40 Chỉ định	90 Hệ thống
50 Giao diện	100 Môi trường

Sinh viên _____ Ngày _____

Người hướng dẫn _____ Chương trình # _____

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa

Mô tả

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa

Mô tả

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa

Mô tả

Bảng 3.2.2 Bản ghi ghi chép sai sót

Bản ghi ghi chép sai sót được thiết kế để giúp cho việc thu thập dữ liệu sai sót. Sử dụng bản ghi này để tập hợp dữ liệu sai sót cho mỗi chương trình mà bạn viết. Mô tả mỗi

sai sót đủ chi tiết để sau này bạn có thể hiểu nó. Sau khi hoàn tất mỗi chương trình, phân tích dữ liệu để thấy được ở chỗ nào bạn mắc phải và loại bỏ sai sót và loại sai sót nào gây rắc rối nhất.

Mục đích	Biểu mẫu này lưu giữ những dữ liệu về các sai sót mà bạn tìm thấy và chỉnh sửa. Sử dụng các dữ liệu này để hoàn tất bản tổng kết kế hoạch dự án
Tổng quát	Ghi nhận tất cả các sai sót xem lại, biên dịch, kiểm thử trong bản ghi này. Ghi nhận mỗi sai sót một cách riêng biệt và hoàn chỉnh.
Đầu đề	Ghi những thông tin sau: - Tên - Ngày lập - Tên người hướng dẫn - Số của chương trình
Ngày	Ghi nhận lại ngày mà sai sót được phát hiện
Số	Đánh số mỗi sai sót Với mỗi chương trình, sử dụng chuỗi số liên tiếp bắt đầu bằng 1 (hay 001, v.v…)
Loại	Ghi nhận loại sai sót từ danh sách các sai sót trong bảng trên. Sử dụng sự phán đoán của bạn trong việc chọn loại sai sót nào thích hợp
Mắc phải	Ghi nhận lại pha mà trong đó sai sót bị mắc phải
Loại bỏ	Ghi nhận lại pha mà trong đó sai sót được loại bỏ
Thời gian sửa chữa	Ước tính hay đo thời gian cần thiết để tìm và chỉnh sửa lỗi.
Sai sót sửa chữa	Bạn có thể bỏ qua mục này trong lúc này. Nếu bạn mắc phải sai sót này trong khi đang chỉnh sửa sai sót khác, ghi nhận lại số sai sót đã chỉnh sửa sai này
Mô tả	Viết mô tả ngắn gọn về sai sót. Mô tả đủ rõ ràng để sau này nó nhắc lại cho bạn nhớ về lỗi gây ra sai sót và tại sao bạn mắc phải.

Bảng 3.2.3 Các chỉ dẫn bản ghi ghi chép sai sót

Phần sau sử dụng bản ghi ghi chép sai sót ví dụ trong bảng 3.2.4 để hướng dẫn cách hoàn tất bản ghi:

1. Khi bắt đầu phát triển một chương trình, lấy vài trang bản ghi ghi chép sai sót và điền thông tin đầu đề của trang đầu tiên.

2. Khi bạn tìm thấy sai sót đầu tiên, ghi nhận lại số thứ tự của sai sót trong bản ghi, nhưng không ghi nhận các thông tin khác cho đến khi bạn chỉnh sửa được sai sót. Khi sinh viên X thử biên dịch chương trình 10 lần đầu tiên, trình biên dịch hiển thị hơn một tá thông điệp lỗi. Mặc dù cậu không biết ngay vấn đề là gì, nhưng cậu biết có ít nhất một lỗi sai. Vì vậy cậu ghi lại thời gian và điền 1 dưới mục *Số* ở dòng đầu tiên của bản ghi sai sót vì đây là lỗi đầu tiên của chương trình 10. Các con số này về sau sẽ giúp bạn trong việc phân tích dữ

liệu sai sót. Trong các chương trình lớn hơn, số thứ tự sai sót được dùng để theo dõi vấn đề sửa lỗi không đúng và giúp ngăn chặn sai sót.

Loại sai sót	
10 Sưu liệu	60 Kiểm tra
20 Cú pháp	70 Dữ liệu
30 Xây dựng, đóng gói	80 Chức năng
40 Chỉ định	90 Hệ thống
50 Giao diện	100 Môi trường

Sinh viên	Sinh viên X	Ngày	28/10/96
Người hướng dẫn	Thầy Z	Chương trình #	10

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
28/10	1	20	Cài đặt	Biên dịch	1 phút	

Mô tả Thiếu ";"

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
	2	20	Cài đặt	Biên dịch	1 phút	

Mô tả Thiếu ";"

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
	3	40	Thiết kế	Biên dịch	1 phút	

Mô tả Kiểu sai do RSH của toán tử nhị phân, phải chuyển kiểu integer sang float

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
	4	40	Cài đặt	Biên dịch	1 phút	

Mô tả Kiểu sai do RSH, hằng phải là 0.0 chứ không phải 0

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
	5	40	Cài đặt	Biên dịch	1 phút	

Mô tả Kiểu sai do RSH, phải chuyển kiểu integer sang float

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
	6	40	Thiết kế	Biên dịch	7 phút	

Mô tả Số mũ phải là integer, tìm hiểu lại và sử dụng thư viện math cho hàm sqrt.
Số nguyên thì không được tính toán đúng đắn.

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
	7	80	Cài đặt	Kiểm thử	14 phút	

Mô tả Đáp án (std. dev.) không đúng – viết code không đúng, viết trừ thay vì chia

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
	8	80	Cài đặt	Kiểm thử	28 phút	

Mô tả Vòng lặp không kết thúc do số mũ âm, quên đổi dấu khi trừ.

Bảng 3.2.4 Bản ghi ghi chép sai sót

3. Sử dụng một dòng riêng biệt cho từng sai sót. Không nên nhóm các sai sót lại trên cùng một dòng.

4. Ghi nhận ngày sai sót được phát hiện. Nếu bạn phát hiện nhiều lỗi trong cùng 1 ngày, bạn có thể để trống các mục ngày tiếp theo cho đến mục đầu tiên của ngày kế tiếp. Trong bảng 3.2.4, sinh viên X tìm thấy tất cả các sai sót vào ngày 28/10, vì vậy cậu không cần nhập lại ngày vì nó được ngầm hiểu là "như trên" cho đến khi có sự thay đổi.

5. Sau khi sửa lỗi xong, ghi nhận loại sai sót. Có thể bạn cảm thấy lúng túng về việc loại sai sót nào thì phù hợp, hãy sử dụng óc phán đoán của bạn. Đừng phí thời gian lo lắng về việc loại sai sót nào mới chính xác nhưng bạn cũng cần nhất quán một cách hợp lý. Về sai sót 1 trong bảng 3.2.4, sinh viên X phát hiện ra rằng vấn đề là thiếu dấu ";". Khi đã giải quyết xong vấn đề, cậu ghi nhận lại con số 20 dưới mục *Loại* cho sai sót 1.

6. *Ghi nhận pha của quy trình khi bạn mắc phải sai sót.* Không phải điều này lúc nào cũng rõ ràng nhưng nó không nên là vấn đề cho các chương trình nhỏ. Sử dụng phán đoán tốt nhất của bạn và đừng phí thời gian lo lắng về nó. Trong ví dụ này, sinh viên X tự tin rằng cậu đã phạm sai sót khi thiếu ";" khi đang viết chương trình, vì vậy cậu ghi nhận từ "*cài đặt*" dưới mục *Mắc phải*.

7. *Ghi nhận lại pha trong qui trình mà bạn loại bỏ được sai sót* Đây thường là pha khi bạn tìm thấy sai sót. Ví dụ, ở đây, với sai sót 1, sinh viên X đang ở pha biên dịch khi cậu tìm thấy và chỉnh sửa sai sót, vì vậy cậu ghi nhận từ "*biên dịch*" ở mục *Loại bỏ.*

8. Với thời gian chỉnh sửa sai sót, ước lượng thời gian từ khi bạn bắt đầu biết được và bắt đầu làm việc với sai sót cho đến khi bạn hoàn tất việc chỉnh sửa và kiểm tra nó. Khi bắt đầu chỉnh sửa sai sót 1, sinh viên X ghi lại thời gian trên đồng hồ của mình. Khi cậu đã sửa xong sai sót và kiểm tra để biết chắc chắn nó được sửa đúng, một lần nữa cậu lại xem đồng hồ và thấy cấu chỉ bỏ ra khoảng 1 phút. Thông thường, với các sai sót biên dịch, thời gian sửa chữa sẽ chỉ là 1 phút hoặc hơn chút xíu. Tuy nhiên, với các sai sót tìm thấy trong pha kiểm thử, việc sửa sai có thể chiếm nhiều thời gian hơn. Bạn có thể sử dụng đồng hồ hay đồng hồ bấm giờ để đo thời gian sửa chữa, nhưng với các sửa chữa ngắn thì việc bạn phán đoán thường sẽ thích hợp hơn.

9. Mục sai sót sửa chữa là các sai sót mắc phải khi đang chỉnh sửa các sai sót khác.

10. Viết một mô tả ngắn gọn về các sai sót trong phần mô tả. Mô tả càng ngắn và càng đơn giản càng tốt nhưng phải rõ ràng. Ví dụ, chỉ cần ghi nhận một dấu ";" để chỉ việc thiếu ";". Với một sai sót phức tạp hơn, hãy viết một vài dòng nếu cần. Với sai sót 1, sinh viên X

chỉ đơn giản ghi "thiếu ;". Với hầu hết các sai sót trong bảng 3.2.4, cậu phải đưa ra một mô tả chi tiết hơn. Tuy nhiên, bởi vì mô tả này chỉ để cho bạn sử dụng nên không cần phải viết gì nhiều hơn cần thiết để nhắc cho bạn nhớ về vấn đề.

Người ta thường lúng túng về loại sai sót và nghĩ rằng nên có một loại riêng dành cho việc hiểu sai và nhầm lẫn. Ví dụ, nếu bạn không hiểu yêu cầu hoặc không quen với môi trường phát triển, bạn sẽ phạm nhiều sai sót. Vấn đề này thì quan trọng, nhưng nó lại liên quan đến *nguyên nhân* sai sót. Vì chỉ có *loại* của sai sót là liên quan nên chỉ có 2 câu hỏi: Có phải có gì sai trong sản phẩm và nếu như vậy thì loại của sai sót là gì? Vì vậy, mặc dù hiểu nguyên nhân thì rất cần thiết để ngăn chặn sai sót nhưng loại của sai sót chỉ mô tả điều gì sai trong sản phẩm mà thôi.

3.2.7 Đếm sai sót

Việc xác định sai sót có vẻ như rõ ràng, nhưng nó không như vậy. Ví dụ, trong biên dịch, chỉ đếm các thay đổi mà bạn sửa. Có nghĩa là nếu trình biên dịch đưa ra 10 thông báo lỗi cho 1 lỗi thiếu ";", và lỗi thiếu ";" là lỗi duy nhất. Vì vậy, ghi nhận 1 sai sót trong bản ghi ghi chép sai sót cho một sửa chữa chương trình, bất chấp số thông báo lỗi của trình biên dịch.

Tương tự, khi bạn phát hiện một sai sót thiết kế trong khi đang viết code thì đó là 1 sai sót thiết kế. Tuy nhiên, trong thiết kế, bạn có thể hay thay đổi ý định về cách làm điều gì đó. Nếu bạn đang sửa một sai sót trong yêu cầu hay đặc tả thì đó là sai sót yêu cầu hay đặc tả. Tuy nhiên, nếu bạn đang nghĩ một cách tốt hơn để thực hiện thiết kế thì đó không phải là một sai sót. Bạn cũng sẽ thường bắt và sửa lỗi ngay khi bạn tạo ra nó. Những sự điều chỉnh như thể là một phần của sáng tạo tự nhiên, không phải là sai sót. Bí quyết là bạn ghi nhận các sai sót mà bạn đã để lại trong sản phẩm khi bạn đã hoàn tất thiết kế ban đầu hay cài đặt.

Ví dụ, nếu bạn nhập một dòng lệnh và ngay lập tức thấy sai và sửa ngay tên biến viết sai thì đây không phải là sai sót. Nhưng nếu bạn đã hoàn tất cài đặt chương trình và về sau nhận ra việc viết sai tương tự như thể thì đó là một sai sót và hãy đếm chúng. Vì vậy, nếu cách làm thông thường của bạn là kiểm tra mỗi dòng lênh ngay sau khi bạn viết ra thì những sai sót kiểu này không được tính.

Hãy bắt đầu việc đếm sai sót bất cứ khi nào bạn hoàn tất một pha của sản phẩm hay một phần của sản phẩm. Ví dụ, sau pha thiết kế, bạn sẽ đếm tất cả các sai sót thiết kế. Tuy nhiên, giả sử rằng bạn đang cài đặt 2 thủ tục chương trình. Sau khi cài đặt thủ tục đầu, bạn

quyết định viết thủ tục thứ 2 trước khi bắt đầu biên dịch. Đang cài đặt cho thủ tục thứ 2 thì bạn nhận ra rằng bạn đã đặt tên một tham số sai trong thủ tục đầu. Đây là một sai sót cho dù bạn đang ở pha cài đặt, vì bạn đã hoàn tất việc cài đặt cho thủ tục thứ nhất.

Chú ý rằng trong tài liệu này không đòi hỏi bạn phải đếm sai sót tìm thấy trong pha thiết kế hay cài đặt. Ban đầu thì quan trọng là phải tập trung vào các sai sót tìm thấy trong biên dịch hay kiểm thử. Một khi bạn đã quen với việc thu thập dữ liệu sai sót, bạn sẽ biết rõ hơn tại sao các dữ liệu sai sót này là cần thiết. Khi đó bạn có thể muốn tìm hiểu nhiều hơn về sai sót bạn tạo ra và chỉnh sửa trong pha thiết kế và cài đặt. Vì bạn sẽ mắc hầu hết các sai sót trong khi thiết kế và cài đặt nên đây là các pha mà bạn phải xem xét để hiểu nguyên nhân sai sót và biết cách ngăn chặn chúng. Tuy nhiên, vào lúc này, hãy bắt đầu chỉ với các sai sót mà bạn tìm thấy trong biên dịch và kiểm thử.

3.2.8 Sử dụng bản ghi ghi chép sai sót

Tại sao bạn phải đếm sai sót? Khi bạn thu thập dữ liệu về sai sót, hãy nhớ tại sao bạn lại đang làm điều đó:

Để cải tiến việc lập trình của bạn. Các dữ liệu sai sót là để giúp bạn cải tiến cách bạn lập trình. Trong khi việc phòng tránh các sai sót thì dễ nhưng bạn không thể quản lý sai sót nếu bạn không hiểu chúng. Điều này có nghĩa bạn phải thu thập dữ liệu chính xác về chúng.

Để giảm số sai sót trong chương trình của bạn. Ai cũng mắc sai sót, nhưng bằng cách sử dụng các phương pháp đúng đắn và cẩn thận, bạn có thể giảm số lượng sai sót mắc phải này.

Để tiết kiệm thời gian. Sai sót tạo ra thêm các sai sót. Sai sót càng tồn tại lâu trong chương trình thì càng cần nhiều thời gian hơn để tìm và càng khó để sửa chữa. Vấn đề về yêu cầu dẫn đến thiết kế sai, lỗi thiết kế gây ra lỗi thực thi, lỗi thực thi làm chương trình có sai sót. Đó là lý do tại sao việc loại bỏ sai sót càng sớm càng tốt sau khi bạn mắc phải lại quan trọng như vậy.

Để tiết kiệm tiền bạc. Sai sót thì đắt đỏ. Sau khi kiểm thử đơn vị, chi phí tìm và sửa lỗi tăng lên 10 lần với mỗi pha kiểm thử hay bảo trì sau đó.

Để thực hiện trách nhiệm công việc của bạn. Sai sót là do kỹ sư mắc phải do đó trách nhiệm của họ là tìm và sửa lỗi.

3.2.9 Bản tổng kết kế hoạch đề án cập nhật

Đến lúc này, bạn sẽ hoàn tất thêm các mục sau trong bản tổng kết kế hoạch đề án với các chỉ dẫn sau:

Sai sót mắc phải thực tế	Sau khi phát triển, tìm và điền số lượng sai sót thực tế mắc phải trong mỗi pha
Đến ngày	Với mỗi pha, nhập vào tổng số sai sót thực tế và sai sót Đến ngày từ chương trình gần nhất.
Đến ngày %	Với mỗi pha, nhập vào 100*(Sai sót Đến ngày của pha đó)/(Tổng sai sót Đến ngày)
Sai sót loại bỏ thực tế	Sau khi phát triển, tìm và điền số lượng sai sót thực tế loại bỏ trong mỗi pha
Đến ngày	Với mỗi pha, nhập vào tổng số sai sót thực tế và sai sót Đến ngày từ chương trình gần nhất.
Đến ngày %	Với mỗi pha, nhập vào 100*(Sai sót Đến ngày của pha đó)/(Tổng sai sót Đến ngày)

Bảng 3.2.5 Một số chỉ dẫn cập nhật cho bản tổng kết kế hoạch

Trong pha tổng kết, xem lại bản ghi sai sót và đếm số lượng sai sót mắc phải trong mỗi pha. Từ bản ghi ghi chép sai sót ở bảng 3.2..4, sinh viên X đầu tiên tính sai sót 3 và 6 mắc phải trong thiết kế do vậy cậu điền 2 dưới cột *Thực tế* ở dòng *thiết kế* của bảng 3.2.6. Sáu sai sót còn lại đều mắc phải trong pha cài đặt, do vậy cậu điền 6 vào dòng *cài đặt.* Tổng cộng là có 8 sai sót mắc phải.

Kế đến, đếm số sai sót loại bỏ được trong mỗi pha. Sinh viên X đếm được 6 sai sót loại bỏ trong biên dịch và 2 trong kiểm thử nên cậu điền 6 và 2 ở các dòng này của phần sai sót được loại bỏ. Một lần nữa, tổng cộng là 8.

Sau khi ghi nhận số sai sót mắc phải và loại bỏ, hoàn tất cột Đến ngày và Đến ngày % tương tự cách bạn đã làm với dữ liệu thời gian.

Với dữ liệu Đến ngày %, thật đáng ngạc nhiên là các kỹ sư có thể ước lượng số sai sót mà họ mắc phải và loại bỏ một cách chính xác như thế nào. Thói quen chi phối các lỗi lầm của chúng ta, do đó khi mà chúng ta còn chưa thay đổi các thói quen này, chúng ta sẽ còn tiếp tục mắc những sai sót tương tự. Vì vậy, trừ khi bạn thực hiện một vài sự thay đổi lớn như là sử dụng một quy trình khác, làm việc với các ứng dụng phức tạp hơn, hay thay đổi môi trường phát triển, bạn có thể sẽ mắc phải số lượng sai sót tương tự trong chương trình kế tiếp như đã từng mắc trong chương trình trước đó.

Phần còn lại của bản tổng kết kế hoạch dự án được hoàn tất khá giống với cách trước đó. Chú ý rằng khi đã có các tỉ lệ Đến ngày, bạn không cần phải theo dõi Đơn vị và

Tốc độ phát triển chương trình trong bản ghi số công việc nữa. Tuy nhiên, vì bản ghi này rất thuận tiện cho việc tham khảo thông tin dự án, bạn nên tiếp tục theo dõi số công việc trong chương trình.

Sinh viên	Sinh viên X	Ngày	28/10/96
Chương trình		Chương trình #	10
Người hướng dẫn	Thầy Z	Ngôn ngữ	Ada

Tóm tắt	**Kế hoạch**	**Thực tế**	***Đến ngày***
Phút/LOC	6.76	6.12	6.50
LOC/Giờ	8.88	9.80	9.23
Sai sót/KLOC			
Hiệu suất			
A/FR			
Kích thước chương trình (LOC)			
Tổng mới và thay đổi	44	57	105
Kích thước tối đa	58		
Kích thước tối thiểu	30		

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	***Đến ngày***	***Đến ngày %***
Lên kế hoạch	13	18	33	4.8
Thiết kế	11	43	55	8.1
Cài đặt	130	162	308	45.2
Xem lại mã				
Biên dịch	44	21	70	10.2
Kiểm thử	82	73	165	24.2
Tổng kết	17	32	51	7.5
Tổng cộng	297	349	682	100.0
Kích thước tối đa	392			
Kích thước tối thiểu	203			

Sai sót mắc phải	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế		2	2	25.0	
Cài đặt		6	6	75.0	
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng		8	8	100.0	

Sai sót loại bỏ	***Kế hoạch***	***Thực tế***	***Đến ngày***	***Đến ngày %***	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch		6	6	75.0	
Kiểm thử		2	2	25.0	
Tổng cộng		8	8	100.0	

Bảng 3.2.6 Một ví dụ bản tổng kết kế hoạch đề án PSP

3.3 Tìm kiếm sai sót

3.3.1 Các bước trong tìm kiếm sai sót

Mặc dù không có cách nào để ngừng việc mắc phải sai sót, nhưng chúng ta có thể tìm và loại bỏ hầu hết các sai sót sớm trong quá trình phát triển. Sau khi đã tìm hiểu quy trình PSP, bạn sẽ nhận thấy rằng việc tìm kiếm và loại bỏ lỗi sớm sẽ tiết kiệm thời gian và sẽ cho sản phẩm tốt hơn. Chẳng hạn như nếu bạn tìm thấy và sửa sai sót thiết kế trước khi chuyển sang cài đặt thì bạn sẽ không phí thời gian thực thi bản thiết kế sai. Tương tự, khi bạn sửa lỗi cài đặt trước khi biên dịch và kiểm thử thì sẽ tiết kiệm được thời gian tìm kiếm và chỉnh sửa những lỗi này trong quá trình biên dịch và kiểm thử. Phần này chỉ cho bạn cách tìm ra sai sót sớm trong quy trình phát triển và cung cấp các dữ liệu mà bạn có thể dùng để đánh giá tính hiệu quả của những phương pháp loại bỏ sai sót này.

Có nhiều cách khác nhau để tìm sai sót trong một chương trình, nhưng xét về bản chất thì những phương pháp này đều gồm những bước sau:

1. Nhận diện các triệu chứng sai sót
2. Luận ra từ những triệu chứng này để định vị sai sót.
3. Hiểu chương trình có sai sót gì
4. Quyết định sửa sai sót như thế nào.
5. Chỉnh sửa sai sót.
6. Kiểm tra xác nhận lại việc chỉnh sửa đã giải quyết được vấn đề

3.3.2 Những cách để tìm và chỉnh sửa lỗi

Có nhiều công cụ và trợ giúp được tạo ra để hỗ trợ cho các kỹ sư trong những bước này. Công cụ đầu tiên mà các kỹ sư vẫn thường dùng là trình biên dịch. Trình biên dịch có thể xác định được hầu hết các lỗi cú pháp nhưng nó không hiểu được ý định của bạn là gì. Vì vậy, trình biên dịch thường đưa ra rất nhiều thông báo lỗi cho các lỗi có vẻ đơn giản. Tuy nhiên, nó chỉ có thể đưa ra các triệu chứng sai sót và bạn phải tự tìm ra vấn đề là gì và nằm ở đâu. Trình biên dịch sẽ không phát hiện ra mọi lỗi chính tả, dấu câu hay những sai sót cú pháp khác. Lý do là vì trình biên dịch có thể thường phát sinh mã từ những chương trình nguồn có sai sót. Trong khi hầu hết những sai sót bị bỏ qua này là thiết kế sai thì cũng có những lỗi về cú pháp đơn giản. Dường như là không thể khi một trình biên dịch có thể bỏ qua những sai sót cú pháp nhưng dữ liệu của tác giả Humphrey từ vài ngàn sai sót C++ đã cho thấy điều này có xảy ra với khoảng 9.4% của lỗi cú pháp. Chỉ vì chương trình

kiểm tra lỗi chính tả không bắt hết tất cả các lỗi chính tả nên trình biên dịch cũng không bắt được hết các sai sót cú pháp.

Cách thứ hai để tìm ra các sai sót là thông qua kiểm thử. Có rất nhiều loại kiểm thử và tất cả chúng đều đòi hỏi các tester phải cung cấp dữ liệu và điều kiện test (thỉnh thoảng gọi là các trường hợp test hay kịch bản test). Chất lượng của việc kiểm thử vì vậy bị ảnh hưởng bởi cấp độ mà các kịch bản test này có bao quát hết tất cả các chức năng quan trọng của chương trình. Tester sẽ chạy các trường hợp test này để xem chương trình có cho kết quả đúng hay không. Điều này bao hàm một trách nhiệm khác của tester: biết kết quả của các bộ test sẽ như thế nào nếu chương trình chạy đúng.

Mặc dù các test có thể được sử dụng để kiểm tra hầu hết bất kỳ chức năng của chương trình nào, nhưng nó cũng có những hạn chế. Đầu tiên, giống như trình biên dịch, kiểm thử chỉ đưa ra bước đầu tiên của quy trình sửa chữa sai sót. Nghĩa là, bạn chỉ chuyển từ triệu chứng sang vấn đề trước khi bạn bắt đầu công việc sửa chữa. Vấn đề thứ hai là chúng ta không thể bao quát hết tất cả các trường hợp test. Nếu kiểm thử tất cả các khả năng thì sẽ phải cần rất nhiều test. Ngay cả những chương trình đơn giản cũng liên quan đến nhiều sự kết hợp có thể của dữ liệu và điều kiện tính toán, test toàn diện thì rất tiêu tốn thời gian. Cuối cùng, với bất cứ chương trình ngoại trừ chương trình đơn giản nào, test toàn diện thì không thể về mặt thực tế.

Cách thứ ba để tìm kiếm sai sót chính thì quá phổ biến. Đó chính là gửi những chương trình có sai sót cho người dùng và đợi họ phát hiện ra và báo cáo lại sai sót. Đây là chiến lược tốn nhiều chi phí nhất. Ví dụ, trong một năm, IBM tiêu tốn khoảng 250 triệu $ chỉ cho việc chỉnh sửa và cài đặt lại các bản vá cho 13000 sai sót được báo cáo lại từ khách hàng, tính ra khoảng 20.000$ cho mỗi sai sót.

Cách cuối cùng và cũng là cách hiệu quả nhất là tìm kiếm và chỉnh sửa các sai sót bằng việc cá nhân xem xét lại chương trình nguồn. Điều này nghe có vẻ là cách khó nhất để làm sạch một chương trình có sai sót, nhưng thực ra đó là cách nhanh nhất và hiệu quả nhất. Những phần sau sẽ giải thích cụ thể lý do tại sao.

3.3.3 Xem xét lại code

Xem xét lại code chính là một cách để tìm sai sót nhanh chóng. Để thực hiện việc xem lại code, bạn sẽ nghiên cứu mã nguồn để tìm lỗi. Thời điểm tốt nhất để làm công việc này là sau khi viết mã nguồn xong và trước khi chuẩn bị biên dịch và kiểm thử. Vì hầu hết những sai sót phần mềm đều bắt nguồn từ sự sơ suất đơn giản nên chúng dễ dàng được

nhận ra ngay sau khi bạn vừa thiết kế hay viết mã xong. Tại thời điểm này, bạn vẫn còn nhớ những gì định làm và đó cũng là lúc bạn biết cách sửa bất cứ sai sót nào.

Có rất nhiều cách để xem xét code, nhưng cách phổ biến nhất là in chương trình mã nguồn và xem xét mỗi dòng. Tất nhiên bạn có thể xem lại code trên màn hình máy tính nhưng các kỹ sư nhận thấy rằng thuận tiện hơn ngay cả với những chương trình nhỏ khi chúng được in trên giấy. Việc in trên giấy cũng cho phép bạn chuyển giữa các đoạn code, ghi chú và đánh dấu các đoạn đã hoàn tất một cách nhanh chóng.

Mặc dù việc xem lại code tiêu tốn thời gian nhưng nó sẽ hiệu quả hơn nhiều so với việc kiểm thử. Số liệu lấy từ các kỹ sư và sinh viên cho thấy việc xem lại code hiệu quả hơn từ 3 -5 lần. Ví dụ, một kỹ sư có thể tìm ra từ 2 - 4 lỗi trong 1 giờ khi kiểm thử đơn vị nhưng sẽ tìm ra 6 – 10 sai sót trong mỗi giờ xem lại code.

Lý do tại sao xem lại code hiệu quả là do trong quá trình xem lại code, bạn sẽ nhận ra sai sót chứ không phải triệu chứng của sai sót. Khi xem lại từng dòng lệnh, bạn nghĩ về những gì mà chương trình sẽ làm. Vì vậy khi có gì đó trông có vẻ không ổn, bạn có thể thấy được vấn đề và nhanh chóng chỉnh sửa. Vì thời gian đòi hỏi để chuyển từ triệu chứng sang vấn đề là phần chính của chi phí tìm và sửa sai sót trong suốt quá trình biên dịch và kiểm thử nên việc xem lại có thể tiết kiệm rất nhiều thời gian.

Tuy nhiên việc xem lại code cũng có nhược điểm. Hai nhược điểm chính là tốn thời gian và khó để thực hiện đúng đắn. Tuy nhiên, việc xem lại cũng là một kỹ năng nên có thể học và cải tiến bằng cách luyện tập. Nhưng ngay cả khi có kinh nghiệm, bạn cũng sẽ chỉ tìm được trung bình từ 75% đến 80% sai sót trong chương trình. Nó cũng sẽ chiếm ít nhất 30 phút để xem xét kỹ lưỡng mỗi 100 LOC của mã nguồn. Khi thực hiện xem lại nhanh hơn nhiều tốc độ này, bạn sẽ luôn bỏ sót rất nhiều sai sót.

3.3.4 Tại sao cần phải tìm sai sót sớm?

Có rất nhiều lý do để xem lại chương trình trước khi biên dịch và kiểm thử chúng. Lý do quan trọng nhất là bạn không thể đưa ra một chương trình có sai sót và sau đó làm cho nó trở thành một sản phẩm có chất lượng. Một khi bạn đã viết một chương trình có khiếm khuyết thì nó sẽ luôn luôn là khiếm khuyết. Bạn có thể đã sửa tất cả những lỗi đã biết và có thể làm cho nó hoạt động theo những đặc tả mà bạn đã kiểm thử, nhưng khi đó nó sẽ là một chương trình khiếm khuyết với nhiều miếng vá.

Hãy xét một ví dụ, giả sử bạn định mua một xe hơi mới. Trước khi quyết định, bạn tham quan nhà máy lắp ráp của 2 hãng sản xuất. Tại một nhà máy, bạn thấy có rất nhiều xe

đẹp đi ra khỏi hàng xe và đi vào kiểm thử. Mặc dù chiếc xe nhìn thật tuyệt, việc kiểm thử đã phát hiện ra trung bình 10 sai sót mỗi xe. Tất cả các sai sót này đều được sửa chữa và xe được gửi đến cho những người buôn bán.

Tại nhà máy thứ 2 cũng thực hiện kiểm thử như nhà máy đầu tiên. Tuy nhiên, ở đây, cứ mỗi 10 xe thì việc kiểm thử chỉ tìm thấy 1 sai sót. Ngay cả nếu xe ở nhà máy này bán đắt hơn một chút thì bạn cũng sẽ vẫn thích chúng hơn, bất chấp các sự khác biệt nào khác. Bạn biết rằng việc kiểm thử sẽ không thể tìm thấy được tất cả các vấn đề, cứ giả sử rằng quy trình sản xuất tạo ra một vật vô dụng thì chiếc xe đó vẫn luôn là một vật vô dụng, bất chấp số lượng kiểm thử và thanh tra như thế nào đi nữa.

Chương trình cũng như vậy. Để sản xuất được phần mềm chất lượng, mỗi bước phát triển phần mềm phải có chất lượng cao. Những việc tuân thủ chất lượng một cách nghiêm ngặt như vậy có thể rất đắt đỏ, nhưng thực ra chúng sẽ tiết kiệm được thời gian.

3.3.5 Chi phí của việc tìm và sửa lỗi

Trong những dự án phần mềm điển hình, một sản phẩm được chia thành nhiều thành phần chương trình nhỏ hay các module. Mỗi kỹ sư sau đó sẽ phát triển một hay nhiều trong số những module này. Sau các giai đoạn thiết kế, thực thi và biên dịch module, người kỹ sư sẽ thực hiện việc kiểm thử đơn vị. Sau đó, các module được kết hợp lại thành các thành phần lớn hơn và được kiểm thử tích hợp. Các thành phần được kiểm thử ở các mức độ khác nhau trước khi được kết hợp thành sản phẩm để kiểm thử sản phẩm. Cuối cùng thì sản phẩm được lắp ráp thành hệ thống và được đưa vào để kiểm thử hệ thống. Mặc dù có nhiều sự khác nhau về kích thước, độ phức tạp của hệ thống, thì những qui trình tương tự cũng được dùng cho hầu hết tất cả các loại sản phẩm phần mềm.

Chi phí cho việc tìm kiếm và sửa lỗi tăng khoảng 10 lần theo mỗi bước trong qui trình phát triển. Trong giai đoạn xem lại code, bạn sẽ tìm và sửa sai sót với tốc độ trung bình là 1 đến 2 phút/sai sót. Trong kiểm thử đơn vị thì tốc độ trung bình sẽ là từ 10 đến 20 phút/sai sót hoặc hơn. Với chỉ một ít sai sót thôi thì chênh lệch cũng đã lên tới vài giờ.

Thời gian để tìm sai sót trong kiểm thử tích hợp, thành phần hay hệ thống cũng sẽ khác nhau cùng với quy mô và độ phức tạp của hệ thống. Tìm và sửa sai sót trong các hệ thống lớn và phức tạp hơn thì sẽ cần nhiều thời gian hơn. Ví dụ, trong kiểm thử tích hợp, mỗi sai sót có thể tốn một giờ hoặc hơn, và trong kiểm thử hệ thống mỗi sai sót có thể chiếm từ 10 đến 40 giờ hoặc hơn. Một khi sản phẩm đã được chuyển tới khách hàng, chi phí sẽ còn nhiều hơn rất nhiều, phụ thuộc vào loại sản phẩm và số lượng khách hàng.

Một lí do quan trọng khác cho việc phải tìm kiếm sai sót sớm là việc biên dịch, gỡ lỗi và kiểm thử không hẳn là hiệu quả tuyệt đối. Trình biên dịch là công cụ tìm lỗi nhanh nhất nhưng chúng cũng chỉ tìm được khoảng 90% lỗi cú pháp, và rất ít lỗi logic. Việc kiểm thử đơn vị có vẻ là chiến lược kiểm thử hiệu quả nhất nhưng nó cũng chỉ tìm ra khoảng một nửa những sai sót trong chương trình. Sau giai đoạn kiểm thử đơn vị, hiệu quả kiểm thử giảm dần.

Vì vậy, nếu muốn sản xuất một sản phẩm chất lượng cao thì hoặc là phải tạo ra một chương trình sạch lỗi ngay từ ban đầu hoặc là phải tốn nhiều thời gian cho việc kiểm thử.

3.3.6 Sử dụng xem xét lại để tìm sai sót

Tiêu chuẩn đầu vào	Kiểm tra các phần sau đã có đầy đủ: - Yêu cầu - Bản thiết kế chương trình - Mã nguồn của chương trình - Tiêu chuẩn cài đặt
1 Quy trình xem lại	Đầu tiên, tạo ra mã nguồn chương trình đã hoàn tất Trước khi biên dịch hay kiểm thử chương trình, in ra mã nguồn chương trình. Kế đến, xem lại code. Trong suốt quá trình xem lại, cẩn thận kiểm tra từng dòng code để tìm và chỉnh sửa càng nhiều sai sót mà bạn có thể tìm thấy càng tốt.
2 Sửa chữa sai sót	Sửa các sai sót được tìm thấy. Kiểm tra lại các chỉnh sửa để bảo đảm rằng chúng đã đúng. Ghi nhận lại các sai sót trong bản ghi ghi chép sai sót.
3 Xem lại về độ bao phủ	Kiểm tra thiết kế đã đáp ứng tất cả các chức năng mô tả trong yêu cầu hay chưa. Kiểm tra xem mã nguồn có thực thi tất cả các thiết kế.
4 Xem lại logic chương trình	Kiểm tra logic thiết kế đã đúng hay chưa. Kiểm tra chương trình đã thực thi chính xác logic thiết kế hay không.
5 Kiểm tra tên và kiểu	Kiểm tra tất cả các tên và kiểu đã được định nghĩa và sử dụng đúng hay không. Kiểm tra về việc định nghĩa đúng các kiểu integer, long integer và kiểu dữ liệu chấm động
6 Kiểm tra tất cả các biến	Bảo đảm rằng tất cả các biến đều được khởi tạo. Kiểm tra các vấn đề tràn (overflow, underflow, out of range…)
7 Kiểm tra cú pháp chương trình	Kiểm tra mã nguồn có theo đúng đặc tả của ngôn ngữ hay không.
Tiêu chuẩn đầu ra	Khi hoàn tất, bạn phải có được: - Một mã nguồn hoàn chỉnh và chính xác. - Bản ghi thời gian hoàn tất. - Bản ghi sai sót hoàn tất.

Bảng 3.3.1 Kịch bản xem lại code

Có lẽ bạn thấy khó tin rằng việc theo dõi và ghi nhận lại các sai sót sẽ cải thiện công việc của bạn nhưng nhiều sinh viên khi áp dụng phương pháp này đã giảm được số lượng sai sót mà họ tìm thấy trong khi biên dịch và test từ 5 đến 10 lần. Việc xem lại code hiệu quả đến nỗi sau khi bạn đã sử dụng chúng và thấy được hiệu quả, bạn sẽ xem việc xem xét lại là một phần hiển nhiên trong quy trình cá nhân của mình.

Bước đầu tiên trong việc xem lại là để hiểu được loại sai sót mà bạn phạm phải. Đây là lí do chính cho việc tại sao phải tập hợp lại các dữ liệu về sai sót. Loại sai sót trong chương trình tiếp theo cũng có thể giống những sai sót trong các chương trình trước. Điều này là sự thật nếu bạn vẫn tiếp tục phát triển phần mềm theo cách thức cũ. Mặt khác, khi bạn có kỹ năng và kinh nghiệm hoặc khi bạn thay đổi quy trình thì số lượng và loại sai sót mới thay đổi.

Mục đích của việc xem lại code là tìm được càng nhiều sai sót càng sớm càng tốt trong qui trình phát triển phần mềm. Bạn có thể cũng muốn tìm được sai sót trong khoảng thời gian càng ngắn càng tốt. Kịch bản để xem lại code được thể hiện trong bảng dưới đây. Điều quan trọng khi xem lại code là:

- Thực hiện xem lại trước khi biên dịch lần đầu.
- Thực hiện xem lại trên mã nguồn được in trên giấy.
- Ghi nhận lại mỗi sai sót tìm thấy được trong bản ghi ghi chép sai sót.
- Trong suốt quá trình xem lại, kiểm tra tất cả các loại sai sót đã gặp trước đây trong biên dịch và kiểm thử.

3.3.7 Lý do xem xét lại trước khi biên dịch

Có một vài lý do để xem lại code trước khi biên dịch chúng:

1. Dù là xem lại trước hay sau khi biên dịch thì cũng chiếm thời gian như nhau để thực hiện việc xem lại một cách kỹ lưỡng.
2. Xem lại trước sẽ tiết kiệm rất nhiều thời gian biên dịch. Trước khi thực hiện xem lại code, các kỹ sư thường phải bỏ ra từ 12% - 15% thời gian phát triển để biên dịch. Một khi họ xem lại code, thời gian biên dịch sẽ giảm xuống còn 3% hay ít hơn.
3. Một khi các kỹ sư đã biên dịch chương trình thì thường công việc xem lại của họ không được kỹ lưỡng.
4. Trước hoặc sau khi xem lại code thì biên dịch đều có hiệu quả ngang nhau.

5. Kinh nghiệm cho thấy khi chương trình có nhiều sai sót trong biên dịch thì sẽ có nhiều sai sót trong kiểm thử.

3.3.8 Các dạng xem lại khác

Trong các tổ chức phần mềm, cách làm phổ biến là các kỹ sư xem xét chương trình lẫn nhau. Việc này gọi là xem xét lại ngang hàng hay thanh tra. Thanh tra tốt có thể tìm được từ 50% - 70 % sai sót trong một chương trình. Việc này có thể tốn nhiều thời gian nhưng chúng tỏ ra vô cùng hiệu quả. Nguyên nhân là các kỹ sư thường khó nhận ra sai sót của chính bản thân mình. Họ tạo ra thiết kế và họ biết rằng thiết kế đó định làm gì. Nếu ý tưởng cơ sở của họ có thiếu sót hoặc họ tạo ra các thiết kế hay các giả định thực thi sai thì họ thường có vấn đề trong việc phát hiện ra chúng. Việc thanh tra có thể giúp vượt qua vấn đề này. Dữ liệu về thời gian để tìm ra sai sót bằng thanh tra được thể hiện trong bảng dưới.

Tham khảo	Thanh tra	Kiểm thử	Sử dụng
Ackerman	1	2-10	
O'Neil	0.26		
Ragland		20	
Russel	1	2-4	33
Shooman	0.6	3.05	
vanGenuchten	0.25	8	
Weller	0.7	6	

Bảng 3.3.2 Số giờ để tìm ra sai sót

Với các chương trình nhỏ thì việc thanh tra thường không xác đáng, nhưng với các dự án lớn hơn hay bất cứ chương trình công nghệ nào thì việc thanh tra nên luôn được thực hiện. Chiến lược tốt nhất là thực hiện xem lại code cá nhân trước khi biên dịch. Sau đó, trước khi thực hiện bất cứ kiểm thử nào, hãy tiến hành thanh tra. Tuy nhiên, trong tài liệu này, chúng ta sẽ không bàn đến vấn đề thanh tra.

3.4 Danh sách kiểm tra (checklist) xem lại code

3.4.1 Tại sao checklist lại có ích?

Một checklist bao gồm một chuỗi các bước thủ tục mà bạn muốn làm theo một cách chính xác. Khi có những việc quan trọng cần làm một cách chính xác như đã định rõ, người ta thường sử dụng checklist. Ví dụ, các phi công sử dụng chúng để kiểm tra chuẩn bị bay trước khi cất cánh. Mặc dù họ vừa mới kiểm tra cũng chiếc máy bay đó cách đó 1 giờ, họ vẫn làm lại một lần nữa. Một nghiên cứu về tai nạn của một hãng hàng không Mỹ cho thấy trước mỗi tai nạn, checklist chuẩn bị bay đã không được tuân theo một cách nghiêm ngặt.

Bởi vì tìm ra và chỉnh sửa được mọi lỗi trong chương trình là điều cần thiết, bạn phải theo một quy trình chính xác. Một checklist có thể bảo đảm việc đi theo quy trình đó. Trong phần này, chúng ta sẽ làm việc với một checklist rất đặc biệt: checklist giúp bạn tìm sai sót trong khi thực hiện xem lại chương trình bạn đã viết. Bạn sẽ biết cách để tạo ra một checklist xem lại mã được thiết kế riêng để tìm các sai sót chính xác đã gây rắc rối cho bạn.

Checklist cũng có thể là một nguồn các ý tưởng. Khi bạn đi theo một checklist cá nhân, bạn sẽ biết bạn xem lại code như thế nào. Nếu bạn sử dụng checklist một cách đúng đắn thì bạn có thể biết được bao nhiêu sai sót bạn sẽ tìm thấy với mỗi bước checklist. Bạn cũng có thể đo đươc tính hiệu quả của quy trình xem lại và cải tiến checklist. So sánh checklist của chính bạn với những kỹ sư khác cũng có thể cho bạn các cách phương pháp xem lại hữu ích khác.

Checklist gói gọn trong đó các kinh nghiệm cá nhân. Bằng cách sử dụng và cải tiến thường xuyên checklist cá nhân, bạn sẽ càng ngày càng tiến bộ trong việc tìm kiếm sai sót trong chương trình. Checklist cũng giúp bạn tìm lỗi với ít thời gian hơn.

3.4.2 Một checklist ví dụ

Checklist ở bảng sau là do tác giả Humphrey thiết kế để xem lại chương trình C++.

Tên chương trình và #:

Mục đích	Hướng dẫn bạn trong việc tiến hành việc xem lại code hiệu quả	#	#	#	Đến ngày	Đến ngày%
Tổng quát	Khi bạn hoàn thành mỗi bước xem lại, ghi chú số lượng sai sót của loại tìm thấy trong các ô bên phải. Nếu không có, thì đánh dấu vào đó. Hoàn tất một checklist cho một chương trình, lớp, đối tượng, hay phương pháp trước khi bắt đầu xem xét tiếp theo.					
Hoàn tất	Kiểm tra lại tất cả các chức năng trong thiết kế đã được cài đặt chưa.					
Includes	Kiểm tra xem các *include* có hoàn tất chưa					
Khởi tạo	Kiểm tra việc khởi tạo của các tham số và các biến. • Tại lúc bắt đầu chương trình. • Lúc bắt đầu của mỗi vòng lặp. • Tại đầu vào của mỗi hàm hay thủ tục.					
Các lời gọi	Kiểm tra các định dạn của các lời gọi hàm • Con trỏ • Tham số • Sử dụng toán tử '&'					
Tên	Kiểm tra việc sử dụng và chính tả của tên: • Nó có phù hợp không? • Nó có ở nằm phạm vi được khai báo không? • Các cấu trúc/lớp có sử dụng tham chiếu '.'?					
Chuỗi	Kiểm tra tất cả các chuỗi: • Có được nhận dạng bởi con trỏ. • Kết thúc bằng ký tự NULL					
Con trỏ	Kiểm tra tất cả các con trỏ: • Có được khởi tạo NULL • Chỉ hủy sau khi có cấp phát new • Luôn xóa sau khi sử dụng nếu cấp phát new					
Định dạng đầu ra	Kiểm tra định dạng đầu ra: • Sự phân dòng có hợp lý không? • Khoảng cách có đúng không?					
Cặp {}	Bảo đảm rằng {} được đặt đúng, khớp nhau.					
Toán tử logic	Kiểm tra việc sử dụng đúng các toán tử ==, =, ||, ... Kiểm tra mọi biểu thức logic có nằm trong ()					
Kiểm tra từng dòng	Kiểm tra từng dòng lệnh: • Đúng cú pháp. • Đúng dấu câu					
Các chuẩn	Bảo đảm rằng mã phù hợp với các chuẩn cài đặt					
Mở và đóng tập tin	Bảo đảm tất cả các tập tin: • Được khai báo đúng • Được mở • Đã đóng					
Tổng thể	Nhìn tổng thể chương trình để kiểm tra những vấn đề của hệ thống và những vấn đề không mong đợi					
Tổng cộng						

Bảng 3.4.1 Hướng dẫn và checklist xem lại code C++

3.4.3 Sử dụng checklist xem lại code

Để sử dụng checklist xem lại code, đọc mỗi mục theo thứ tự và làm theo những gì mô tả một cách chính xác. Khi hoàn tất một mục thì đánh dấu vào checklist. Cuối cùng,

xem lại toàn bộ checklist để đảm bảo là bạn đã kiểm tra hết mọi mục. Nếu bạn vẫn chưa kiểm tra hết thì quay lại và thực hiện các mục đã bỏ sót, đánh dấu lại và lại tiếp tục lướt qua danh sách để bảo đảm không bỏ sót gì khác. Khi sử dụng checklist, làm theo các bước sau:

1. Xem xét tỉ mỉ chương trình với mỗi mục trong checklist. Ví dụ, trong checklist ở bảng 3.4.1, đầu tiên ta sẽ xem toàn bộ chương trình để bảo đảm nó hoàn toàn thực thi theo đúng thiết kế. Trong suốt quá trình xem lại, nếu tìm thấy bất cứ sai sót nào khác, hãy sửa chúng. Tuy nhiên, dự định của bản vẫn là kiểm tra chương trình có theo đúng thiết kế. Kế đến, tiếp tục xem lại với mục kế tiếp của checklist…
2. Trong bất cứ quá trình kiểm tra nào, nếu tìm thấy sai sót thì ghi chú lại bằng một gạch trong chỗ trống đầu tiên chưa dùng # ở bên phải. Với lỗi thứ 2, gạch thêm một gạch cũng ở ô đó. Vì vậy, sau khi hoàn tất việc xem lại, bạn có thể biết được có bao nhiêu sai sót bạn đã tìm thấy với mỗi bước xem xét lại.
3. Sau khi hoàn tất mỗi mục kiểm tra, nếu không tìm thấy sai sót nào thì ta đặt một dấu '×' vào ô # đầu tiên chưa dùng.
4. Khi xem lại chương trình có một vài hàm, đối tượng, hay thủ tục, tốt nhất là xem xét chúng một cách riêng biệt. Nghĩa là, xem lại hàm đầu tiên hoàn tất và điền vào cột # đầu tiên, sau đó xem lại hàm thứ 2 và lại điền vào cột # thứ 2, cứ như vậy cho đến khi xem xét hết tất cả các thành phần của chương trình.
5. Cuối cùng, nên xem lại tổng thể toàn bộ chương trình để tìm ra những loại vấn đề mới, không mong đợi, các vấn đề về hệ thống hay người dùng.

Đi theo qui trình chung với 1 checklist xem lại code được mô tả trong kịch bản xem lại code cập nhật ở bảng 3.4.2 và kịch bản quy trình PSP cập nhật ở bảng 3.4.3 sau. Kịch bản này có một số thay đổi khi thêm vào checklist và đòi hỏi cần phải hoàn thành nó khi thực hiện xem lại. Biểu mẫu tổng kết kế hoạch dự án PSP và các chỉ dẫn vẫn không thay đổi.

Tiêu chuẩn đầu vào	Kiểm tra các phần sau đã có đầy đủ: - Yêu cầu - Bản thiết kế chương trình - Mã nguồn của chương trình - Tiêu chuẩn cài đặt ***- Một bản checklist xem lại code***
Tổng quát	***Sử dụng checklist xem lại code.*** ***Làm theo các chỉ dẫn của checklist trong quá trình xem lại.*** ***Khi kết thúc xem lại, điền giá trị vào cột Đến ngày và Đến ngày% và dòng Tổng cộng***
1 Quy trình xem lại	Đầu tiên, tạo ra mã nguồn chương trình đã hoàn tất Trước khi biên dịch hay kiểm thử chương trình, in ra mã nguồn chương trình Kế đến, xem lại code. Trong suốt quá trình xem lại, cẩn thận kiểm tra từng dòng code để tìm và chỉnh sửa càng nhiều sai sót mà bạn có thể tìm thấy càng tốt.
2 Sửa chữa sai sót	Sửa các sai sót được tìm thấy. Kiểm tra lại các chỉnh sửa để bảo đảm rằng chúng đã đúng. Ghi nhận lại các sai sót trong bản ghi ghi chép sai sót.
3 Xem lại về độ bao phủ	Kiểm tra thiết kế đã đáp ứng tất cả các chức năng mô tả trong yêu cầu hay chưa. Kiểm tra xem mã nguồn có thực thi tất cả các thiết kế.
4 Xem lại logic chương trình	Kiểm tra logic thiết kế đã đúng hay chưa. Kiểm tra chương trình đã thực thi chính xác logic thiết kế hay không.
5 Kiểm tra tên và kiểu	Kiểm tra tất cả các tên và kiểu đã được định nghĩa và sử dụng đúng hay không. Kiểm tra về việc định nghĩa đúng các kiểu integer, long integer và kiểu dữ liệu chấm động
6 Kiểm tra tất cả các biến	Bảo đảm rằng tất cả các biến đều được khởi tạo. Kiểm tra các vấn đề overflow, underflow, out of range…
7 Kiểm tra cú pháp chương trình	Kiểm tra mã nguồn có theo đúng đặc tả của ngôn ngữ hay không.
8 Đọc lướt	***Đọc lướt tổng thể chương trình để kiểm tra các vấn đề về hệ thống và các vấn đề không mong đợi***
Tiêu chuẩn đầu ra	Khi hoàn tất, bạn phải có được: - Một mã nguồn hoàn chỉnh và chính xác. - Bản ghi thời gian hoàn tất. - Bản ghi sai sót hoàn tất.

Bảng 3.4.2 Kịch bản xem lại code

3.4.4 Xây dựng một checklist cá nhân

Để xây dựng một checklist xem lại code, đầu tiên xem lại dữ liệu sai sót để biết được loại sai sót nào hay gây ra vấn đề nhất. Khởi đầu thì bạn sẽ có rất ít thông tin về dữ liệu sai sót nhưng bạn có nhiều hơn qua mỗi dự án mới. Để hiệu quả nhất, nhớ rằng

checklist phải được thiết kế cho chính bản thân bạn, cho ngôn ngữ mà bạn sử dụng, và cho loại sai sót mà bạn thường tìm thấy và bỏ sót. Checklist của ai đó khác có thể giúp bạn khi bắt đầu, nhưng nó sẽ không hiệu quả bằng checklist được thiết kế riêng cho các nhu cầu riêng của bạn.

Sau đây là một số mẹo giúp bạn tạo ra một checklist cá nhân có ích:

1. Lập một danh sách theo loại của số sai sót tìm thấy trong mỗi pha của qui trình phần mềm. Bảng dưới là ví dụ dữ liệu của sinh viên X. Ở phần dưới của bảng, cậu liệt kê các sai sót tìm thấy trong mỗi pha cho mỗi chương trình. Điều này giúp kiểm tra một cách dễ dàng tất cả các sai sót có được đếm.

Loại	**Mắc phải ở pha**			**Loại bỏ ở pha**			**Bỏ sót**
	Thiết kế	*Cài đặt*	*Khác*	*Xem lại*	*Biên dịch*	*Kiểm thử*	*trong Xem lại*
10							
20		8		4	4		4
30							
40	2	3		1	4		4
50		2			1	1	2
60							
70							
80	2	3			1	4	5
90							
100							
Tổng cộng	4	16		5	10	5	15
Chương trình							
10	2	6			6	2	8
11	1	5		3	2	1	3
12	1	5		2	2	2	4

Bảng 3.4.3 Bản phân tích dữ liệu sai sót của sinh viên X

Loại	**Mắc phải ở pha**			**Loại bỏ ở pha**			**Bỏ sót**
	Thiết kế	*Cài đặt*	*Khác*	*Xem lại*	*Biên dịch*	*Kiểm thử*	*trong xem lại*
80	2	3			1	4	5
20		8		4	4		4
40	2	3		1	4		4
50		2			1	1	2
60							
100							
30							
10							
70							
90							
Tổng cộng	4	16		5	10	5	15

Bảng 3.4.4 Dữ liệu sai sót được sắp xếp của sinh viên X

2. Sắp xếp các loại sai sót theo thứ tự giảm dần của số sai sót được tìm thấy trong pha biên dịch và kiểm thử. Ví dụ về danh sách này ở bảng trên đây.
3. Với các loại sai sót có số lượng sai sót nhiều nhất đó, kiểm tra bản ghi ghi chép sai sót để biết vấn đề gây ra hầu hết các rắc rối là gì? Từ bảng 3.4.4, ta nhận thấy các lỗi này là lỗi chức năng loại 80, lỗi cú pháp loại 20 và lỗi chỉ định loại 40.
4. Với những sai sót bắt nguồn từ những vấn đề quan trọng nhất này, hãy định rõ các bước cần thực hiện trong xem lại code để tìm ra chúng. Giả sử rằng, với những sai sót cú pháp loại 20, sinh viên X nhận thấy đa số vấn đề của cậu là thiếu hoặc để nhầm chỗ dấu ";". Khi đó cậu quyết định thêm 1 mục kiểm tra để nhắc nhở phải xem xét mỗi dòng lệnh mã nguồn để kiểm tra các dấu ";".
5. Thêm vào các mục kiểm tra trong checklist để đảm bảo là bạn sẽ đi qua những bước này. Ví dụ, ở đây, sinh viên X sẽ thêm vào 1 mục ";" nhằm nói rằng: xem lại mỗi dòng chương trình nguồn đẻ kiểm tra việc sử dụng dấu ";" đã đúng chưa.
6. Sau khi sử dụng checklist mới, kiểm tra dữ liệu sai sót một lần nữa theo cách tương tự.
7. Nếu checklist hiệu quả trong việc tìm ra những sai sót loại này, hãy thêm vào một loại khác và lại sử dụng nó như vậy.
8. Nếu checklist không hiệu quả trong việc tìm kiếm sai sót, cố gắng thay đổi chúng để nhận diện tốt hơn các sai sót này và thử dùng nó 1 lần nữa xem sao. Trong trường hợp này, nếu sinh viên X nhận thấy cậu ta thường xuyên gõ ":" thay vì ";", anh ta có thể thêm một lời nhắc nhở để kiểm tra mỗi dấu ";" để chắc chắn là nó không bị gõ sai thành ":". Checklist đã được cập nhật của cậu ở bảng 3.4.5.
9. Trong quá trình phát triển hay cập nhật lại checklist, nhóm các mục tương tự nhau lại và không nhân đôi việc thực hiện chúng. Nếu một mục kiểm tra nào đó không làm tốt, hãy thay thế nó thay vì thêm những mục kiểm tra cho cùng một thứ.
10. Sau khi phát triển mỗi chương trình mới, kiểm tra ngắn gọn dữ liệu sai sót và checklist theo cách như thế này để xác định những thay đổi và những bổ sung có lợi cho chương trình.
11. Đó cũng là một ý kiến hay nếu chúng ta cân nhắc xem những bước nào sẽ ngăn chặn những sai sót này trong tương lai. Ví dụ như chúng ta cập nhật tiêu chuẩn cài đặt hay là thêm một bước vào qui trình thiết kế.

Tên chương trình và #:

Mục đích	Hướng dẫn bạn trong việc tiến hành việc xem lại code hiệu quả	#	#	#	#	Đến ngày	Đến ngày%
Tổng quát	Khi bạn hoàn thành mỗi bước xem lại, ghi chú số lượng sai sót của loại đó tìm thấy trong các ô bên phải. Nếu không có, thì đánh dấu vào đó. Hoàn tất một checklist cho một chương trình, lớp, đối tượng, hay phương pháp trước khi bắt đầu xem xét tiếp theo.						
Hoàn tất	Kiểm tra lại tất cả các chức năng trong thiết kế đã được cài đặt chưa.	X					
Includes	Kiểm tra xem *with* có hoàn tất chưa	X					
Khởi tạo	Kiểm tra việc khởi tạo của các tham số và các biến. • Tại lúc bắt đầu chương trình. • Lúc bắt đầu của mỗi vòng lặp. • Tại đầu vào của mỗi hàm hay thủ tục.	X					
Các lời gọi	Kiểm tra các định dạng của các lời gọi hàm • Dấu câu • Tham số	X					
Tên	Kiểm tra việc sử dụng và chính tả của tên: • Nó có phù hợp không? • Nó có ở nằm phạm vi được khai báo không? • Tất cả các cấu trúc/gói có sử dụng tham chiếu '.'?	1				2	40
Chuỗi	Kiểm tra tất cả các chuỗi: • Có được nhận dạng bởi con trỏ. • Kết thúc bằng ký tự NULL	X					
Con trỏ	Kiểm tra tất cả các con trỏ: • Có được khởi tạo NULL • Chỉ hủy sau khi có cấp phát new. • Luôn luôn xoá sau khi sử dụng nếu cấp phát new.	X					
Định dạng đầu ra	Kiểm tra định dạng đầu ra: • Sự phân dòng có hợp lý không? • Khoảng cách có đúng không?	X					
Cặp {}	Bảo đảm rằng {} được đặt đúng, khớp nhau.	X					
Toán tử logic	Kiểm tra việc sử dụng đúng các toán tử logic Kiểm tra mọi biểu thức logic có nằm trong ()	X					
Kiểm tra từng dòng	Kiểm tra từng dòng lệnh: • Đúng cú pháp. • ***";" có được sử dụng đúng*** • ***kiểm tra ";" không bị gõ nhầm thành ":"*** • Đúng dấu câu	1				3	60
Các chuẩn	Bảo đảm rằng mã phù hợp với các chuẩn cài đặt	X					
Mở và đóng tập tin	Bảo đảm tất cả các tập tin: • Được khai báo đúng • Được mở • Đã đóng	X					
Tổng thể	Nhìn tổng thể chương trình để kiểm tra những vấn đề của hệ thống và những vấn đề không mong đợi	X					
Tổng cộng		2					

Bảng 3.4.5 Checklist đã cập nhật của sinh viên X

Trong bảng 3.4.3 và 3.4.4, sinh viên X liệt kê tất cả các sai sót cậu mắc phải và loại bỏ được từ khi cậu bắt đầu thu thập dữ liệu sai sót. Dữ liệu này chỉ bao gồm 20 sai sót,

nhưng đây là tất cả dữ liệu mà cậu ta có. Ở bảng 3.4.3, đầu tiên cậu liệt kê tổng số các chương trình trong bản tổng kết kế hoạch dự án và xem qua các bản ghi sai sót để lấy thông tin về loại sai sót. Cậu thu được bảng 3.4.4 từ việc sắp xếp bảng 3.4.3 theo thứ tự số lượng sai sót giảm dần của cột bên phải nhất. Các mục trên cùng vì vậy sẽ liệt kê loại sai sót mà cậu hay bỏ sót nhất trong xem lại code. Liệt kê như vậy được gọi là sắp xếp Pareto, nó sẽ liệt kê các mục theo một thứ tự ưu tiên của dữ liệu. Chú ý rằng vì sinh viên X không thực hiện xem lại code cho chương trình 10 nên cậu tính tất cả các sai sót tìm thấy trong biên dịch và kiểm thử như là sai sót bị bỏ sót trong xem lại code.

3.4.5 Cải tiến checklist

Hãy tập thói quen xem lại thường xuyên các dữ liệu sai sót và kiểm tra lại checklist. Nếu các bước của checklist hiệu quả thì hãy giữ lại chúng. Nhưng khi có một số bước không hiệu quả, hãy nghĩ cách để cho chúng hiệu quả hơn và cập nhật lại checklist. Checklist vì vậy trở thành một tập gói gọn các kinh nghiệm cá nhân.

Đoạn sau đưa ra những đề nghị để cải tiến checklist của bạn:

1. Sau khi hoàn tất một chương trình, điền giá trị vào cột Đến ngày của checklist bằng cách cộng giá trị Đến ngày từ checklist đã hoàn tất gần đây nhất với số sai sót tìm thấy trong mỗi của việc xem lại này. (Xem bảng 3.4.5)
2. Hoàn tất việc điền các giá trị cho cột Đến ngày% bằng cách chia giá trị Đến ngày của từng dòng cho tổng số sai sót Đến ngày ở dòng cuối của checklist.
3. Trong pha tổng kết cho mỗi chương trình, so sánh checklist với bản ghi sai sót để tìm xem checklist cần được cải thiện như thế nào và ở đâu để tìm kiếm sai sót tốt hơn. Ngoài ra hãy xem xét việc bỏ đi các bước xem lại không tìm ra hay bỏ sót bất cứ sai sót nào trong khoảng từ 5 – 10 chương trình gần đây nhất.
4. Bạn nên tập hợp dữ liệu về khoảng hơn 20 sai sót trước khi cập nhật checklist. Khi bạn gặp phải cùng một sai sót nhiều lần trong quá trình biên dịch hay kiểm thử thì nên nghĩ đến việc cập nhật checklist để chỉ ra vấn đề đặc biệt này.
5. Hãy lượt bớt checklist theo định kỳ. Checklist theo thời gian sẽ lớn dần lên mà thế mạnh của checklist lại là tập trung sự chú ý. Khi nó phát triển quá lớn, bạn sẽ mất tập trung. Vì vậy, rất quan trọng để xem xét dữ liệu sai sót định kỳ và loại bỏ các mục không tìm ra vấn đề nữa.

Hãy nhớ rằng, việc cải thiện sẽ đến một cách chậm chạp. Ban đầu, khả năng tìm ra sai sót của bạn sẽ cải tiến với từng giai đoạn xem lại. Sau đó, việc cải tiến sẽ trở nên khó

khăn hơn. Hãy tiếp tục thu thập và phân tích dữ liệu sai sót và nghĩ về việc bạn sẽ làm gì để ngăn chặn hay tìm ra các sai sót bị bỏ sót một cách tốt hơn. Khi bạn còn làm điều này, bạn sẽ tiếp tục tiến bộ trong việc xem lại, bạn cũng sẽ tiếp tục cải tiến được chất lượng của chương trình mà bạn tạo ra.

3.4.6 Các chuẩn cài đặt

Một lý do tại sao checklist quan trọng là vì nó cung cấp một tiêu chuẩn để xem lại chương trình. Mặc dù những chuẩn xem lại code chủ yếu là những đặc tả cú pháp ngôn ngữ lập trình, chúng không định rõ các định dạng hay cách cài đặt. Vì những lý do như vậy mà bạn cần một chuẩn cài đặt.

Một **chuẩn** là một cơ sở đã được chấp nhận một cách chính thức. Một **chuẩn cài đặt** vì vậy định nghĩa một tập các cách thức cài đặt đã được chấp nhận, đóng vai trò như một mô hình mẫu cho công việc của bạn. Chuẩn này nên được dùng như là một hướng dẫn cho bạn viết mã nguồn. Những chuẩn như vậy thông thường chỉ rõ định dạng của mã nguồn, những câu gì để chia cách các dòng văn bản, các câu dự định như thế nào. Việc viết các lời bình thường được được định nghĩa, bao gồm cả việc khi nào thì cần các lời bình có tính cách giải thích. Thường thường, tên kỹ sư, ngày làm việc, tên chương trình, tên dự án và phiên bản cũng được đưa vào trong lời bình header nằm ở đầu chương trình. Bảng 3.4.6 là một ví dụ về chuẩn cài đặt của C++.

Mục đích	Hướng dẫn cách phát triển các chương trình viết bằng C++
Header chương trình	Tất cả các chương trình đều bắt đầu với một header mô tả

Định dạng Header	`/*************************************************/`
	`/* Chương trình   : Số chương trình            */`
	`/* Tên            : Tên của bạn                */`
	`/* Ngày          : Ngày bắt đầu phát triển    */`
	`/* Mô tả         : Mô tả ngắn gọn về chức năng */`
	`/*                chương trình                 */`
	`/*************************************************/`

Danh sách nội dung	Cung cấp một bản tóm tắt danh sách các nội dung

Ví dụ nội dung	`/*************************************************/`
	`/* Danh sách nội dung                          */`
	`/*    Sử dụng lại các chỉ dẫn                  */`
	`/*    Includes                                 */`
	`/*    Khai báo lớp                             */`
	`/*          Cdata                              */`
	`/*          ASet                               */`
	`/*    Mã nguồn ở C:\classes\CData.cpp          */`
	`/*          Cdata                              */`
	`/*          CData()                            */`
	`/*          Empty()                            */`
	`/*************************************************/`

Sử dụng lại các chỉ dẫn	Mô tả chương trình được sử dụng như thế nào. Cung cấp định dạng khai báo, các giá trị, kiểu tham số và giới hạn của các tham số.

	Cung cấp các cảnh báo về các giá trị không hợp lệ, điều kiện tràn, hay những điều kiện khác có khả năng dẫn đến những kết quả sai.

Ví dụ	`/*********************************************/`
	`/* Sử dụng lại các chỉ dẫn                    */`
	`/*   int PrintLine(char *line_of_character)  */`
	`/*   Mục đích: in chuỗi                      */`
	`/*      'line_of_character' nằm trên một dòng in*/`
	`/*   Giới hạn : Chiều dài tối đa là LINE_LENGTH */`
	`/*   Trả về : 0 nếu máy in không sẵn sàng in    */`
	`/*   Ngược lại : 1                           */`
	`/*********************************************/`

Từ định danh (identifier)	Sử dụng các tên có tính chất mô tả cho tất cả các tên biến, hàm, hằng, và những định danh khác. Tránh viết tắt hay các tên biến chỉ có 1 ký tự
Ví dụ từ định danh	`int number_of_student;        /* Thế này là TỐT */` `float x4,j, ftave;          /* Thế này là XẤU */`
Lời bình	Sưu liệu đầy đủ mã nguồn để người đọc có thể hiểu được hoạt động của nó. Lời bình nên giải thích cả mục đích và các hành vi của mã nguồn. Ghi chú lại các khai báo biến để chỉ ra mục đích của chúng.
Lời bình tốt	`If(record_count >limit) /* tất cả các record đều*/` `/* được xử lý ?                          */`
Lời bình xấu	`If(record_count >limit) /* kiểm tra nếu       */` `/* record_count lớn hơn limit             */`
Các phần chính	Những phần chính của chương trình nên được đi kèm với một đoạn lời bình trước đó để mô tả cách xử lý sẽ được thực hiện

Ví dụ	`/*********************************************/`
	`/* Phần chương trình này sẽ kiểm tra nội dung  */`
	`/* của mảng "grades" và sẽ tính điểm trung bình */`
	`/* cho lớp                                 */`
	`/*********************************************/`

Khoảng trắng	Viết chương trình với các khoảng trắng thích hợp để dễ đọc. Cách mỗi thành phần của chương trình với ít nhất một khoảng trắng.
Thụt lề	Thụt vào với mỗi cấp độ dấu ngoặc so với những dấu ngoặc trước. Mở và đóng ngoặc nên được gióng theo hàng thẳng với nhau.

Ví dụ	`While (miss_distance > threshold)`
	`{`
	`  success_code = move_robot(target_location);`
	`  if (success_code == MOVE_FAILED)`
	`  {`
	`  printf("The robot move has failed");`
	`  }`
	`}`

Viết hoa	Tất cả các định nghĩa đều được viết hoa. Tất cả các định danh, và những từ được dành riêng khác đều viết thường. Các thông báo được xuất ra cho người sử dụng có thể viết thường và hoa để diễn đạt một cách rõ ràng.
Ví dụ	`#define DEFAULT_NUMBER_OF_STUDENT  15` `int class_size = DEFAULT_NUMBER_OF_STUDENT;`

Bảng 3.4.6 Chuẩn cài đặt trong C++

Chuẩn cài đặt cũng giúp ích trong việc ngăn chặn lỗi. Chẳng hạn, bạn có thể liệt kê những thói quen nào đó để tránh phạm lỗi, như sử dụng câu lệnh go–to, có nhiều đầu ra từ các thủ tục, hay sử dụng đệ quy, hoặc là luôn khởi tạo các biến ngay từ đầu vào của vòng lặp hay khi vừa khai báo chúng…Việc đặt tên một cách không đầy đủ cũng là một nguyên nhân lỗi chủ yếu. Chỉ dùng những tên có quan hệ một cách rõ ràng với chức năng của biến, và các tên phải khác nhau đủ để chúng không dễ dàng bị lẫn lộn.

Một chuẩn cài đặt sẽ giúp bạn tạo ra một chương trình dễ đọc và dễ hiểu hơn. Mã nguồn dễ đọc cũng sẽ có ích trong kiểm thử và gỡ rối chương trình và có ích cho ai muốn sử dụng hay chỉnh sửa lại chương trình của bạn.

3.5 Dự đoán sai sót

3.5.1 Sử dụng dữ liệu sai sót

Cho đến lúc này, bạn đã tập hợp được các dữ liệu sai sót để giúp hiểu được những sai sót mà bạn mắc phải. Sau đó bạn sử dụng những dữ liệu này để thiết kế một checklist cá nhân để thực hiện xem lại code. Ở phần này bạn sẽ sử dụng các dữ liệu này để ước lượng số sai sót bạn có thể mắc phải trong một chương trình mới. Bằng cách sử dụng dữ liệu lịch sử, bạn sẽ thấy được cách để dự đoán một cách hợp lý số sai sót mà bạn mắc phải và loại bỏ trong mỗi pha của một dự án lập trình.

Việc ước lượng chính xác các mức độ sai sót rất quan trọng. Có thể bạn luôn phải thực hiện một kiểm thử khác hay xem lại code một lần nữa. Nhưng cách duy nhất để quyết định thực hiện kiểm thử và xem lại như vậy là phải phân tích dữ liệu sai sót. Bằng cách so sánh dữ liệu của dự án hiện hành với kinh nghiệm lịch sử bản thân, bạn có thể xác định gần như chính xác chất lượng của chương trình đang được phát triển. Sau đó bạn có thể quyết định việc thêm các bước loại bỏ các sai sót có cần thiết hay không.

3.5.2 Mật độ sai sót

Phép đo sai sót cơ bản được sử dụng trong PSP là số các sai sót trên một ngàn dòng lệnh (KLOC) hay còn gọi là mật độ sai sót (Dd – Defect density) và đơn vị của nó là số sai sót/KLOC. Để tính tổng số sai sót/KLOC trong một chương trình:

1. Tính tổng số sai sót (D - defects) tìm được trong tất cả các pha của qui trình.
2. Đếm số LOC Mới và Thay đổi (N) trong chương trình.
3. Tính số sai sót trên mỗi KLOC: Dd = 1000*D/N.

Ví dụ, một chương trình có 96 LOC và có tổng cộng 14 sai sót thì mật độ sai sót sẽ là 1000*14/96 = 145.83 sai sót/KLOC.

3.5.3 Dự đoán mật độ sai sót

Khi phát triển một chương trình mới, bạn sẽ gặp vấn đề trong việc ước lượng có bao nhiêu sai sót sẽ mắc phải vì con số này khác nhau trong mỗi chương trình. Có một số lý do giải thích điều này.

Đầu tiên, cùng với kinh nghiệm kỹ năng của bạn sẽ cải tiến dần. Khi bắt đầu một chương trình, bạn đối đầu với nhiều vấn đề mà trước đây bạn chưa hề gặp phải. Bạn có thể sẽ không chắc chắn một số hàm hay thủ tục hoạt động như thế nào, bạn có thể bị lúng túng bởi cấu trúc của ngôn ngữ, hoặc là gặp một trình biên dịch mới hay các vấn đề về môi trường. Các vấn đề này sẽ gây ra sự dao động trong thời gian phát triển và tỉ lệ mắc phải sai sót của bạn. Cùng với kinh nghiệm, dần dần bạn sẽ vượt qua các vấn đề này và ít phạm lỗi hơn. Điều này sẽ vừa giảm thiểu tổng số sai sót vừa giảm mức độ thay đổi trong các con số này. Một số sự giảm thiểu ban đầu trong mức độ sai sót vì vậy là kết quả của nhiều kinh nghiệm hơn và nắm được ngôn ngữ. Tuy nhiên để tiến xa hơn việc cải tiến ban đầu này, bạn cần thu thập và phân tích dữ liệu sai sót để tiếp tục cải tiến.

Lý do thứ hai vì sao tỉ lệ sai sót biến động là vì quy trình của bạn không ổn định. Khi bạn học cách viết chương trình, bạn cũng học luôn cả các phương pháp và các thủ tục mới. Việc tiến hành công việc của bạn cũng sẽ trở nên tiến triển hơn, gây ra dao động trong thời gian thực thi các nhiệm vụ lập trình khác nhau và trong số lượng sai sót mà bạn mắc phải.

Cuối cùng, sai sót bản thân chúng cũng là một nguồn biến đổi. Bạn càng mắc phải sai sót thì bạn càng phải cần thêm thời gian để sửa chữa chúng. Thời gian để sửa chữa chúng càng nhiều thì càng làm tăng nguy cơ mắc thêm nhiều sai sót hơn. Vì thời gian sửa lỗi thay đổi trong một phạm vi rộng nên một quy trình mắc phải nhiều sai sót cũng vì vậy mà không thể đoán trước được.

Tuy nhiên, khi quy trình của bạn cải tiến, nó sẽ trở nên ổn định hơn, giúp cho việc dự đoán các sai sót của bạn trở nên chính xác hơn. Nếu bạn bỏ ra một lượng thời gian vừa đủ để xem lại code, quy trình của bạn sẽ nhanh chóng ổn định hơn, khi đó nó sẽ dễ dự đoán hơn. Lúc đó, bằng cách theo dõi số lượng sai sót/KLOC mà bạn mắc phải và loại bỏ được trong các chương trình gần nhất, bạn có thể ước lượng khá chính xác số lượng sai sót mà bạn sẽ mắc phải và loại bỏ được trong tương lai.

3.5.4 Ước lượng sai sót

Khi lên kế hoạch một chương trình mới, đầu tiên ước lượng số LOC Mới và Thay đổi mà chương trình có thể có. Kế tiếp, tính số sai sót/KLOC trung bình cho các chương trình đã phát triển trước đó. Với những con số này, giờ bạn có thể tính được số sai sót/KLOC mong đợi trong chương trình mới là:

$$Dd_{\text{kế hoạch}} = 1000\ (D_1 + \ldots + D_i) / (N_1 + \ldots + N_i)$$

Chương trình số	Số sai sót (D)	Số LOC(N)
1	6	37
2	11	62
3	7	49
4	9	53
5	5	28
Đến ngày tổng cộng	38	229

Bảng 3.5.1 Một ví dụ về dữ liệu sai sót

Ví dụ, giả sử bạn có dữ liệu cho 5 chương trình như ở bảng trên, khi đó giá trị $Dd_{\text{kế hoạch}}$ được tính như sau:

$$Dd_{\text{kế hoạch}} = 1000*(6 + 11 + 7 + 9 + 5)/(37 + 62 + 49 + 53 + 28)$$
$$= 1000*38/229 = 165.94 \text{ sai sót/KLOC}$$

Giả sử chương trình mới cũng có cùng mật độ sai sót như thế, vậy ta tính số sai sót được dự đoán là:

$$D_{\text{kế hoạch}} = N_{\text{kế hoạch}} * Dd_{\text{kế hoạch}} / 1000$$

Bây giờ, cũng với ví dụ như trên, và giả sử thêm LOC ước lượng cho chương trình mới là 56, như vậy số sai sót dự đoán là:

$$D_{\text{kế hoạch}} = 56 * 165.94 / 1000 = 9.29 \text{ sai sót}$$

Sử dụng các dữ liệu này, bạn vì vậy cũng đoán được sẽ có 9 sai sót cho một dự án lập trình dự định có 56 LOC.

Kích thước Đến ngày và dữ liệu sai sót trong bản tổng kết kế hoạch dự án được thiết kế để giúp cho các công việc tính toán này

$$D_{\text{kế hoạch}} = N_{\text{kế hoạch}} * D_{\text{Đến ngày}} / N_{\text{Đến ngày}}$$

Sử dụng dữ liệu ở bảng 2.14.1, biểu thức này cho kết quả:

$$D_{\text{kế hoạch}} = 56 * 38 / 229 = 9.29 \text{ sai sót (cùng kết quả như đã tính trước đây)}$$

Với con số tổng sai sót dự đoán cho chương trình mới, bạn có thể tính số sai sót dự đoán sẽ mắc phải và loại bỏ được trong mỗi pha bằng:

$$D_{\text{kế hoạch/pha}} = D_{\text{kế hoạch}} * \text{Đến ngày\%} / 100$$

Đây là lý do tại sao chúng ta cần tính các giá trị Đến ngày và Đến ngày %. Chúng cung cấp dữ liệu lịch sử cần thiết cho việc ước lượng sai sót.

3.5.5 Kịch bản quy trình và bản tổng kết kế hoạch dự án cập nhật

Mục đích	Hướng dẫn bạn trong việc phát triển những chương trình nhỏ
Tiêu chuẩn đầu vào	- Mô tả vấn đề - Bản tổng kết kế hoạch dự án PSP ***- Bản checklist xem lại code*** - Dữ liệu về thời gian và kích thước thật sự của những chương trình trước - Bản ghi thời gian ***- Bản ghi sai sót***
1. Lên kế hoạch	- Ghi nhận những mô tả về chức năng của chương trình - Ước tính tổng số, tối đa, tối thiểu dòng lệnh cần thiết. - Xác định Phút/LOC - Xác định giá trị lớn nhất, nhỏ nhất và tổng cộng thời gian phát triển ***- Ước lượng số sai sót sẽ mắc phải và loại bỏ theo pha.*** - Ghi nhận những dữ liệu kế hoạch trong bản tổng kết kế hoạch dự án. - Ghi lại thời gian lên kế hoạch trong bản ghi thời gian.
2. Thiết kế	- Thiết kế chương trình - Ghi nhận lại thiết kế theo một định dạng chuẩn. - Ghi nhận lại thời gian thiết kế trong bản ghi thời gian
3. Cài đặt	- Thực thi thiết kế - Sử dụng dạng chuẩn để viết code. - Ghi nhận lại thời gian viết code trong bản ghi thời gian.
4. Xem lại code	***- Xem lại mã nguồn một cách đầy đủ.*** ***- Đi theo kịch bản xem lại code.*** ***- Chỉnh sửa và ghi nhận lại mỗi sai sót tìm thấy.*** ***- Ghi nhận thời gian xem lại trong bản ghi thời gian.***
5. Biên dịch	- Biên dịch chương trình - Sửa ***và ghi nhận*** tất cả các lỗi tìm thấy. - Ghi nhận lại thời gian biên dịch trong bản ghi thời gian.
6. Kiểm thử	- Kiểm thử chương trình. - Sửa ***và ghi nhận*** tất cả các lỗi tìm thấy. - Ghi nhận lại thời gian kiểm thử trong bản ghi thời gian.
7. Tổng kết	- Hoàn tất bản tổng kết kế hoạch dự án với thời gian, kích thước thực tế ***và dữ liệu sai sót*** ***- Xem lại dữ liệu sai sót và cập nhật lại checklist xem lại code*** - Ghi nhận thời gian tổng kết trong bản ghi thời gian.
Tiêu chuẩn đầu ra	- Một chương trình đã được kiểm thử kỹ càng. - Một thiết kế đã được sưu liệu một cách chính xác. ***- Một checklist xem lại code hoàn chỉnh*** - Danh sách các chương trình hoàn tất. - Bản tổng kết kế hoạch dự án đã hoàn tất. - Bản ghi thời gian và bản ghi sai sót đã hoàn tất.

Bảng 3.5.2 Kịch bản quy trình PSP

Sinh viên		Ngày	
Chương trình		Chương trình #	
Người hướng dẫn		Ngôn ngữ	

Tóm tắt	**Kế hoạch**	**Thực tế**	**Đến ngày**
Phút/LOC			
LOC/Giờ			
Sai sót/KLOC			
Hiệu suất			
A/FR			
Kích thước chương trình (LOC)			
Tổng mới và thay đổi			
Kích thước tối đa			
Kích thước tối thiểu			

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	**Đến ngày**	**Đến ngày %**
Lên kế hoạch				
Thiết kế				
Cài đặt				
Xem lại mã				
Biên dịch				
Kiểm thử				
Tổng kết				
Tổng cộng				
Kích thước tối đa				
Kích thước tối thiểu				

Sai sót mắc phải	***Kế hoạch***	**Thực tế**	**Đến ngày**	**Đến ngày %**	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Sai sót loại bỏ	***Kế hoạch***	**Thực tế**	**Đến ngày**	**Đến ngày %**	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã					
Biên dịch					
Kiểm thử					
Tổng cộng					

Bảng 3.5.3 Bản tổng kết kế hoạch dự án PSP

Mục đích	Mẫu này ghi nhận các thông tin ước lượng và thực tế của đề án
Đầu trang	Nhập các thông tin: - Tên và ngày hiện tại - Tên và mã số chương trình - Tên người hướng dẫn - Ngôn ngữ sử dụng để lập trình
Tóm tắt	
Phút/LOC	Trước khi phát triển: - Nhập giá trị Phút/LOC dự kiến cho đề án này. Sử dụng tốc độ Đến ngày từ chương trình gần nhất trong bản ghi công việc hay bản tổng kết kế hoạch dự án. Sau khi phát triển: - Chia tổng thời gian phát triển cho độ lớn chương trình thực tế để có chỉ số Phút/LOC thực tế ***và Đến ngày*** - Ví dụ, nếu dự án phát triển mất 196 phút và gồm 29 LOC, chỉ số Phút/LOC sẽ là 196/29=6.76
LOC/Giờ	Trước khi phát triển: - Tính LOC/Giờ dự kiến bằng cách lấy 60 chia cho Phút/LOC dự kiến Sau khi phát triển: - Để tính LOC/Giờ thực tế và Đến ngày, lấy 60 chia cho Phút/LOC thực tế Đến ngày Ví dụ: Với chỉ số Phút/LOC thực tế là 6.76, chỉ số LOC/Giờ thực tế là 60/6.76=8.88
Sai sót/KLOC	***Trước khi phát triển:*** ***- Tìm số sai sót/KLOC trong các chương trình gần đây nhất.*** ***- Sử dụng giá trị này như là số sai sót/KLOC kế hoạch cho dự án này.*** ***Sau khi phát triển:*** ***- Tính số sai sót/KLOC thực tế và Đến ngày cho chương trình này.*** ***- Với giá trị thực tế: Tổng số sai sót thực tế *1000 / Tổng LOC Mới và Thay đổi thực tế*** ***- Tính toán tương tự cho giá trị Đến ngày*** ***- Ví dụ: với 17 sai sót Đến ngày và 153 LOC Mới và Thay đổi thì chỉ số sai sót/KLOC Đến ngày là = 1000*17/153 = 111.11***
Độ lớn chương trình (LOC)	Trước khi phát triển: - Nhập giá trị Tổng cộng, Tối đa và tối thiểu của LOC Mới & Thay đổi Sau khi phát triển: - Đếm và nhập giá trị LOC Mới & Thay đổi thực tế. - Với Đến ngày, cộng thêm LOC Mới và Thay đổi thực sự với LOC mới và Thay đổi Đến ngày của chương trình trước đó.
Thời gian bỏ ra ở từng giai đoạn	
Kế hoạch	Đối với Tổng thời gian phát triển (Total Development time), nhân LOC Mới & Thay đổi với Phút/LOC Đối với Thời gian tối đa, nhân độ lớn tối đa (Maximum size) với Phút/LOC. Đối với Thời gian tối thiểu, nhân độ lớn tối thiểu (Minimum size) với Phút/LOC. Từ bản tổng kết kế hoạch dự án của chương trình gần nhất, tìm giá trị Đến ngày % cho mỗi pha. Sử dụng Đến ngày % từ chương trình trước đó, tính toán thời gian kế hoạch cho mỗi pha.
Thực tế	Sau khi hoàn tất, nhập thời gian thực tế tính theo phút trong mỗi pha phát triển.

	Lấy dữ liệu này từ Bản ghi nhận thời gian
Đến ngày	Với mỗi pha, điền vào tổng thời gian thực tế và thời gian Đến ngày từ chương trình gần nhất.
Đến ngày %	Với mỗi pha, điền vào (thời gian Đến ngày * 100) / Tổng thời gian Đến ngày.
Sai sót mắc phải	
Kế hoạch	***Trước khi phát triển, ước lượng tổng số sai sót sẽ có thể mắc phải trong chương trình: sai sót/KLOC kế hoạch * LOC Mới và Thay đổi kế hoạch của chương trình / 1000*** ***Ví dụ, với sai sót/KLOC kế hoạch là 75.9 và LOC Mới và Thay đổi là 75, tổng số sai sót kế hoạch = 75.9*75/1000 = 5.96, làm tròn thành 6.*** ***Trước khi phát triển, ước lượng sai sót mắc phải trong từng pha bằng cách sử dụng tổng sai sót ước lượng và sự phân bố sai sót mắc phải Đến ngày % của chương trình trước.***
Thực tế	Sau khi phát triển, tìm và điền số lượng sai sót thực tế mắc phải trong mỗi pha
Đến ngày	Với mỗi pha, nhập vào tổng số sai sót thực tế và sai sót Đến ngày từ chương trình gần nhất.
Đến ngày %	Với mỗi pha, nhập vào 100*(Sai sót Đến ngày của pha đó)/(Tổng sai sót Đến ngày)
Sai sót loại bỏ	
Kế hoạch	***Ở dòng tổng cộng, điền vào tổng số sai sót ước lượng.*** ***Sử dụng các giá trị Đến ngày từ chương trình gần nhất, tính toán sai sót kế hoạch loại bỏ được trong mỗi pha.***
Thực tế	Sau khi phát triển, tìm và điền số lượng sai sót thực tế loại bỏ trong mỗi pha
Đến ngày	Với mỗi pha, nhập vào tổng số sai sót thực tế và sai sót Đến ngày từ chương trình gần nhất.
Đến ngày %	Với mỗi pha, nhập vào 100*(Sai sót Đến ngày của pha đó)/(Tổng sai sót Đến ngày)

Bảng 3.5.4 Chỉ dẫn cho bản tổng kết kế hoạch

Kịch bản quy trình PSP và bản tổng kết kế hoạch dự án mới giờ đây được cập nhật như trong các bảng 3.5.2 và 3.5.3. Các chỉ dẫn cho bản tổng kết kế hoạch dự án ở bảng 3.5.4. Các mục mới trong các bảng này được ***in nghiêng đậm***. Các mục được thêm vào là cột Kế hoạch cho các sai sót mắc phải, sai sót được loại bỏ và dòng sai sót/KLOC trong bản tổng kết.

3.5.6 Một ví dụ về bản tổng kết dự án

Bảng 3.5.5 đưa ra một ví dụ về bản tổng kết kế hoạch dự án của sinh viên X cho chương trình 13. Đoạn sau mô tả cậu sử dụng dữ liệu từ bản tổng kết kế hoạch dự án đã hoàn tất của chương trình 12 (ở bảng 3.5.6) để hoàn tất biểu mẫu này như thế nào. Các giá trị khác nhau trong các bảng này được gán nhãn với các ký tự abc để giúp nhận diện ra nguồn xuất phát của dữ liệu.

Sinh viên		Sinh viên X				Ngày		18/11/96		
Chương trình						Chương trình #		13		
Người hướng dẫn		Thầy Z				Ngôn ngữ		Ada		
Tóm tắt		**Kế hoạch**		**Thực tế**				**Đến ngày**		
Phút/LOC	b	5.92	p	4.87			x	5.73		
LOC/Giờ		10.14	q	12.32			y	10.47		
Sai sót/KLOC	i	94.79	r	106.4			z	96.90		
Hiệu suất										
A/FR										
Kích thước chương trình (LOC)										
Tổng mới và thay đổi	a	58	o	47			w	258		
Kích thước tối đa	a	72								
Kích thước tối thiểu	a	41								
Thời gian trong pha (phút)		**Kế hoạch**		**Thực tế**		**Đến ngày**			**Đến ngày %**	
Lên kế hoạch	f	18	m	22	s	88	u	6.0		
Thiết kế	f	35	m	24	s	151	u	10.2		
Cài đặt	f	149	m	93	s	637	u	43.1		
Xem lại mã	f	20	m	37	s	111	u	7.5		
Biên dịch	f	24	m	4	s	92	u	6.2		
Kiểm thử	f	64	m	8	s	240	u	16.2		
Tổng kết	f	33	m	41	s	160	u	10.8		
Tổng cộng	c	343	m	229	s	1479	u	100		
Kích thước tối đa	d	426								
Kích thước tối thiểu	d	243								

Sai sót mắc phải	***Kế hoạch***		**Thực tế**		**Đến ngày**		**Đến ngày %**	***Sai sót/giờ***
Lên kế hoạch								
Thiết kế	l 1			t	4	u	16.0	
Cài đặt	l 5	n	5	t	21	u	84.0	
Xem lại mã								
Biên dịch								
Kiểm thử								
Tổng cộng	j 6	n	5	t	25	u	100.0	

Sai sót loại bỏ	***Kế hoạch***		**Thực tế**		**Đến ngày**		**Đến ngày %**	***Sai sót/giờ***
Lên kế hoạch								
Thiết kế								
Cài đặt								
Xem lại mã	l 2	n	3	t	8	u	32.0	
Biên dịch	l 3	n	2	t	12	u	48.0	
Kiểm thử	l 1			t	5	u	20.0	
Tổng cộng	j 6	n	5	t	25	u	100.0	

Bảng 3.5.5 Một ví dụ bản tổng kết kế hoạch dự án PSP

Sinh viên	Sinh viên X			Ngày	11/11/96			
Chương trình				Chương trình #	12			
Người hướng dẫn	Thầy Z			Ngôn ngữ	Ada			
Tóm tắt	**Kế hoạch**		**Thực tế**		**Đến ngày**			
Phút/LOC	6.30		4.93	b	5.92			
LOC/Giờ	9.52		12.17		10.14			
Sai sót/KLOC				g	94.79			
Hiệu suất								
A/FR								
Kích thước chương trình (LOC)								
Tổng mới và thay đổi	51		58	h	221			
Kích thước tối đa	65							
Kích thước tối thiểu	37							
Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**		**Đến ngày**		**Đến ngày %**		
Lên kế hoạch	16	18	s	66	e	5.3		
Thiết kế	27	44	s	127	e	10.2		
Cài đặt	146	104	s	544	e	43.5		
Xem lại mã	12	38	s	74	e	5.9		
Biên dịch	26	11	s	88	e	7.0		
Kiểm thử	68	29	s	232	e	18.6		
Tổng kết	26	42	s	119	e	9.5		
Tổng cộng	321	286	s	1250		100.0		
Kích thước tối đa	410							
Kích thước tối thiểu	233							
Sai sót mắc phải	***Kế hoạch***	**Thực tế**		**Đến ngày**		**Đến ngày %**	***Sai sót/giờ***	
Lên kế hoạch								
Thiết kế		1	t	4	k	20.0		
Cài đặt		5	t	16	k	80.0		
Xem lại mã								
Biên dịch								
Kiểm thử								
Tổng cộng		6	i	20		100.0		
Sai sót loại bỏ	***Kế hoạch***	**Thực tế**		**Đến ngày**		**Đến ngày %**	***Sai sót/giờ***	
Lên kế hoạch								
Thiết kế								
Cài đặt								
Xem lại mã		2	t	5	k	25.0		
Biên dịch		2	t	10	k	50.0		
Kiểm thử		2	t	5	k	25.0		
Tổng cộng		6	t	20		100.0		

Bảng 3.5.6 Bản kế hoạch chương trình 12 của sinh viên X

3.5.6.1 Điền dữ liệu kế hoạch:

1. Sinh viên X đầu tiên ước lượng kích thước của chương trình mới là 58 LOC (a) và LOC tối đa và tối thiểu là 72 (a) và 41 (a).

2. Kế đó, cậu ta nhìn vào bản tổng kết kế hoạch dự án của chương trình 12 để lấy giá trị phút/LOC Đến ngày là 5.92 (b), sử dụng giá trị này là tốc độ kế hoạch cho chương trình 13.
3. Với tổng số LOC ước lượng là 58 (a), tổng thời gian ước lượng của dự án là 58*5.92=343.36 hay 343 phút (c).
4. Sinh viên X sau đó tính thời gian phát triển tối đa và tối thiểu bằng cách nhân kích thước tối đa và tối thiểu (a) với 5.92 (b) để có được giá trị 426 và 243 (d) phút.
5. Với thời gian trong pha, cậu lấy thời gian Đến ngày % từ bản tổng kết của chương trình 12 (e) nhân với tổng thời gian ước lượng là 343 phút (c) và chia cho 100, các kết quả ở (f). Để thuận tiện, cậu làm tròn các con số này.
6. Với tổng số sai sót mắc phải và loại bỏ được, sinh viên X lấy được giá trị sai sót/KLOC Đến ngày trong bản tổng kết của chương trình 12 là 94.79(g) (giá trị này được tính từ LOC Đến ngày (h) và Sai sót Đến ngày (i)).
7. Với sai sót/KLOC 94.79 và LOC kế hoạch 58 (a), tổng số sai sót dự đoán là 94.79*58/1000 = 5.50, hay khoảng 6 sai sót.
8. Dựa trên các giá trị Đến ngày % về các sai sót mắc phải và loại bỏ được từ chương trình 12 (k), cậu tính được số sai sót dự đoán mắc phải và loại bỏ được ở mỗi pha (l).

Vậy là dự đoán sai sót cho chương trình 13 đã hoàn thành.

3.5.6.2 Điền dữ liệu thực tế

Sau khi phát triển chương trình, sinh viên Y điền dữ liệu thực tế trong biểu mẫu như sau:

1. Điền các giá trị thực tế về thời gian trong pha từ bản ghi ghi chép thời gian (m).
2. Điền các giá trị thực tế về sai sót mắc phải và sai sót loại bỏ được từ bản ghi ghi chép sai sót (n).
3. Cậu đếm được 47 LOC Mới và Thay đổi trong chương trình đã hoàn tất và nhập con số đó dưới mục Thực tế (o).
4. Với các dữ liệu này, cậu tính được số phút/LOC thực tế bằng cách lấy tổng số phút là 229 (m) chia cho LOC Mới và Thay đổi là 47 (o), được 229/47=4.87 (p).
5. Cậu cũng tính được LOC/Giờ thực tế từ số phút/LOC thực tế (p) là 60/4.87=12.32 (q).
6. Cậu tính sai sót/KLOC thực tế từ tổng sai sót thực tế (n) và số LOC thực tế (o) là 1000*5/47=106.4 (r).

7. Kế đó cậu tính các giá trị Đến ngày cho Thời gian trong pha bằng cách cộng thời gian của pha trong dự án này với thời gian Đến ngày của dự án trước (s).
8. Cậu cũng tính các giá trị Đến ngày cho Sai sót mắc phải và Sai sót loại bỏ được bằng cách cộng sai sót thực tế trong mỗi pha với sai sót Đến ngày của dự án trước (t).
9. Các giá trị Đến ngày % cũng được tính (u).
10. Kế đó cậu tính giá trị Đến ngày cho phút/LOC bằng cách chia Tổng thời gian Đến ngày là 1479 phút (s) cho Tổng LOC Mới và Thay đổi Đến ngày là 258 (w), bằng 1479/258=5.73 (x).
11. Sử dụng giá trị 5.73 này (x), cậu tính được LOC/Giờ Đến ngày là 60/5.73=10.47 (y).
12. Cuối cùng, cậu tính Sai sót/KLOC Đến ngày bằng cách nhân 1000 với Tổng sai sót mắc phải Đến ngày (t) và chia cho Tổng LOC Mới và Thay đổi Đến ngày (w), bằng 1000*25/258=96.90 (z).

Chú ý rằng các con số Đến ngày và Đến ngày % giữ số liệu về tổng thời gian đang hoạt động và sự phân bố sai sót. Khi có nhiều kinh nghiệm hơn, bạn có thể sẽ nhận ra rằng hiệu suất của bạn và các mức độ sai sót thay đổi quá đáng kể đến nỗi dữ liệu về các chương trình trước đó thì không còn là chỉ dẫn hữu ích cho việc lên kế hoạch hiện tại nữa. Đến lúc này, có thể bạn sẽ mong muốn tính toán lại tất cả các giá trị Đến ngày và Đến ngày % mà không cần có các chương trình trước đó nữa.

3.6 Tính kinh tế của việc loại bỏ sai sót

3.6.1 Vấn đề loại bỏ sai sót

Loại bỏ sai sót đắt đỏ vì các sai sót rất khó tìm thấy và sửa chữa. Khi chúng ta xây dựng các hệ thống phần mềm phức tạp hơn và lớn hơn, vấn đề này sẽ chỉ càng trở nên tồi tệ hơn. Kích thước và độ phức tạp của các hệ thống phần mềm đã tăng thêm khoảng 10 lần cứ mỗi 10 năm. Trong khi một máy in laser hồi đầu chỉ cần khoảng 20.000LOC trong các chương trình hỗ trợ của nó thì phiên bản mới nhất cần hơn 1 triệu LOC. Cách đây 10 năm, xe ô tô chẳng có phần mềm nào thì ngày nay các xe hơi mới nhất chứa tới nhiều ngàn dòng lệnh phần mềm. Vì kích thước và độ phức tạp của phần mềm sẽ ngày càng tiếp tục tăng lên nữa, do đó việc loại bỏ sai sót sẽ càng trở nên đắt đỏ và tốn nhiều thời gian hơn nữa.

Để hiểu được và kiểm soát được chi phí sai sót, cần phải đo hiệu quả của việc loại bỏ sai sót. Một phép đo kiểu này là số lượng sai sót loại bỏ được trong 1 giờ. Một phép đo hữu ích khác là hiệu suất loại bỏ sai sót. Hiệu suất (yield) tính phần trăm của sai sót tìm được bởi một phương pháp loại bỏ sai sót. Vì vậy, nếu một sản phẩm bao gồm 100 sai sót khi vào khâu kiểm thử và phát hiện ra 45 sai sót trong khâu này thì hiệu suất kiểm thử sẽ là 45%. Khi bạn biết được hiệu suất và tốc độ loại bỏ của mỗi phương pháp loại bỏ sai sót thì bạn có thể quyết định được cách tốt nhất để tìm và sửa chữa sai sót.

3.6.2 Sự tiết kiệm của việc loại bỏ sai sót

Một sự hiểu biết về tốc độ loại bỏ sai sót thì quan trọng trong các dự án lớn và nó có thể có ích cho cá nhân các kỹ sư phần mềm. Khi phát triển một chương trình kích thước vừa phải, và giả sử như dữ liệu của bạn tương tự với các kỹ sư khác, bạn sẽ mắc phải khoảng 100 sai sót/KLOC và tìm thấy khoảng 50 sai sót mỗi KLOC trong biên dịch và tìm ra 40 sai sót nữa cho mỗi KLOC trong kiểm thử đơn vị.

	500 LOC		**10000 LOC**	
	Không có PSP	*Với PSP*	*Không có PSP*	*Với PSP*
Sai sót				
Tổng số sai sót	50	25	1000	500
Tìm thấy trong xem lại code	0	15	0	300
Sai sót còn lại	50	10	1000	200
Tìm thấy trong biên dịch	25	5	500	100
Tìm thấy trong kiểm thử đơn vị	20	4	400	80
Sai sót còn lại	5	1	100	20
Thời gian (giờ)				
Thời gian xem lại code	0	2.5	0	50
Thời gian biên dịch	2	0.5	40	10
Thời gian kiểm thử đơn vị	10	2	200	40
Thời gian loại bỏ sai sót cá nhân	12	5	240	100
Thời gian kiểm thử tích hợp và kiểm thử hệ thống			1000	200
Tổng thời gian loại bỏ sai sót	12	5	1240	300

Bảng 3.6.1 Ví dụ về việc mắc phải và loại bỏ sai sót

Như trong bảng trên, điều này có nghĩa rằng 1 chương trình 500 LOC sẽ có tổng cộng khoảng 50 sai sót. Trong số này, 25 sai sót sẽ được tìm thấy trong biên dịch và, nếu may mắn, 20 sai sót nữa sẽ được tìm thấy trong kiểm thử đơn vị. Nếu bạn không thực hiện xem lại code, bạn có thể mất khoảng 2 giờ để biên dịch sạch lỗi và tìm ra 25 sai sót này.

Kiểm thử đơn vị sẽ chiếm khoảng 10 giờ để tìm thấy 20 sai sót kia. Tổng cộng cần có 12 giờ đồng hồ để loại bỏ sai sót.

Sau khi học PSP, bạn sẽ chỉ mắc phải khoảng 50 sai sót/KLOC. Nếu xem lại code, bạn có thể sẽ tìm ra được 60% đến 70% các sai sót này trước khi biên dịch lần đầu tiên. Lúc này, biên dịch chỉ chiếm khoảng 30 phút và chỉ còn 4 đến 5 sai sót cần phải tìm ra trong kiểm thử đơn vị. Kiểm thử vì vậy cần khoảng 2 giờ. Giả sử bạn bỏ ra 2.5 giờ trong xem lại code, tổng thời gian loại bỏ sai sót bây giờ chỉ là 5 giờ, tiết kiệm được 7 giờ.

Dường như không đáng để nỗ lực nhiều chỉ để cắt giảm thời gian loại bỏ sai sót xuống 7 giờ đồng hồ với một công việc mà việc thực hiện với khoảng vài ngày, nhưng hãy xem xét đến việc những con số này tăng lên như thế nào khi kích thước chương trình tăng lên.

Để thấy được điều này có ý nghĩa như thế nào trong thực tiễn, giả sử bạn và 4 kỹ sư khác định phát triển một sản phẩm phần mềm 50000 LOC. Mỗi người dự định phát triển một thành phần 10000 LOC và sau đó tích hợp lại và kiểm thử toàn bộ hệ thống. Dựa vào dữ liệu điển hình của các kỹ sư, bạn và các đồng sự có thể sẽ mắc phải khoảng 100 sai sót/KLOC, nghĩa là sẽ có 5000 sai sót cần được tìm thấy và sửa chữa. Sử dụng các tỉ lệ đã nói ở trên, có khoảng 2500 sai sót sẽ được tìm thấy trong biên dịch và 2000 sai sót trong kiểm thử đơn vị. Như vậy còn 500 sai sót cần tìm thấy trong kiểm thử tích hợp và kiểm thử hệ thống. Giả sử như sản phẩm của bạn đơn giản hơn Microsoft's NT, bạn có thể tìm được các sai sót này với chi phí trung bình chỉ khoảng 10 giờ. Loại bỏ các sai sót này vì vậy sẽ chiếm mất của đội gồm 5 kỹ sư của bạn khoảng 5000 giờ. Nếu cả đội không làm gì khác trong 40 giờ một tuần thì sẽ chiếm hết 6 tháng. Tuy nhiên, nếu nhóm của bạn xem lại theo cá nhân và sau đó thanh-tra-nhóm chương trình của bạn thì bạn có thể tiết kiệm được ít nhất 5 tháng kiểm thử. Với một dự án 2 năm, sự khác biệt trong thời gian kiểm thử này sẽ là sự khác nhau giữa chuyển giao sản phẩm đúng thời hạn và việc bị trễ một cách trầm trọng.

3.6.3 Tính số sai sót/giờ và hiệu suất trong bản tổng kết kế hoạch

Công thức tính như sau:

Số sai sót mắc phải/giờ Đến ngày = 60*(Sai sót mắc phải Đến ngày trong pha)/(số phút Đến ngày bỏ ra trong pha)

Tương tự:

Số sai sót loại bỏ/giờ Đến ngày = 60*(Sai sót loại bỏ Đến ngày trong pha)/(số phút Đến ngày bỏ ra trong pha)

Sinh viên	Sinh viên X		Ngày	28/11/96
Chương trình			Chương trình #	14
Người hướng dẫn	Thầy Z		Ngôn ngữ	Ada

Tóm tắt	**Kế hoạch**	**Thực tế**	**Đến ngày**
Phút/LOC	5.73	4.65	5.48
LOC/Giờ	10.47	12.90	10.95
Sai sót/KLOC	96.90	77.9	92.53
Hiệu suất	33.3	80.0	40.0
A/FR			
Kích thước chương trình (LOC)			
Tổng mới và thay đổi	67	77	335
Kích thước tối đa	85		
Kích thước tối thiểu	49		

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	**Đến ngày**	**Đến ngày %**
Lên kế hoạch	23	32	120	6.5
Thiết kế	39	44	195	10.6
Cài đặt	166	155	792	43.1
Xem lại mã	29	34	145	7.9
Biên dịch	24	8	100	5.5
Kiểm thử	62	39	279	15.2
Tổng kết	41	46	206	11.2
Tổng cộng	384	358	1837	100.0
Kích thước tối đa	487			
Kích thước tối thiểu	281			

Sai sót mắc phải	***Kế hoạch***	**Thực tế**	**Đến ngày**		**Đến ngày %**	***Sai sót/giờ***
Lên kế hoạch						
Thiết kế	1	1	5	k	16.1	1.54
Cài đặt	5	4	25	k	80.7	1.89
Xem lại mã						
Biên dịch		1	1		3.2	
Kiểm thử						
Tổng cộng	6	5	31		100.0	

Sai sót loại bỏ	***Kế hoạch***	**Thực tế**	**Đến ngày**	**Đến ngày %**	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã	2	4	12	38.7	4.97
Biên dịch	3	1	13	41.9	7.80
Kiểm thử	1	1	6	19.4	1.29
Tổng cộng	6	5	31	100.0	

Bảng 3.6.2 Ví dụ bản tổng kết kế hoạch dự án

Khi loại bỏ sai sót trong một pha, bạn cũng sẽ muốn biết có bao nhiêu sai sót được tìm thấy và bao nhiêu bị bỏ sót. Không có cách nào để biết được điều này vào lúc đang phát triển, nhưng dữ liệu lịch sử có thể giúp bạn khá tốt về điều này bằng cách tính toán và theo dõi chỉ số hiệu suất. Với PSP, hiệu suất quy trình được định nghĩa là phần trăm sai sót tìm thấy trước khi biên dịch và kiểm thử lần đầu. Bảng trên là ví dụ minh họa về cách tính các số liệu trên.

3.6.4 Tăng tỉ lệ loại bỏ sai sót

Bạn có thể cải tiến nhanh chóng tỉ lệ loại bỏ sai sót bằng cách thực hiện xem lại code. Tuy nhiên, một khi bạn đạt được các lợi ích cải tiến ban đầu này, việc cải tiến xa hơn nữa sẽ khó khăn hơn. Bởi vì thử thách đối đầu với các kỹ sư phần mềm tăng dần theo mỗi năm, bạn không được phép ngừng cải tiến. Một số lời khuyên cho việc tiếp tục cải tiến tỉ lệ loại bỏ sai sót của bạn là:

- Tập trung vào hiệu suất trước tiên. Hãy nhớ rằng, mục tiêu là loại bỏ tất cả các sai sót. Mục tiêu đầu tiên của bạn vì vậy là phải đạt được hiệu suất 70% hay nhiều hơn.
- Thực hiện xem lại code trước khi biên dịch lần đầu. Để đạt được hiệu suất cao nhất có thể, sử dụng trình biên dịch để kiểm tra chất lượng của việc xem lại code của bạn.
- Một khi bạn đã đạt được hiệu suất đáng kể, sử dụng các phương pháp ở các phần trước để cải tiến tốc độ xem lại. Theo dõi xem checklist tìm thấy và bỏ sót sai sót ở đâu và thực hiện điều chỉnh định kỳ. Nếu có một số bước không tìm thấy hay không bỏ sót nhiều sai sót, hãy nghĩ đến việc loại chúng ra. Tuy nhiên, khi bạn vẫn tiếp tục bỏ sót sai sót, hãy xem xét việc thêm vào một bước trong checklist để một cách đặc biệt nhằm vào loại này.
- Nếu bạn cứ làm hoài một việc thì đừng mong có một kết quả khác đi. Nếu bạn không thay đổi checklist, bạn sẽ vẫn tiếp tục bỏ sót cũng các sai sót đó.

Hãy tiếp tục thu thập dữ liệu sai sót, tính toán hiệu suất, tỉ lệ mắc phải sai sót và loại bỏ sai sót. Sau đó, theo dõi các dữ liệu này và thí nghiệm với nhiều phương pháp khác nhau để tìm được đúng phương pháp giúp bạn cải tiến.

3.6.5 Giảm tỉ lệ mắc phải sai sót

Để giảm tỉ lệ mắc phải sai sót thì khó hơn vì bạn mắc phải sai sót trong mọi phần của quy trình. Vì vậy một số cách để giảm tỉ lệ sai sót như sau:

Ghi nhận lại tất cả sai sót của bạn. Có ý thức về các sai sót của mình, bạn sẽ làm việc cẩn thận hơn và sẽ giảm được số lượng sai sót mắc phải.

Tạo ra các thiết kế tốt hơn. Tạo ra các thiết kế hoàn chỉnh hơn và được sưu liệu tốt, bạn có thể cải tiến chất lượng chương trình theo 2 phương diện. Đầu tiên, điều này sẽ thật sự ngăn chặn các sai sót mà một bản thiết kế không hoàn chỉnh hay khó hiểu gây ra. Thứ hai, một thiết kế hoàn chỉnh hơn sẽ tiết kiệm thời gian cài đặt. Bởi vì tỉ lệ mắc phải sai sót trong thiết kế thì thấp hơn trong cài đặt nên điều này cũng sẽ giảm được sai sót.

Sử dụng các phương pháp tốt hơn. Vì sai sót có thể bị mắc phải trong bất cứ pha nào, cải tiến cách bạn phát triển yêu cầu, đặc tả, thiết kế, trường hợp kiểm thử và mã nguồn đều giúp để giảm sai sót. Bằng cách sử dụng PSP và đánh giá công việc của bạn bằng các phương pháp này, bạn có thể thấy chúng làm tốt như thế nào.

Sử dụng các công cụ tốt hơn. Nếu một công cụ tiết kiệm thời gian thì nó sẽ giảm số sai sót mà bạn mắc phải. Các công cụ phần mềm mới được phát triển hàng năm và dữ liệu PSP sẽ cho phép bạn đo lường và đánh giá chúng. Khi đó bạn có thể biết được công cụ nào giúp bạn và đến mức nào.

3.7 Các sai sót thiết kế

3.7.1 Tính tự nhiên của sai sót thiết kế

Nhiều sai sót tìm thấy trong kiểm thử hầu như chắc chắn là bị mắc trong pha cài đặt. Điều này hàm ý vấn đề sai sót chủ yếu là các lỗi cài đặt đơn giản. Thật ra, với các lỗi tìm thấy trong kiểm thử, 14 sinh viên trong một lớp của tác giả Humphrey đã mắc lại một nửa số lỗi trong cài đặt như họ đã từng mắc trong thiết kế. Vì trình biên dịch đã loại bỏ hầu hết các sai sót về lỗi cú pháp nên điều này khá ngạc nhiên. Tuy nhiên, như bạn có thể thấy ở bảng 2.16.1, tỉ lệ này thay đổi khi sinh viên học PSP. Trong số sai sót tìm thấy trong kiểm thử, các sinh viên khi mới bắt đầu học PSP mắc sai sót trong cài đặt nhiều gấp rưỡi lần so với trong thiết kế. Đến cuối khoá học, con số này nhiều hơn chỉ là khoảng 14%. Ta thấy họ giảm số lượng sai sót mắc phải cả trong 2 pha, nhưng việc cải tiến có phần hơi ít với các sai sót kiểm thử mắc phải trong thiết kế (xem %Giảm trong bảng dưới).

Nếu xem xét loại sai sót,chúng ta lại có một vấn đề khác để bàn đến nữa. Trong PSP, loại sai sót được sắp xếp theo độ phức tạp của vấn đề với loại 10, 20 là đơn giản nhất và 90, 100 là các loại phức tạp nhất. Vì vậy chúng ta có thể nói rằng từ loại 10 đến 40 là đơn giản hơn hay gần như là sai sót cài đặt (codinglike) và loại 50 đến 100 thì phức tạp hơn hay là loại sai sót gần như thiết kế (designlike).

	Sai sót thiết kế/KLOC	Sai sót cài đặt/KLOC	Tỉ lệ của sai sót cài đặt với thiết kế
Bài tập 1	8.66	12.99	1.50
Bài tập 10	5.05	5.77	1.14
%Giảm	41.67%	55.56%	

Bảng 3.7.1 Các lỗi kiểm thử bị mắc trong các pha thiết kế và cài đặt

Nếu chúng ta phân loại các loại sai sót này theo loại sai sót, chúng ta được bảng 3.7.2. Ở đây, phần lớn sai sót kiểm thử là loại thiết kế, nhưng nhiều trong số chúng là mắc phải trong cài đặt. Mặc dù các kỹ sư này đã giảm mạnh số lượng sai sót mắc phải nhưng họ vẫn phạm nhiều lỗi thiết kế trong quá trình cài đặt. Như ta đã biết trong chương trước, cũng các kỹ sư này, mắc từ 5 đến 8 sai sót mỗi giờ trong cài đặt và chỉ 1 đến 3 sai sót mỗi giờ trong thiết kế mà thôi. Từ đó ta thấy cần phải có một cách hiệu quả để ít mắc sai sót thiết kế hơn là không thiết kế trong pha cài đặt nữa. Phần này sẽ giải quyết vấn đề này.

	Loại thiết kế Sai sót/KLOC	Loại cài đặt Sai sót/KLOC	% sai sót của loại thiết kế
Bài tập 1			
Pha thiết kế	7.22	1.44	83.33%
Pha cài đặt	9.38	3.61	72.22%
Tổng cộng	16.59	5.05	76.67%
Bài tập 10			
Pha thiết kế	3.61	1.44	71.43%
Pha cài đặt	3.61	2.16	62.50%
Tổng cộng	7.22	3.61	66.67%
% giảm			
Pha thiết kế	50.00%	0%	
Pha cài đặt	61.54%	40.00%	
Tổng cộng	56.52%	28.57%	

Bảng 3.7.2 Các loại sai sót kiểm thử phân loại theo pha bị mắc

3.7.2 Nhận dạng các sai sót thiết kế

Không có cách đơn giản và khách quan nào để định nghĩa ra các sai sót thiết kế. Có 2 lựa chọn là:

- Định nghĩa tất cả các lỗi mắc trong pha thiết kế là sai sót thiết kế.

- Định nghĩa những loại thiết kế liên quan đến vấn đề lập trình hàm, logic, biểu diễn và thời gian sai sót lỗi thiết kế.

Ở đây, chúng ta sẽ theo cách định nghĩa thứ 2.

3.7.3 Thiết kế là gì?

Trong việc xem xét các sai sót thiết kế, điều quan trọng là định nghĩa những gì ta muốn đề cập với từ "thiết kế". Điều này hóa ra lại không dễ dàng, vì mọi thứ về cấu trúc và thực thi của một chương trình đều liên quan đến thiết kế của nó. Nó bao gồm luồng chương trình, cấu trúc và bản chất của ngôn ngữ xây dựng, và ngay cả hệ thống chấm câu của mã nguồn.

Vấn đề là thiết kế là một vấn đề về cách nhìn (perspective). Có thể thiết kế chi tiết và có thể thiết kế ở một mức độ cao. Khi phát triển chương trình, bạn thực hiện công việc thiết kế ở hầu hết mỗi bước. Khi thực hiện các chi tiết của một câu lệnh case hay vòng lặp, bạn chỉ rõ thiết kế chi tiết. Khi định nghĩa một cấu trúc dữ liệu hay thiết lập một giao diện module, bạn sẽ thực hiện ở một mức độ nào đó cao hơn.

Sự phân tách giữa thiết kế và cài đặt vì vậy không bị bó buộc. Việc chia các hoạt động theo mức độ chi tiết liên quan đến công việc giúp tập trung vào đúng vấn đề theo đúng trật tự. Ví dụ, kỹ sư phạm ít lỗi hơn và hiệu quả hơn khi họ nghĩ qua thiết kế trước khi thực thi chúng. Điều này không có nghĩa họ thực hiện tất cả công việc của mình từ trên xuống dưới mà họ thường bắt đầu với khái niệm ở mức độ cao trước khi đào sâu vào chi tiết.

Thuật ngữ sử dụng để mô tả cách tiếp cận khái niệm ở mức cao này là *khái niệm trừu tượng (abstraction)*. Vì vậy khi nói về một số khái niệm trừu tượng như căn bậc hai, ta đang nói về một chức năng chương trình tính toán căn bậc hai. Tuy nhiên ở mức này ta không quan tâm về tất cả các chi tiết căn bậc hai này được tính toán như thế nào. Bằng cách theo chiến thuật khái niệm trừu tượng này, có thể tạo ra một thiết kế mức cao hoàn chỉnh trước khi đi sâu vào chi tiết của việc mỗi khái niệm trừu tượng hay mỗi hàm được xây dựng như thế nào.

Ví dụ, bạn có thể bắt đầu thiết kế một chương trình đơn giản bằng cách định nghĩa 4 khái niệm trừu tượng chức năng hay các thủ tục: Input, File, Calculate và Output. Khi đó bạn có thể thiết kế quy trình chính của chương trình sử dụng những khái niệm trừu tượng này. Bước kế tiếp tất nhiên là thiết kế mỗi khái niệm trừu tượng này. Với chương trình lớn hơn, bạn có thể có một số mức độ của khái niệm trừu tượng và ngay cả xây dựng một thư

viện các khái niệm trừu tượng được phát triển trước đây để sử dụng trong các chương trình sau này.

3.7.4 Quy trình thiết kế

Trong thiết kế, kỹ sư có kinh nghiệm thường di chuyển linh động giữa các mức thiết kế vì họ đang giải quyết với nhiều khái niệm trừu tượng mang tính chức năng. Trước khi cảm thấy thoải mái khi sử dụng các khái niệm trừu tượng này trong một thiết kế mức cao, họ thường phải hiểu họ chúng làm việc như thế nào. Nếu đã từng sử dụng các hàm như thế này, họ có thể trì hoãn việc định nghĩa các chi tiết. Nhưng nếu chưa, họ thường phải dừng lại để hoàn thành thiết kế chi tiết của chúng. Thậm chí họ có thể viết và kiểm thử một prototype trước khi đủ thỏa mãn để tiếp tục với thiết kế mức cao. Làm như vậy vì các khái niệm có vẻ đơn giản thường có các phức tạp khó thấy. Đôi khi các thiết kế hệ thống dựa trên các khái niệm trừu tượng có vẻ đơn giản nhưng được định nghĩa kém thì toàn bộ cách tiếp cận thiết kế sau này sẽ không có khả năng thực thi.

Khái niệm trừu tượng rất có ích khi giải quyết các hàm được định nghĩa và hiểu rõ. Tuy nhiên, rất dễ gặp rắc rối khi sử dụng các tiếp cận quan niệm này để thực hiện một công việc thiết kế khó khăn.

Vì vậy việc phân biệt giữa thiết kế và cài đặt cũng như đặc tả cái gì cấu thành một thiết kế hoàn chỉnh là tùy ý. Hơn nữa cũng quan trọng để phân biệt giữa quy trình thiết kế và pha thiết kế cụ thể trong PSP. Pha thiết kế là khi bạn tạo ra thực thể mà bạn gọi là thiết kế. Quy trình thiết kế khi đó mô tả công việc bạn thực hiện để tạo ra sản phẩm thiết kế hay thực thể này.

3.7.5 Nguyên nhân của sai sót thiết kế

Các sai sót thiết kế gây ra bởi một số vấn đề:

- *Đầu tiên là một sai lầm thiết kế*. Bạn đã suy nghĩ thận trọng vấn đề nhưng sau đó lại quyết định một thiết kế sai. Bất chấp nguyên nhân là gì, bạn đã đưa ra một quyết định thiết kế tinh táo nhưng lại không đúng.
- *Nguyên nhân thứ hai là biết thiết kế như thế nào nhưng lại tạo ra một lỗi đơn giản*. Những lỗi như thế này là phổ biến nhất khi bạn vội vã hay quá mệt mỏi.
- *Nguyên nhân thứ ba là hiểu sai yêu cầu*. Thiết kế của bạn đúng theo cách nó thực hiện chức năng mà bạn dự định, tuy nhiên bạn lại xây dựng chức năng sai.

- *Cuối cùng là vấn đề được gọi là "hiểu theo nghĩa đen" (literal intepratation), không hiểu ngữ cảnh hệ thống*. Đây là một vấn đề trong phần mềm vì có nhiều quyết định thiết kế liên quan đến các chi tiết ảnh hưởng đến người sử dụng hệ thống. Khi người thiết kế không có kiến thức rõ về ngữ cảnh người sử dụng, họ có khả năng sẽ diễn dịch các yêu cầu không đúng đắn. Mặc dù các hệ thống này đáp ứng được tất cả các yêu cầu đã định nhưng chúng thường bất tiện và đôi khi không thể sử dụng được.

Điều lạ là các sai sót thiết kế thường là do quá tự tin. Ví dụ phổ biến thường thấy là một chỗ vá 1 hay 2 dòng lệnh. Vì các thay đổi này nhỏ nên kỹ sư thường nghĩ chúng đơn giản và không bỏ thời gian để thật sự hiểu vấn đề hay các tiềm ẩn trong lỗi vá. Các lỗi này có thể được xếp vào bất cứ loại nào trong các loại ở trên, nhưng thường thì chúng được xếp vào trường hợp thứ 3 hay 4: hiểu sai hay lỗi về ngữ cảnh.

3.7.6 Ảnh hưởng của sai sót thiết kế

Khi kỹ sư phần mềm đối mặt một vấn đề thiết kế thách thức, họ thường rất cẩn thận. Họ biết họ có thể phạm lỗi và cẩn thận kiểm tra công việc của mình. Tuy nhiên họ lại phạm quá nhiều lỗi đơn giản và vì các lỗi đơn giản có thể rất khó để tìm thấy, các lỗi đơn giản này là nguyên nhân gây ra hầu hết các rắc rối. Nhiều sai sót như vậy đi qua toàn bộ quá trình phát triển và quy trình kiểm thử làm ảnh hưởng và gây ra nhiều vấn đề cho người dùng hệ thống.

Thành ra những lỗi này có thể tránh được. Kỹ sư biết cái gì đã được định sẵn. Thiết kế được hiểu đúng, nhưng lại được trình bày nghèo nàn nên khi thực hiện cài đặt, người thực thi không thể thấy được cái gì đã được định sẵn. Không thích dừng lại để hỏi nhà thiết kế, người thực thi khi đó tức tốc hoàn thành bản thiết kế. Tuy nhiên, vì họ làm việc ở mức độ thực thi, họ không hiểu hết tất cả các hàm ý trong thiết kế. Vì vậy họ dễ mắc lỗi hơn. Trong khi người thiết kế biết cái gì được định sẵn, người thực thi lại không.

Bạn có thể có những hiểu sai như thế khi bạn thực thi một thiết kế do chính bạn thiết kế. Khi đưa ra thiết kế, bạn thường đạt được đến điểm mà toàn bộ thiết kế dường như rõ ràng. Nếu bạn dự định thực thi cài đặt, dường như có ít lý do để ghi lại tài liệu phần thiết kế này. Không may, sau này trong việc thực thi, bạn có thể không nhớ thiết kế đã-từng-rõ-ràng và phải tạo lại thiết kế trong suốt quá trình thực thi. Vì bạn có khả năng quên ngữ cảnh thiết kế, bạn cũng phải xây dựng lại tất cả các khái niệm và điều kiện có liên quan. Vì

quy trình tái xây dựng này dễ xảy ra sai sót nên bạn có thể tạo ra lỗi thiết kế. Tuy nhiên, nguyên nhân gây ra lỗi không phải là do thực thi tồi mà là thiết kế trình bày tồi.

3.7.7 Trình bày thiết kế

Một bản thiết kế hoàn chỉnh và rõ ràng sẽ giúp bạn giảm số lỗi mắc phải. Một khi bạn đã hình dung ra thiết kế thì cần phải trình bày nó để rõ ràng với người thực thi. Đây là một vấn đề quan trọng then chốt với các chương trình 10000 dòng lệnh hay nhiều hơn và thậm chí nó có thể là một vấn đề với module chương trình chỉ 50 hay 100 dòng lệnh.

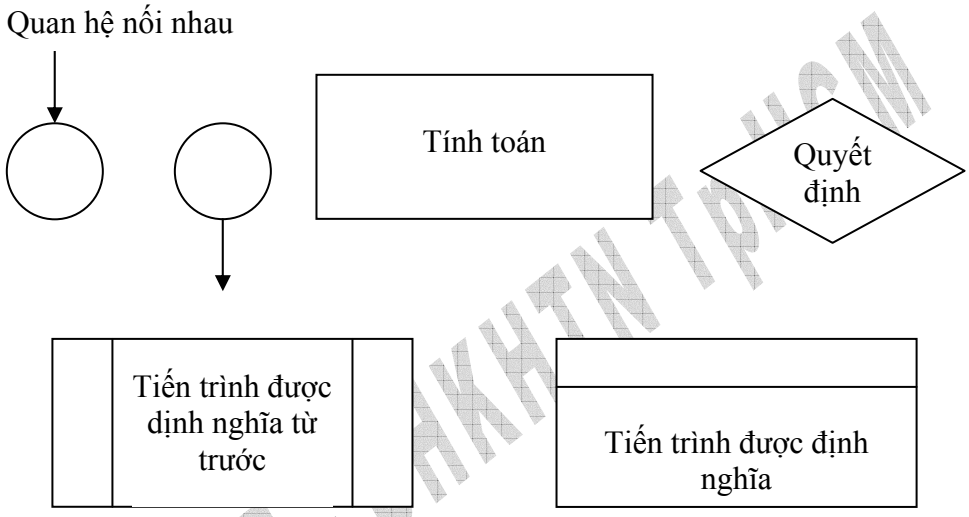
Ngoài ra việc này còn giúp bạn tiết kiệm thời gian bởi bạn thường viết code nhanh hơn rất nhiều từ một bản thiết kế rõ ràng và hoàn chỉnh so với từ một bản thiết kế mơ hồ và không hoàn chỉnh. Thời gian cài đặt ít đồng nghĩa với ít lỗi cài đặt hơn nên một thiết kế tốt sẽ đưa đến một chương trình chất lượng cao hơn.

Có 3 cách phổ biến để trình bày thiết kế: bằng đồ họa, mã giả, hay bằng toán học. Phần này sẽ trình bày các phương pháp trình diễn này. Đây là một chủ đề lớn và cần phải nói thêm nhiều về nó, tuy nhiên, bàn luận ngắn gọn về nó sẽ cho bạn một số ý tưởng về thiết kế các chương trình có kích thước module. Nó cũng sẽ cung cấp một ngữ cảnh để suy nghĩ về thiết kế khi bạn sử dụng PSP trong tương lai. Mục tiêu ở đây không phải là chỉ dẫn về các phương pháp trình bày mà là để thuyết phục bạn về tầm quan trọng của việc thực hiện một bản thiết kế và của việc trình bày thiết kế đó rõ ràng. Khi bạn được xem các phương pháp trình bày khác nhau, bạn sẽ biết tại sao chúng quan trọng và suy nghĩ đến việc thử sử dụng chúng.

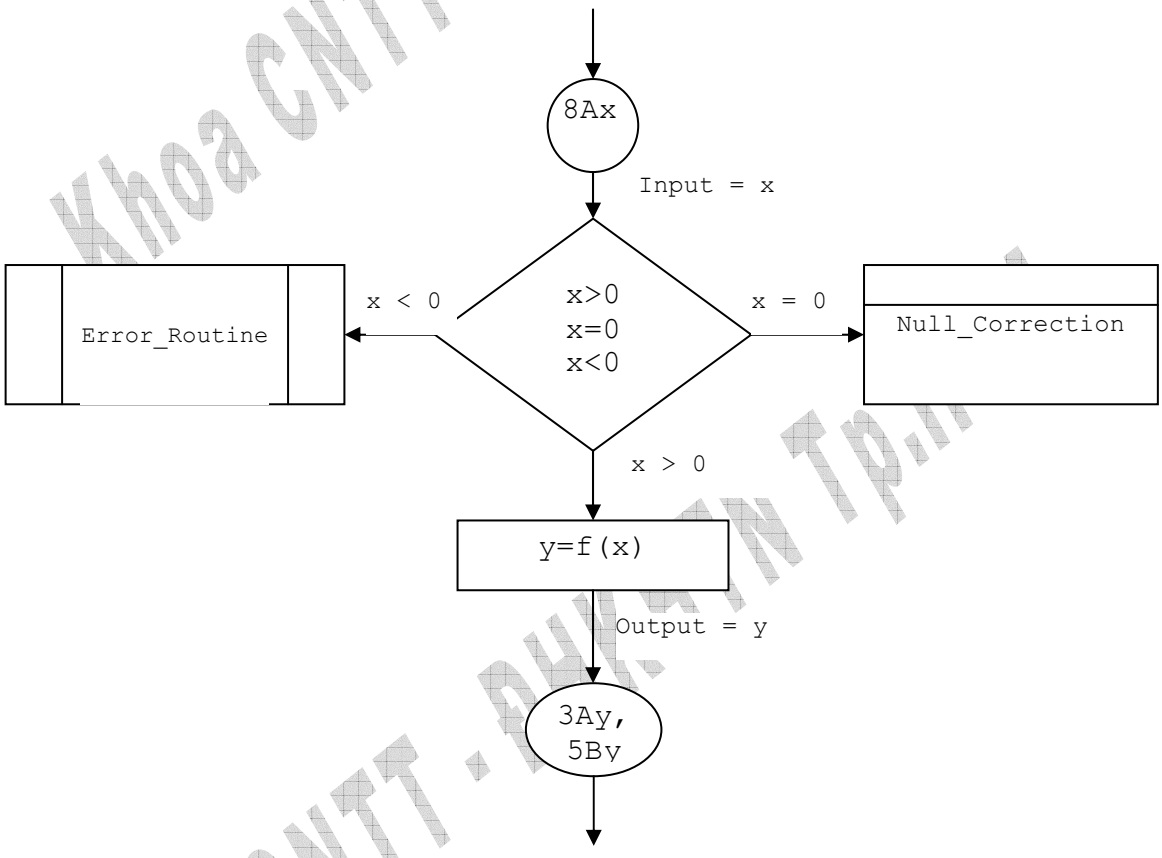
3.7.7.1 Trình bày thiết kế bằng đồ họa

Hnh ảnh thường được hiểu nhanh hơn công thức hay văn bản. Khi phần lớn vấn đề của thiết kế liên quan đến việc hiểu, bạn nên thường xuyên sử dụng trình diễn bằng đồ họa để nhấn mạnh thiết kế.

Biểu mẫu phổ biến nhất của trình bày đồ họa là biểu đồ. Đây là một biểu đồ với các chức năng chương trình được biểu diễn bởi các box và luồng chương trình logic được biểu diễn bởi các đường thẳng nối với box. Có nhiều cách để vẽ các biểu đồ này, ở đây chỉ mô tả các ký hiệu lưu đồ cơ bản như trong hình dưới.



Hình 3.7.1 Các ký hiệu của biểu đồ



Hình 3.7.2 Ví dụ biểu đồ logic

Một ví dụ đơn giản của các ký hiệu này được thể hiện trong hình 3.7.2. Ở đỉnh của biểu đồ, biểu tượng đường nối chỉ rằng input x từ 8Ax và ở cuối biểu đồ output y là 3Ay và 5By. Các ký hiệu trong các đường nối tham chiếu đến các trang khác trong thiết kế. Trong

ký hiệu này, biến x đến từ trang 8 của thiết kế tại điểm Ax. Tương tự, biến y đến điểm Ay trên trang 3 và By trên trang 5.

Hộp giữa hình 3.7.2 chỉ ra hàm quyết định, có thể thực thi bằng cấu trúc if-then-else hay một câu lệnh case. Hộp tính toán, y=f(x), mô tả một tính toán.

2 hộp bên trái và phải biểu diễn hàm được định nghĩa. Hàm Error_Routine bên trái có 2 thanh dọc để biểu diễn một hàm được định nghĩa từ trước, được sử dụng lại trong chương trình này. Hộp bên phải, Null_Correction, có 1 đường ngang để biểu diễn một hàm được định nghĩa. Đây là một khái niệm trừu tượng chức năng mới bạn đang phát triển và sẽ sử dụng bởi chương trình này khi nó hoàn tất.

Tuy nhiên, trình bày bằng biểu đồ thường hoặc không chính xác hoặc đồ sộ. Đây không phải là vấn đề cố hữu của phương pháp mà chỉ là một ví dụ khác về một cách trình bày không chính xác và không hoàn tất. Vấn đề về tính chính xác này có thể giải quyết bằng cách có một bản thiết kế được viết hoàn chỉnh và sử dụng cách biểu diễn bằng đồ họa để giúp giải thích cho logic của chương trình. Nhưng nhớ rằng, càng thêm thông tin vào biểu đồ, bạn càng làm cho chúng lộn xộn và khó hiểu hơn. Cách tiếp cận tốt nhất cho vấn đề này là sử dụng các mức khác nhau của biểu đồ mà mỗi mức bao gồm các khái niệm trừu tượng được định nghĩa ở các biểu đồ mức thấp hơn.

Với các ưu điểm của chúng, bạn nên sử dụng cách trình bày bằng đồ hoạ để minh họa thiết kế của mỗi chương trình mà bạn tạo ra. Trên thực tế, thường thì quy trình vẽ biểu đồ sẽ bộc lộ ra các mối quan hệ mà bạn chưa xem xét. Bất chấp việc bạn tạo ra nó như thế nào, các biểu đồ như thế này có thể tiết kiệm được thời gian và giảm nhầm lẫn khi thiết kế, kiểm thử, sửa chữa hay nâng cấp chương trình sau này.

3.7.7.2 Trình bày thiết kế bằng mã giả

Cách tiếp cận ở đây là viết chương trình bằng ngôn ngữ tương tự như ngôn ngữ sử dụng trong thực thi nhưng tốc ký và dễ hiểu cho các diễn đạt phức tạp. Ý tưởng là biểu diễn logic nhưng lờ đi nhiều yêu cầu về cú pháp của ngôn ngữ lập trình. Ví dụ được thể hiện trong hình 3.7.3. Bạn có thể mở rộng mô tả này bằng cách thêm các câu để mô tả cách tính hàm f(x), hay thêm các định nghĩa về kiểu, các khai báo hay các chi tiết khác nếu cần vào lúc cài đặt.

```
algorithm (Function f(x) calculation)
if (x<0)
      then (Error_Routine)
      elseif (x=0)
                  then (Null_Correction)
                        else
                              y = f(x)
end.
```

Bảng 3.7.3 Ví dụ về biểu diễn mã giả

Mô tả thiết kế mã giả là sự kết hợp của ngôn ngữ con người và cấu trúc lập trình. Thay vì logic hoàn chỉnh cho các hàm đã biết thì sử dụng các câu đơn giản. Tương tự, luồng chương trình có thể được biểu diễn bằng các câu điều kiện truyền thống, sử dụng biểu thức viết để mô tả logic. Không có chuẩn ký hiệu được chấp nhận chung cho mã giả nhưng tốt hơn hãy xây dựng ngôn ngữ tương tự với ngôn ngữ cấp cao được sử dụng. Khi đó, mã giả sẽ quen thuộc vào lúc cài đặt và sẽ cung cấp một bộ khung cho quá trình thực thi.

Vì mã giả không yêu cầu các chi tiết của ngôn ngữ lập trình nên tạo nó dễ dàng và nhanh chóng hơn để tạo hơn một chương trình nguồn hoàn chỉnh. Bạn có thể ghi lại thiết kế nhiều chi tiết như bạn muốn và sẽ dễ dàng để nắm được thiết kế khi bắt đầu phát triển.

Ưu điểm chính của mã giả là có thể chính xác như bạn mong muốn và lại dễ hiểu. Khuyết điểm chính là mã giả khi quá chi tiết sẽ trở nên khó hiểu. Đó là lý do tại sao tốt hơn ta nên sử dụng kết hợp cả mã giả và biểu đồ cho bất cứ thiết kế nào (trừ các thiết kế đơn giản).

Một vấn đề tiềm ẩn trong mã giả xảy ra bởi nó bỏ qua nhiều hệ thống dấu câu và cấu trúc dùng để định nghĩa đầy đủ ý nghĩa của chương trình nguồn. Vì vậy dễ mắc lỗi với các câu logic thông thường. Ví dụ, tốt nhất nên thụt vào đầu dòng hay dùng dấu câu để nối mỗi câu else với câu if thích hợp để biểu diễn phạm vi của vòng lặp hay các điều kiện case. Phạm vi của biến và tham số cũng có thể gây ra mơ hồ và nên được ghi chú cẩn thận những chỗ không rõ ràng.

Một vấn đề phổ biến của mã giả liên quan đến mức độ chi tiết. Hầu hết các kỹ sư sẽ viết 1 dòng mã giả cho khoảng từ 3 đến 5 câu của chương trình nguồn hoàn chỉnh. Với tính linh động, một bản thiết kế có thể ở bất cứ mức độ chi tiết nào nên không có cách nào để

định rõ những gì mà một bản thiết kế mã giả phải có. Trên thực tế, thuận lợi chính của mã giả là cho phép định rõ mức độ chi tiết của thiết kế cần thiết cho mỗi tình huống.

Khi sử dụng mã giả, dữ liệu PSP có thể giúp bạn quyết định về mức độ chi tiết thích hợp. Ví dụ, khi xem lại dữ liệu sai sót, nếu bạn thấy bạn mắc các lỗi thiết kế trong pha cài đặt, hãy xem xét đến việc sử dụng một thiết kế chính xác hơn. Ngược lại, nếu không có các vấn đề về biểu diễn thiết kế, bạn có thể thử một mã giả ít chi tiết hơn để xem nó có tăng tốc công việc thiết kế của bạn mà không gây ra lỗi thiết kế trong thực thi không.

3.7.7.3 Các phương pháp trình bày khác

Các phương pháp toán học khác nhau cũng được khuyên dùng để định nghĩa chính xác các hệ thống phần mềm. Ưu điểm của chúng là chính xác tuy nhiên lại có khuyết điểm là khó học, nhất là với những ai không được đào tạo toán học thích hợp.

Tuy nhiên, các phương pháp hình thức đưa ra các hứa hẹn về việc tạo ra chương trình với một số lượng sai sót ít nhất.

Nếu bạn quan tâm đến việc sử dụng các trình diễn khác nhau trong thiết kế, hãy xem xét đến các điểm sau:

- Thiết kế là một quy trình suy nghĩ.
- Một chú thích thiết kế giàu ngữ nghĩa có thể giúp bạn nghĩ chính xác và biểu diễn được một thiết kế phức tạp.
- Các chú thích giàu ngữ nghĩa lại khó học.
- Khi sử dụng một chú thích thiết kế không quen thuộc, có thể bạn sẽ không suy nghĩ được trong chú thích đó.
- Khi đó bạn phải nghĩ qua thiết kế trong một chú thích quen thuộc hơn và sau đó dịch sang chú thích không biết rõ kia.
- Quá trình dịch này hạn chế sự sáng tạo, trì hoãn công việc thiết kế và gây lỗi.

Khi bạn thử các phương pháp thiết kế và chú thích mới, cố gắng trở nên sử dụng chúng đủ trôi chảy trước khi đánh giá chúng. Nhớ rằng chú thích thiết kế là một phương tiện truyền đạt sự giao tiếp. Nếu người thực thi không thấy thoải mái với nó, họ sẽ có nhiều vấn đề được mô tả như trên.

3.8 Chất lượng sản phẩm

3.8.1 Nhìn nhận về bộ lọc kiểm thử

Hãy thử nghĩ việc loại trừ sai sót như là một bộ lọc. Mỗi quá trình xem lại, biên dịch, kiểm thử sẽ loại bỏ một số phần trăm sai sót trong sản phẩm. Mọi quy trình loại bỏ sai sót đều bỏ sót một phần nhỏ các sai sót và số lượng sai sót đi qua bộ lọc tỉ lệ thuận với số lượng đi vào bộ lọc. Do đó, càng nhiều sai sót đi vào pha kiểm thử, biên dịch, hay xem lại thì sẽ bỏ sót lại nhiều trong sản phẩm khi hoàn tất pha.

Vấn đề kế tiếp liên quan đến chất lượng của bộ lọc. Trong bảng dưới, việc xem lại và thanh tra code có hiệu suất cao nhất, trong khi biên dịch, kiểm thử đơn vị và các dạng kiểm thử khác ít hiệu quả hơn. Cả 2 phương pháp có hiệu suất cao nhất này đều làm thủ công và không bao gồm một công cụ tự động nào vì đầu óc con người là một công cụ dò tìm lỗi cực kỳ mạnh, hơn bất kỳ công cụ phần mềm mới nhất nào.

Phương pháp	**Hiệu suất xấp xỉ (%)**
Xem lại code	70-80
Thanh tra code	50-70
Biên dịch	50
Kiểm thử đơn vị	40-50
Kiểm thử phối hợp	45
Kiểm thử yêu cầu	45
Kiểm thử thuật toán	8

Bảng 3.8.1 Hiệu suất loại trừ lỗi

Kết luận từ dữ liệu trên: để tạo ra sản phẩm chất lượng cao, phải có ít lỗi nhất khi bắt đầu kiểm thử. Tất nhiên, để tạo ra các sản phẩm chất lượng cao nhất, bạn nên đánh giá, phân tích và cải tiến mọi pha loại trừ lỗi.

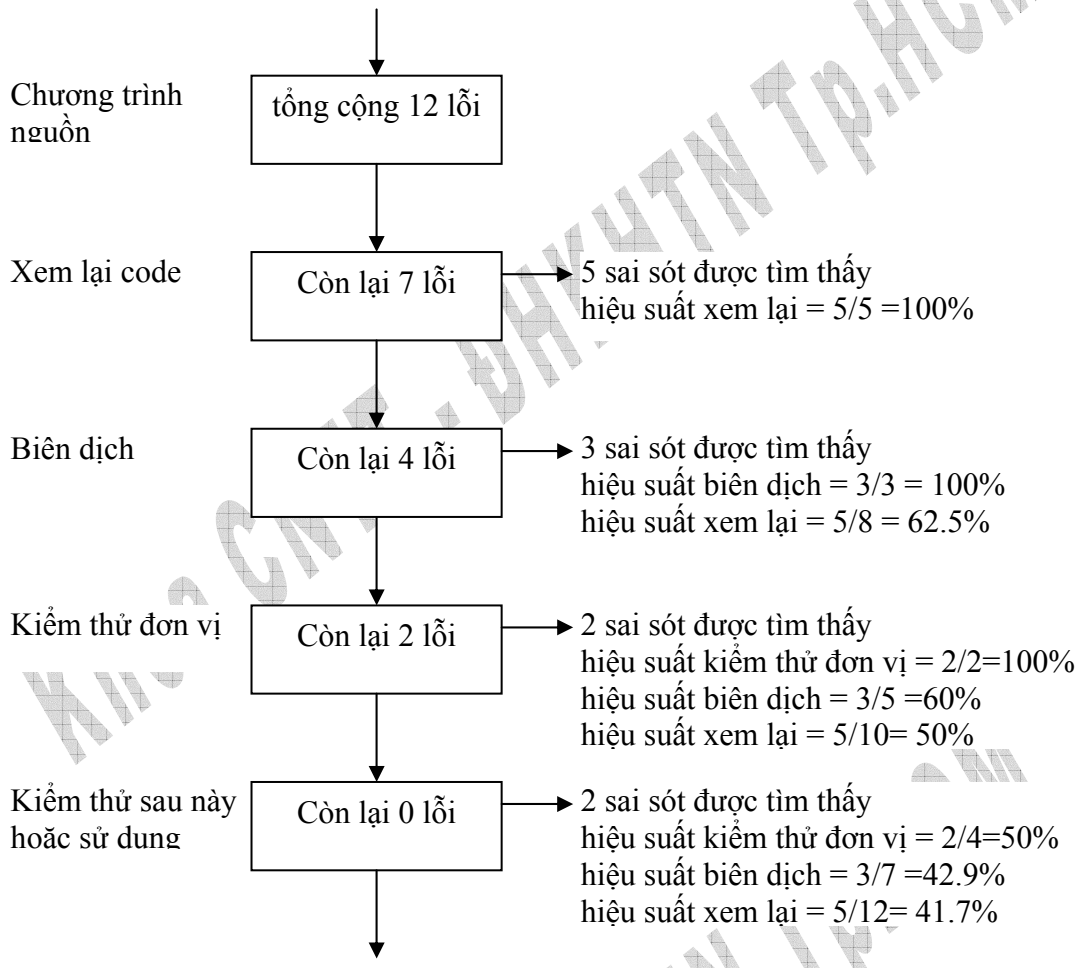
3.8.2 Tính toán các giá trị hiệu suất

Đơn vị đo hiệu suất quy trình được giới thiệu trong các phần trước liên quan tới phần trăm sai sót được loại bỏ trước khi biên dịch. Tuy nhiên đơn vị đo hiệu suất này có thể được áp dụng vào bất cứ bước loại trừ sai sót nào. Vì vậy hiệu suất pha có thể được tính như sau:

Hiệu suất pha = 100 * (sai sót loại bỏ trong pha)/(sai sót trong sản phẩm khi bắt đầu pha)

Một hiệu suất đúng đắn đòi hỏi dữ liệu sai sót từ tất cả pha quy trình trước đó. Ví dụ, bạn tìm thấy 5 lỗi trong xem lại code, 3 trong biên dịch, 2 trong kiểm thử, như vậy xem lại code đã tìm ra 5 trong số 10 sai sót nên hiệu suất là 50%. Mặc dù có thể còn sai sót trong chương trình, nhưng đây là hiệu suất xem lại theo quan điểm này.

Như trong hình 2.17.4, khi bạn tìm lỗi một cách tuần tự, hiệu suất của các pha bỏ sót các lỗi này giảm dần.



Bảng 3.8.2 Các giá trị hiệu suất

Bạn không thể chắc chắn hiệu suất cuối cùng. Một khi sản phẩm được chuyển giao cho người sử dụng cuối, các sai sót vẫn có thể tiếp tục được phát hiện, làm giảm hiệu suất của tất cả các pha loại trừ lỗi.

Hiệu suất quy trình dựa vào phần trăm sai sót được loại bỏ trước khi biên dịch. Vì vậy hiệu suất quy trình được tính như sau:

Hiệu suất quy trình=100*(sai sót loại bỏ trước khi biên dịch)/(sai sót mắc phải trước biên dịch)

3.8.3 Ước lượng hiệu suất cuối cùng

Mặc dù bạn không bao giờ có thể chắc chắn hiệu suất khi bạn hoàn tất một pha, phép đo hiệu suất có thể giúp bạn đánh giá và cải tiến quy trình.

Cách duy nhất để định rõ bao nhiêu sai sót còn lại là theo dõi sai sót được tìm thấy trong sản phẩm trong suốt quá trình hữu ích còn lại của nó. Tuy nhiên, nếu con số sai sót được tìm thấy giảm mạnh qua mỗi pha loại trừ sai sót, con số hiệu suất có thể khá chính xác rồi. Sau khi bạn thu thập được dữ liệu sai sót, bạn có thể phát triển phép đo hiệu suất cho mỗi pha, sau đó tính toán khả năng số sai sót còn lại trong sản phẩm.

Một kinh nghiệm hữu ích là giả sử các lỗi còn lại trong sản phẩm bằng với con số tìm thấy trong pha loại trừ lỗi cuối cùng, điều này đồng nghĩa với việc giả sử hiệu suất của pha cuối này là 50%. Dựa trên dữ liệu của tác giả, con số này hơi thấp trong pha xem lại và thanh tra code được thực hiện tốt, gần đúng với pha biên dịch và hơi cao cho hầu hết các pha kiểm thử.

Hãy xem xét trường hợp 17 lỗi trong xem lại code, 2 trong biên dịch và 1 trong kiểm thử, ước lượng hiện tại của hiệu suất xem lại là 17/(17+2+1)=85%. Theo kinh nghiệm, khi đó sẽ giả sử rằng có một sai sót nữa sẽ được tìm thấy, điều này sẽ đưa ra ước lượng cuối cùng là 17/(17+2+1+1)=80.95%.

Với dữ liệu lịch sử về hiệu suất thực tế cho mỗi pha loại bỏ sai sót, bạn có thể đưa ra các ước lượng chính xác hơn cho hiệu suất cuối cùng bằng cách sử dụng giá trị hiệu suất đã biết của pha cuối cùng để tính toán số sai sót có thể bị bỏ sót. Nếu có đủ dữ liệu hiệu suất bạn thậm chí có thể tính toán giới hạn bằng thống kê cho số sai sót còn lại. Bạn sẽ không bao giờ biết giá trị hiệu suất thực sự nên các dữ liệu này sẽ giúp bạn đưa ra các ước lượng khá tốt.

3.8.4 Lợi ích của hiệu suất quy trình 100%

Mục tiêu của xem lại code nên đạt tới hiệu suất quy trình 100%. Nếu bạn làm được điều này bạn sẽ đạt được các lợi ích sau:

- Bạn sẽ tìm ra được tất cả các sai sót và không còn lỗi nào để phát hiện trong biên dịch hay kiểm thử.
- Tiết kiệm thời gian biên dịch và kiểm thử.
- Lợi ích lớn trong chi phí và kế hoạch đề án.
- Tạo ra sản phẩm tốt hơn.

Tiến đến hiệu suất 100% không phải là dễ dàng. Nó cần thời gian, luyện tập, tập hợp và phân tích một lượng lớn dữ liệu. Tuy nhiên, phát triển phần mềm hiệu suất cao là một kỹ năng có thể học và cải tiến.

3.8.5 Prototyping

Tuy khó để đạt được hiệu suất cao, nhưng có thể thực hiện được bằng cách luyện tập. Bạn nên đạt được hiệu suất hơn 70%.

Một khi bạn đã đạt được hiệu suất hơn 70%, tiếp tục cải tiến sẽ khó khăn hơn rất nhiều. Đôi khi bạn có thể bỏ sót 1 hay 2 lỗi, hầu hết các lỗi bỏ sót đều liên quan đến thiết kế. Hoặc bạn sẽ đối mặt với hệ thống mới hay các vấn đề hỗ trợ hay cần phải sử dụng các hàm chương trình xa lạ. Với các chương trình lớn, giao diện, trình diễn, quản lý bộ nhớ và nhiều vấn đề khác sẽ trở nên quan trọng hơn.

Mỗi khi bạn làm việc gì mới bạn đều có thể mắc sai sót. Vì vậy, để hạn chế sai sót, bạn nên thử viết các chương trình prototype nhỏ trước khi sử dụng nó trong chương trình. Kiểm tra các ý tưởng mới và xây dựng các prototype đơn giản của bất cữ cấu trúc hay xây dựng nào mà bạn chưa từng sử dụng trước đây.

Bạn cũng sẽ phát hiện ra rằng một số sai sót nhất định thì khó tìm ra hoặc ngăn ngừa hơn những cái khác. Khi bạn nghĩ ra cách để phát hiện ra sai sót mà bạn thường mắc phải nhất thì nhiều sai sót lại liên quan đến ứng dụng, hỗ trợ hệ thống và các vấn đề về môi trường phát triển. Bí quyết là nhận ra sự khác biệt giữa cái gì bạn thật sự biết rõ và cái nào mà bạn nghĩ là bạn chỉ hiểu nó. Đôi khi bạn sẽ mắc một lỗi mà bạn thật sự biết, nhưng hầu hết thời gian sai sót của bạn liên quan đến sự giả định thừa nhận một cái không hoàn toàn đúng. Hãy học cách để nhận ra sự giả định này. Sau đó xây dựng 1 prototype để kiểm tra chúng trước khi bạn xây dựng các giả định này trong sản phẩm.

3.9 Chất lượng quy trình

3.9.1 Các phép đo quy trình

Chất lượng sản phẩm được quyết định bởi chất lượng của quy trình, mà chất lượng của quy trình lại được quyết định bởi cách làm việc của bạn. Để phát triển chương trình tốt hơn, bạn cần phải làm việc một cách hiệu quả nhất, do đó cần phải biết quy trình của bạn đã tốt hay chưa bằng cách đo chất lượng quy trình.

Các phép đo quy trình cơ bản nhất liên quan đến quy mô sản phẩm tạo ra, chất lượng của chúng, thời gian và tài nguyên cần thiết để thực hiện. Từ các yếu tố này, bạn có thể biết được hiệu quả của cách làm việc hiện tại của mình và bạn có thể thay đổi như thế nào để tạo ra các sản phẩm tốt hơn trong tương lai.

3.9.2 Nghịch lý của việc loại trừ sai sót

Một hiện tượng bạn cần phải chú ý là tốc độ loại trừ sai sót giảm khi chất lượng sản phẩm tăng. Điều này là tự nhiên khi bạn xem xét đến trường hợp sau: Khi có nhiều sai sót, thường bạn sẽ phát hiện ra nhiều một cách nhanh chóng. Trong khi đó, với chỉ một sai sót trong một chương trình rất lớn, tất cả thời gian trải qua để xem lại và kiểm thử sẽ chỉ tìm ra một sai sót này. Việc này rất tốn thời gian.

Vì vậy, với ít lỗi hơn thì sẽ mất nhiều thời gian hơn để tìm chúng, bất chấp phương pháp mà bạn sử dụng.

Vì vậy khi bạn đưa các chương trình chất lượng cao dần vào một số pha liên tiếp của kiểm thử tích hợp và kiểm thử hệ thống, các sai sót còn lại của chúng sẽ khó khăn để tìm và sửa chữa hơn. Điều này khuyên rằng càng nhiều sai sót tìm thấy thì càng quan trọng để tìm và sửa chữa tất cả chúng.

3.9.3 Một chiến lược loại trừ sai sót

Các sai sót trong hệ thống có thể được chia làm 2 loại: những lỗi chỉ liên quan đến một module và những lỗi liên quan đến sự tương tác giữa các module.

Khi bạn được một hiệu suất cao, bạn sẽ loại trừ hầu hết các lỗi của loại thứ nhất. Loại thứ 2 tuy nhiên lại khó hơn nhiều. Đó là vì các hệ thống lớn và phức tạp bao gồm quá nhiều các tương tác, khó khăn cho người thiết kế mường tượng tất cả chúng. Sau đây là chiến lược để giải quyết:

- Cố gắng phát triển các module có chất lượng cao nhất có thể.
- Thanh tra kỹ lưỡng tất cả các giao diện và tương tác module.
- Thanh tra các yêu cầu để chắc chắn rằng tất cả các khả năng quan trọng được hiểu, thiết kế và thực thi đúng đắn.
- Thanh tra hệ thống và thiết kế của chương trình dựa vào các yêu cầu để bảo đảm nó đã chú ý đến tất cả các yêu cầu chính.
- Kiểm thử đơn vị toàn diện sau khi mã nguồn đã được thanh tra.
- Kiểm thử sự phối hợp toàn diện.
- Kiểm thử hệ thống kỹ lưỡng.

Các bước này ngoại trừ bước đầu tiên vượt quá phạm vi của quy trình cá nhân. Bước đầu tiên phụ thuộc vào bạn. Nếu các module của bạn không có chất lượng tốt nhất,

phần còn lại của các bước sẽ kém hiệu quả. Vì vậy quy trình phát triển đầu tiên phải tập trung vào việc phát hiện và vá lỗi trong các module.

3.9.4 Chi phí của chất lượng

Là một kỹ sư phần mềm, bạn cần phải cân bằng thận trọng giữa thời gian tiêu tốn để tìm sai sót và chất lượng sản phẩm vì khi bạn bỏ ra một lượng thời gian đến thế nào đi nữa cũng không thể chắc chắn đã phát hiện ra hết các sai sót. Chi phí chất lượng (COQ – cost of quality) cung cấp một cách để giải quyết vấn đề này. Chi phí chất lượng có 3 thành phần chính: chi phí sai sót, chi phí đánh giá và chi phí phòng ngừa.

Chi phí sai sót bao gồm tất cả các chi phí cho vá lỗi sản phẩm. Chi phí này bao gồm tất cả mọi việc để sửa lỗi, bao gồm cả việc thiết kế lại, biên dịch lại hay kiểm thử lại.

Chi phí đánh giá bao gồm tất cả các công việc đánh giá sản phẩm để xem sản phẩm có sai sót không, loại trừ thời gian tiêu tốn để vá lỗi. Chi phí này bao gồm việc xem lại code, thời gian biên dịch và kiểm thử một chương trình không lỗi nên không bao gồm chi phí sửa chữa.

Chi phí phòng ngừa là khi bạn chỉnh sửa quy trình để tránh việc mắc lỗi. Ví dụ, nó bao gồm các phân tích thực hiện để hiểu các sai sót, công việc phát triển quy trình để cải tiến các yêu cầu, thiết kế hay các quy trình thực thi. Thời gian tiêu tốn trong việc thiết kế lại và kiểm thử một quy trình mới cũng là chi phí phòng ngừa. Chi phí để viết một prototype nhỏ cũng là chi phí phòng ngừa.

3.9.5 Tính toán chi phí của chất lượng

PSP tính chi phí chất lượng theo một cách đơn giản. Vì thời gian bỏ ra khi biên dịch bao gồm một số thời gian biên dịch không lỗi, PSP tính tất cả các thời gian biên dịch này là chi phí sai sót. Tương tự như vậy với việc kiểm thử. Thời gian xem lại bao gồm một số chi phí sửa chữa nhưng PSP tính tất cả thời gian xem lại là chi phí đánh giá.

Bạn có thể tính toán giá trị COQ với các phương pháp được mô tả trong phần *Tính toán chi phí chất lượng thật sự 3.9.8*, nhưng nó liên quan đến rất nhiều công việc và không thay đổi đáng kể các phép đo. Vì vậy bạn nên sử dụng các định nghĩa PSP đơn giản:

Chi phí chất lượng được tính là phần trăm của tổng thời gian phát triển. Với PSP, chi phí đánh giá và sai sót được tính như sau:

- Chi phí đánh giá là phần trăm tổng thời gian xem lại với tổng thời gian phát triển.

Sinh viên	Sinh viên X	Ngày	9/12/96
Chương trình		Chương trình #	15
Người hướng dẫn	Thầy Z	Ngôn ngữ	Ada

Tóm tắt	**Kế hoạch**	**Thực tế**	**Đến ngày**
Phút/LOC	5.48	4.60	5.35
LOC/Giờ	10.95	13.04	11.21
Sai sót/KLOC	92.53	52.6	86.7
Hiệu suất	40.0	100.0	45.5
A/FR	0.375	1.93	0.44

Kích thước chương trình (LOC)			
Tổng mới và thay đổi	49	57	392
Kích thước tối đa	62		
Kích thước tối thiểu	36		

Thời gian trong pha (phút)	**Kế hoạch**	**Thực tế**	**Đến ngày**	**Đến ngày %**
Lên kế hoạch	17	20	140	6.7
Thiết kế	29	38	233	11.1
Cài đặt	116	119	911	43.4
Xem lại mã	21	29	174	8.3
Biên dịch	15	5	105	5.0
Kiểm thử	41	10	289	13.8
Tổng kết	30	41	247	11.7
Tổng cộng	269	262	2099	100.0
Kích thước tối đa	340			
Kích thước tối thiểu	197			

Sai sót mắc phải	***Kế hoạch***	**Thực tế**	**Đến ngày**	**Đến ngày %**	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế	1		5	14.7	1.29
Cài đặt	4	3	28	82.4	1.84
Xem lại mã					
Biên dịch			1	2.9	
Kiểm thử					
Tổng cộng	5	3	34	100.0	

Sai sót loại bỏ	***Kế hoạch***	**Thực tế**	**Đến ngày**	**Đến ngày %**	***Sai sót/giờ***
Lên kế hoạch					
Thiết kế					
Cài đặt					
Xem lại mã	2	3	15	44.1	5.17
Biên dịch	2		13	38.2	7.43
Kiểm thử	1		6	17.1	1.25
Tổng cộng	5	3	34	100.0	

Bảng 3.9.1 Ví dụ bản tổng kết kế hoạch dự án

- Chi phí sai sót là phần trăm tổng thời gian biên dịch và kiểm thử với tổng thời gian phát triển.

Ví dụ, giả sử bạn có dữ liệu quy trình cho trong bảng tóm tắt kế hoạch ở bảng 3.9.1.

Bạn có thể tính chi phí đánh giá như sau:

- Tổng thời gian phát triển thực tế = 262 phút
- Thời gian xem lại code thực tế = 29 phút
- Chi phí đánh giá = 100*29/262=11.07%

Với chi phí sai sót:

- Thời gian biên dịch thực tế = 5 phút
- Thời gian kiểm thử thực tế = 10 phút
- Chi phí sai sót = 100*(5+10)/262=100*15/262=5.73%

3.9.6 Tỉ lệ chi phi đánh giá/sai sót(A/FR – Appraisal/Failure Ratio)

A/FR = chi phí đánh giá / chi phí sai sót

Trong ở ví dụ phần trên, với chi phí đánh giá là 11.07% và chi phí sai sót 5.73% thì:

A/FR = 11.07 / 5.73 = 1.93

Cách đơn giản hơn để tính A/FR:

A/FR = thời gian xem lại code / tổng thời gian biên dịch và kiểm thử.

= 29/(5+10)=1.93 (cho kết quả tương tự như trên)

Dưới đây là các chỉ dẫn cập nhật cuối cùng cho bản tổng kết kế hoạch PSP:

Mục đích	Mẫu này ghi nhận các thông tin ước lượng và thực tế của đề án
Đầu trang	Nhập các thông tin: - Tên và ngày hiện tại - Tên và mã số chương trình - Tên người hướng dẫn - Ngôn ngữ sử dụng để lập trình
Tóm tắt	
Phút/LOC	Trước khi phát triển: - Nhập giá trị Phút/LOC dự kiến cho đề án này. Sử dụng tốc độ Đến ngày từ chương trình gần nhất trong bản ghi công việc hay bản tổng kết kế hoạch dự án. Sau khi phát triển: - Chia tổng thời gian phát triển cho độ lớn chương trình thực tế để có chỉ số Phút/LOC thực tế và Đến ngày - Ví dụ, nếu dự án phát triển mất 196 phút và gồm 29 LOC, chỉ số Phút/LOC sẽ là 196/29=6.76
LOC/Giờ	Trước khi phát triển: - Tính LOC/Giờ dự kiến bằng cách lấy 60 chia cho Phút/LOC dự kiến Sau khi phát triển: - Để tính LOC/Giờ thực tế và Đến ngày, lấy 60 chia cho Phút/LOC thực

	tế Đến ngày Ví dụ: Với chỉ số Phút/LOC thực tế là 6.76, chỉ số LOC/Giờ thực tế là 60/6.76=8.88
Sai sót/KLOC	Trước khi phát triển: - Tìm số sai sót/KLOC trong các chương trình gần đây nhất. - Sử dụng giá trị này như là số sai sót/KLOC kế hoạch cho dự án này. Sau khi phát triển: - Tính số sai sót/KLOC thực tế và Đến ngày cho chương trình này. - Với giá trị thực tế: Tổng số sai sót thực tế *1000 / Tổng LOC Mới và Thay đổi thực tế - Tính toán tương tự cho giá trị Đến ngày - Ví dụ: với 17 sai sót Đến ngày và 153 LOC Mới và Thay đổi thì chỉ số sai sót/KLOC Đến ngày là = 1000*17/153 = 111.11
Hiệu suất	Tính hiệu suất kế hoạch, thực tế và Đến ngày. Hiệu suất = 100 * (sai sót loại bỏ trước khi biên dịch) / (sai sót mắc phải trước khi bên dịch) Với 5 sai sót mắc phải và 4 sai sót loại bỏ, hiệu suất = 100*4/5=80.8%
A/FR	***Tính giá trị A/FR kế hoạch, thực tế và Đến ngày.*** ***Ví dụ với A/FR thực tế = (Thời gian xem lại code thực tế) / (Tổng thời gian biên dịch và kiểm thử thực tế)*** ***Với thời gian biên dịch 29 phút, thời gian biên dịch 5 phút và kiểm thử là 10 phút, A/FR = 29/(5+10)=1.93***
Độ lớn chương trình (LOC)	Trước khi phát triển: - Nhập giá trị Tổng cộng, Tối đa và tối thiểu của LOC Mới & Thay đổi Sau khi phát triển: - Đếm và nhập giá trị LOC Mới & Thay đổi thực tế. - Với Đến ngày, cộng thêm LOC Mới và Thay đổi thực sự với LOC mới và Thay đổi Đến ngày của chương trình trước đó.
Thời gian bỏ ra ở từng giai đoạn	
Kế hoạch	Đối với Tổng thời gian phát triển (Total Development time), nhân LOC Mới & Thay đổi với Phút/LOC Đối với Thời gian tối đa, nhân độ lớn tối đa (Maximum size) với Phút/LOC. Đối với Thời gian tối thiểu, nhân độ lớn tối thiểu (Minimum size) với Phút/LOC. Từ bản tổng kết kế hoạch dự án của chương trình gần nhất, tìm giá trị Đến ngày % cho mỗi pha. Sử dụng Đến ngày % từ chương trình trước đó, tính toán thời gian kế hoạch cho mỗi pha.
Thực tế	Sau khi hoàn tất, nhập thời gian thực tế tính theo phút trong mỗi pha phát triển. Lấy dữ liệu này từ Bản ghi nhận thời gian
Đến ngày	Với mỗi pha, điền vào tổng thời gian thực tế và thời gian Đến ngày từ chương trình gần nhất.
Đến ngày %	Với mỗi pha, điền vào (thời gian Đến ngày * 100) / Tổng thời gian Đến

	ngày.
Sai sót mắc phải	
Kế hoạch	Trước khi phát triển, ước lượng tổng số sai sót sẽ có thể mắc phải trong chương trình: sai sót/KLOC kế hoạch * LOC Mới và Thay đổi kế hoạch của chương trình / 1000 Ví dụ, với sai sót/KLOC kế hoạch là 75.9 và LOC Mới và Thay đổi là 75, tổng số sai sót kế hoạch = 75.9*75/1000 = 5.96, làm tròn thành 6. Trước khi phát triển, ước lượng sai sót mắc phải trong từng pha bằng cách sử dụng tổng sai sót ước lượng và sự phân bố sai sót mắc phải Đến ngày % của chương trình trước.
Thực tế	Sau khi phát triển, tìm và điền số lượng sai sót thực tế mắc phải trong mỗi pha
Đến ngày	Với mỗi pha, nhập vào tổng số sai sót thực tế và sai sót Đến ngày từ chương trình gần nhất.
Đến ngày %	Với mỗi pha, nhập vào 100*(Sai sót Đến ngày của pha đó)/(Tổng sai sót Đến ngày)
Sai sót loại bỏ	
Kế hoạch	Ở dòng tổng cộng, điền vào tổng số sai sót ước lượng. Sử dụng các giá trị Đến ngày từ chương trình gần nhất, tính toán sai sót kế hoạch loại bỏ được trong mỗi pha.
Thực tế	Sau khi phát triển, tìm và điền số lượng sai sót thực tế loại bỏ trong mỗi pha
Đến ngày	Với mỗi pha, nhập vào tổng số sai sót thực tế và sai sót Đến ngày từ chương trình gần nhất.
Đến ngày %	Với mỗi pha, nhập vào 100*(Sai sót Đến ngày của pha đó)/(Tổng sai sót Đến ngày)

Bảng 3.9.2 Chỉ dẫn cho bản tổng kết kế hoạch

A/FR đo lượng thời gian tiêu tốn tương đối trong việc tìm lỗi trước khi biên dịch lần đầu. Khi A/FR<1, việc kiểm thử chương trình thường tìm ra nhiều lỗi. Mặc dù điều này là không bảo đảm, hầu hết các chương trình nằm trong giới hạn này có một số lượng khá cao sai sót kiểm thử/KLOC. Ngoài ra, các quy trình với A/FR>2 có ít lỗi hơn là quy trình có A/FR<1. Do đó bạn nên cố gắng đạt được quy trình với A/FR>=2 để tạo ra sản phẩm không lỗi.

Sử dụng giá trị A/FR

Để đạt được A/FR trên 2.0, hãy xem lại thời gian kiểm thử và biên dịch lịch sử, lên kế hoạch để dùng ít nhất 2 lần lượng thời gian đó trong việc xem lại code lần tới. Bạn có thể làm tăng A/FR bằng cách chỉ việc dùng nhiều thời gian hơn trong việc xem lại code. Tuy nhiên, chỉ trừ khi bạn tìm thấy sai sót, còn nếu không nó sẽ không cải tiến chất lượng chương trình.

Chừng nào mà hiệu suất của bạn chưa đạt được giới hạn 80% đến 100%, hãy tiếp tục tăng A/FR lên. Tuy nhiên bạn không được làm điều này bằng cách chỉ đơn thuần thực hiện nhiều thời gian hơn mà còn phải xem lại dữ liệu sai sót cho các chương trình phát triển gần đây và nghĩ ra cách để tìm thấy tất cả các sai sót mà bạn thường hay bỏ sót. Sau đó, thêm vào các bước thích hợp trong danh sách xem lại code. Cuối cùng, theo các bước xem lại khi bạn thực hiện xem lại code. Nếu bạn làm như vậy, bạn sẽ mất thời gian hơn cho thời gian xem lại, sẽ tìm ra nhiều lỗi hơn và sẽ tăng A/FR, đồng thời giảm số sai sót bạn tìm thấy trong biên dịch và kiểm thử. Điều này sẽ tiết kiệm rất nhiều thời gian kiểm thử, giảm chi phí sai sót và tạo ra sản phẩm có chất lượng cao hơn.

Chú ý rằng khi chỉnh sửa các chương trình lớn hơn, thường cần kiểm thử nhiều hơn cho dù không có sai sót nào. Trong những trường hợp này, giá trị A/FR thường sẽ thấp hơn.

3.9.7 Cải tiến tốc độ xem lại

Đừng quan tâm đến tốc độ xem lại sai sót/giờ cho đến khi bạn tìm thấy hầu hết các lỗi trước khi biên dịch và kiểm thử một cách ổn định.

Tuy nhiên, một khi bạn đã tìm ra hầu hết lỗi trong xem lại code, hãy nghĩ đến việc cải tiến tốc độ xem lại bằng cách: nhận diện bất cứ bước xem lại nào mà không tìm thấy lỗi cũng như là bỏ sót lỗi. Kế đó, kiểm tra lại các lý do tại sao ban đầu bạn đưa các bước này vào. Nếu các vấn đề này không còn là vấn đề nữa thì bỏ qua các bước này. Mặt khác, nếu bạn nghĩ chúng vẫn còn quan trọng, kết hợp một vài trong số chúng để bạn có thể thực hiện chúng nhanh chóng. Với mỗi chương trình, tiếp tục giám sát dữ liệu sai sót và phục hồi lại các bước xem lại đã bắt được lỗi mà sau đó bạn đã bỏ sót. Tuy nhiên, khi các bước không hiệu quả, đừng do dự bỏ chúng.

3.9.8 Tính toán chi phí chất lượng thật sự

Với PSP, các tính toán chi phí chất lượng đơn giản là thích hợp. Tuy nhiên, khi bạn làm việc với các dự án phát triển lớn, bạn có thể muốn sử dụng phép đo chi phí chất lượng chính xác hơn.

Để thực hiện, bạn phải chia các thời gian xem lại, biên dịch và kiểm thử thành các thành phần đánh giá và sai sót tương ứng. Ví dụ, chúng ta có thể ghi nhãn cho thời gian biên dịch khi không có sai sót được tìm thấy là biên dịch đánh giá hay C_A và thời gian vá lỗi suốt quá trình biên dịch là biên dịch sai sót hay C_F. Vì vậy, $C_F + C_A = C$ (tổng thời gian

biên dịch). Với thời gian xem lại và kiểm thử, $R_F + R_A = R$ (tổng thời gian xem lại), và $T_F + T_A = T$ (tổng thời gian kiểm thử).

Tính toán như sau:

Chi phí chất lượng đánh giá = $100*(R_A + C_A + T_A)$/(tổng thời gian phát triển)

Chi phí chất lượng sai sót = $100*(R_F + C_F + T_F)$/(tổng thời gian phát triển)

Loại sai sót	
10 Sưu liệu	60 Kiểm tra
20 Cú pháp	70 Dữ liệu
30 Xây dựng, đóng gói	80 Chức năng
40 Chỉ định	90 Hệ thống
50 Giao diện	100 Môi trường

Sinh viên: Sinh viên X — Ngày: 9/12/96

Người hướng dẫn: Thầy Z — Chương trình #: 15

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
12/9	1	40	cài đặt	xem lại	2	

Mô tả: Thiếu khai báo Set_X

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
	2	80	thiết kế	xem lại	8	

Mô tả: quên rằng chỉ tiến lên 1 bước trong vòng lặp while nếu lẻ

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa
	3	20	cài đặt	xem lại	1	

Mô tả: thiếu ";"

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa

Mô tả

Ngày	Số	Loại	Mắc phải	Loại bỏ	T/g sửa chữa	Sai sót sửa chữa

Mô tả

Bảng 3.9.3 Ví dụ bản ghi ghi chép sai sót

Ví dụ sau sử dụng dữ liệu trong bảng tóm tắt kế hoạch dự án trong bảng 3.9.1 và bản ghi ghi chép sai sót trong bảng 3.9.3. Đầu tiên tính các giá trị R_A, C_A, T_A, R_F, C_F, T_F như sau:

- R_F: tính từ bản ghi ghi chép sai sót, là tổng của thời gian vá các lỗi trong xem lại code: $R_F = 2+8+1=11$
- $R_A = R-R_F = 29-11=18$
- Vì không có lỗi được tìm thấy trong biên dịch nên tất cả thời gian là thời gian đánh giá: $C_A = 5$ và $C_F = 0$
- Vì không có lỗi được tìm thấy trong kiểm thử, tất cả thời gian kiểm thử là thời gian đánh giá, vì vậy $T_A = 10$ và $T_F = 0$

Với các giá trị này, chúng ta tính các giá trị đánh giá và sai sót như sau:

Chi phí chất lượng đánh giá = $100*(R_A + C_A + T_A)$/(tổng thời gian phát triển)

$= 100*(18+5+10)/262 = 100*33/262 = 12.60\%$

Chi phí chất lượng sai sót = $100*(R_F + C_F + T_F)$/(tổng thời gian phát triển)

$= 100*(11+0+0)/262 = 100*11/262 = 4.20\%$

Các giá trị này hơi khác so với các giá trị được tính trước đây. Chúng cũng đưa ra một giá trị A/FR cao hơn đáng kể là 3.0 thay vì 1.93. Vì các giá trị A/FR và COQ chính xác hơn này khá nhạy với thời gian sửa lỗi nên bạn không nên sử dụng phương pháp này trừ khi bạn đo thời gian sửa lỗi bằng đồng hồ bấm giờ. Bạn cũng sẽ muốn đưa ra một mục tiêu A/FR khác vì A/FR có giá trị là 2.0 bây giờ có thể dẫn đến quá nhiều sai sót kiểm thử.

Chương 4. Một số kết quả áp dụng PSP vào trong thực tế

4.1 Trong môi trường công nghiệp [5]

Mặc dù PSP đã được giới thiệu khá nhiều về mặt lý thuyết, nhưng về mặt thực tế áp dụng thì vẫn còn khá ít công ty, tổ chức sử dụng nó. Nguyên nhân là do việc đem áp dụng rộng rãi PSP vào trong các kỹ sư đòi hỏi phải có thời gian và sự nỗ lực. Bên cạnh đó, PSP cũng đòi hỏi phải có những dữ liệu lịch sử. Đó chính là điểm hạn chế mà cho đến hiện nay, những số liệu sử dụng hiệu quả PSP còn khá ít.

Tuy nhiên, có 3 nhóm phát triển phần mềm đã sử dụng PSP và đã ghi nhận lại các số liệu chứng minh tính hiệu quả của việc sử dụng nó. Ba nhóm này là: Advanced Information Services, Motorola Paging Products Groups và Union Switch & Signal Inc. Mỗi nhóm này đã huấn luyện một đội ngũ các kỹ sư sử dụng PSP và đo kết quả của một vài dự án có sử dụng PSP.

Ngôn ngữ sử dụng trong các dự án của 3 công ty trên đều là C và C++.

4.1.1 Advanced Information Services (AIS)

4.1.1.1 Giới thiệu

AIS có trụ sở chính đặt tại Illinois và một bộ phận hỗ trợ đặt tại Madras, Ấn Độ, là một công ty phần mềm chuyên phát triển những sản phẩm phần mềm, các dịch vụ mạng và huấn luyện qui trình.

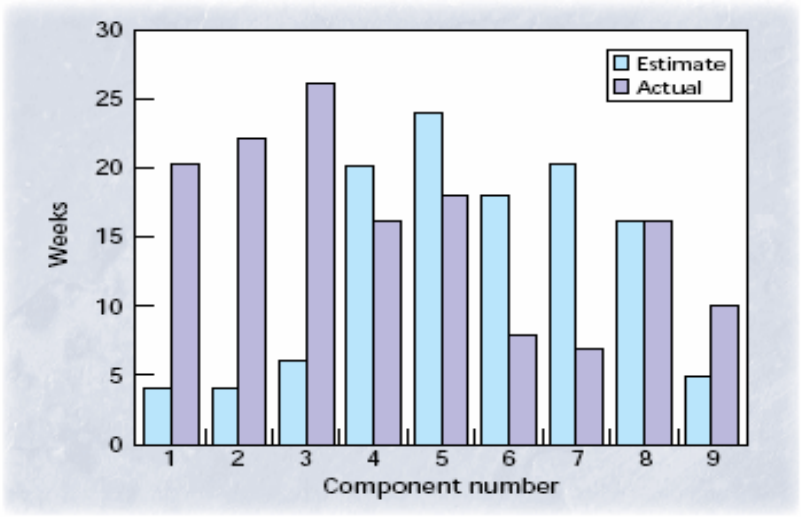
Để thử nghiệm tính hiệu quả của PSP, AIS đã gửi những kỹ sư của công ty đến viện SEI để học tập và sử dụng qui trình này. Sau khi kết thúc khoá học, các kỹ sư đã áp dụng những điều đã học vào trong quá trình thực hiện công việc của mình. AIS đã ghi nhận số liệu của 7 dự án có sử dụng qui trình PSP.

4.1.1.2 Kết quả thu được

4.1.1.2.1 Dự án A

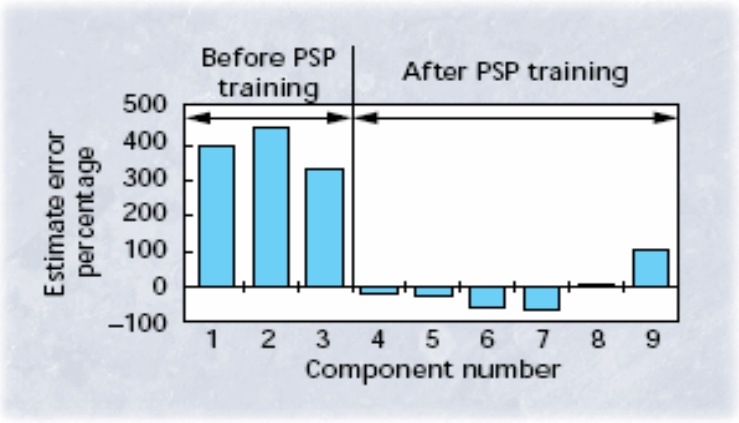
Dự án đầu tiên có sử dụng PSP vào năm 1995 có sự tham gia các của các kỹ sư ở Illinois và Madras. Các thành phần (component) trong dự án này có kích thước từ 500 đến 2200 LOC. Trước tháng 4 năm 1995, các kỹ sư đã hoàn tất các thành phần 1, 2 và 3 nhưng dự án vẫn không thực hiện đúng như ngày đã lập kế hoạch. Do đó, dự án A đã phải lên kế hoạch lại và công ty phải thương lượng với khách hàng về thời gian bàn giao sản phẩm.

Đến thời điểm này, người trưởng dự án đã quyết định sử dụng PSP. Các kỹ sư trong nhóm phát triển lúc này sẽ tự lên kế hoạch và phát triển các thành phần từ 4 đến 9.



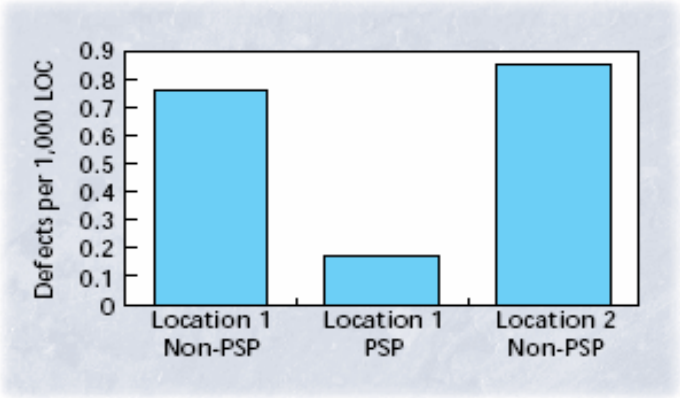
Hình 4.1.1 Ước lượng kế hoạch cho dự án A của AIS

Trong hình trên, ở các thành phần từ 1 đến 3, mức độ chênh lệch giữa số tuần ước lượng và số tuần thực tế đã thực hiện chênh lệch khá nhiều. Tuy nhiên, ở những thành phần sau, sự chênh lệch giảm dần và đặc biệt ở thành phần 8, sự ước lượng và thực tế là bằng nhau.



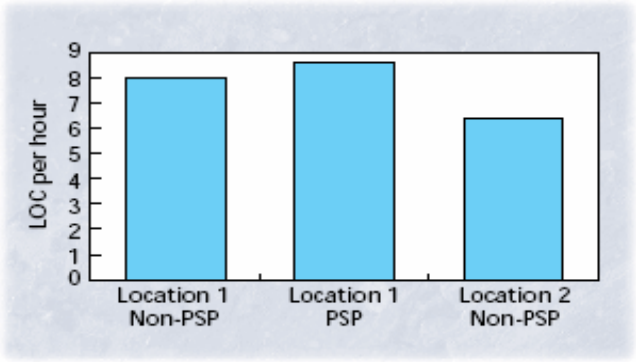
Hình 4.1.2 Tỉ lệ chênh lệch kế hoạch trong dự án A của AIS

Hình trên cho thấy, trước khi sử dụng PSP, tỉ lệ sai sót trong ước lượng là 394% (thành phần 1) nhưng ở thành phần 4 tỉ lệ này còn – 10.4%.



Hình 4.1.3 Chất lượng của dự án A

Trong hình trên, ở Illinois (location 1), trong những thành phần đầu (từ 1 đến 3), do các kỹ sư không sử dụng PSP, tỉ lệ sai sót là 0.76 sai sót/1000LOC (Location 1 Non – PSP). Nhưng sau khi áp dụng PSP, từ các thành phần 4 đến 9, tỉ lệ sai sót là 0.17 sai sót/1000LOC => chất lượng đã tăng 78%. Ở Madras (location 2), các kỹ sư không được huấn luyện PSP, tỉ lệ sai sót là 0.85 sai sót/1000 LOC.



Hình 4.1.4 Hiệu quả làm việc của các kỹ sư

Hình trên mô tả, ở Illinois (Location 1), trước khi sử dụng PSP, hiệu suất làm việc của các kỹ sư là 7.99 LOC/giờ. Sau khi huấn luyện PSP (Location 1 - PSP), hiệu suất tăng lên 8.58 LOC/giờ, nghĩa là tăng 7.4%. Với các kỹ sư ở Madras (location 2) hiệu suất chỉ là 6.4 LOC/giờ.

4.1.1.2.2 Dự án B, C, D, E, F, G

Dự án B sử dụng 3 kỹ sư và cả 3 kỹ sư này đều được huấn luyện PSP. Tổng số sai sót trong quá trình kiểm tra là 1 và không có những sai sót do khách hàng báo lại. Dự án kết thúc sớm hơn so với kế hoạch, do đó đáp ứng đúng yêu cầu thời hạn giao sản phẩm cho khách hàng.

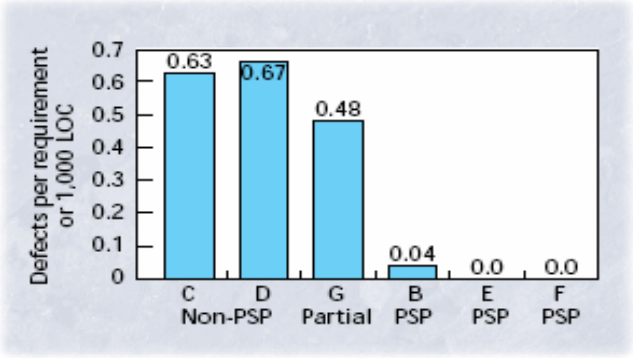
Dự án	Kỹ sư được huấn luyện PSP	Kỹ sư không được huấn luyện PSP	Kích thước sản phẩm	Kế hoạch/Thực tế (Tháng)	Số sai sót trong pha kiểm tra	Số sai sót trong quá trình khách hàng sử dụng
B	3	0	24 yêu cầu	7/5	1	0
C	0	3	19 yêu cầu	2/5	11	1
D	0	3	30 yêu cầu	10/19	6	14
E	1	0	2255 LOC	6/6	0	0
F	1	0	1400 LOC	2/2	0	0
G	2	1	6196 LOC	2/2	0	3

Bảng 4.1.1 bản tổng kết của các dự án B, C, D, E, F, G

Dự án C sử dụng tổng cộng 3 kỹ sư và 3 kỹ sư này chưa được huấn luyện PSP. Tuy nhiên, dự án B và C được đánh giá là tương tự nhau về các mặt ngôn ngữ lập trình, độ phức tạp của yêu cầu… Ba kỹ sư sử dụng trong dự án này là những người có kinh nghiệm hàng đầu trong công ty. Kết quả là dự án C trễ thời hạn giao cho khách hàng. Sai sót trong pha kiểm tra là 11 và sau khi bàn giao sản phẩm vẫn còn 1 sai sót do khách hàng báo lại.

Các dự án E, F sử dụng đội ngũ kỹ sư đều được huấn luyện PSP thì kết quả rất tốt. Giao sản phẩm đúng lịch với khách hàng, sai sót trong pha kiểm tra và những sai sót do khách hàng báo lại không có. Trong khi đó, dự án G, sử dụng 2 kỹ sư được huấn luyện PSP và một kỹ sư không được huấn luyện PSP, kết quả là phần mềm vẫn kịp thời gian giao cho khách hàng nhưng sau khi bàn giao vẫn còn 3 sai sót do khách hàng báo lại.

Hình dưới cho thấy, nhóm 1 gồm những dự án mà các kỹ sư không được huấn luyện PSP, tỉ lệ sai sót/KLOC cao. Nhóm 2 gồm những dự án có một số phần kỹ sư được huấn luyện PSP, số còn lại không được huấn luyện thì tỉ lệ sai sót thấp hơn so với nhóm 1 nhưng vẫn còn ở mức cao. Trong khi đó nhóm 2 với những dự án sử dụng 100% kỹ sư được huấn luyện PSP thì tỉ lệ này rất thấp và có khi đạt được giá trị là 0.



Hình 4.1.5 Chất lượng của các dự án B, C, D, E, F, G

Ngoài những lợi ích đã bàn ở trên, khi sử dụng PSP trong quá trình phát triển phần mềm, các tổ chức phần mềm cũng đã giảm thiểu khá lớn thời gian kiểm tra hệ thống. Dữ liệu cụ thể như sau:

	Kích thước	Thời gian kiểm tra hệ thống
Dự án không sử dụng kỹ sư được huấn luyện PSP		
A1	15800 LOC	1.5 tháng
C	19 yêu cầu	3 vòng kiểm tra (test cycles)
D	30 yêu cầu	2 tháng
H	30 yêu cầu	2 tháng
Dự án có sử dụng kỹ sư được huấn luyện PSP		
A2	11700 LOC	1.5 tháng
B	24 yêu cầu	5 ngày
E	2300 LOC	2 ngày
F	1400 LOC	4 ngày
G	6200 LOC	4 ngày
I	13300 LOC	2 ngày

Bảng 4.1.2 Một số dữ liệu về thời gian kiểm tra hệ thống

Trước khi sử dụng PSP, thời gian dành cho việc kiểm tra hệ thống tốn khá nhiều. Tuy nhiên với những dự án có sử dụng đội ngũ kỹ sư được huấn luyện PSP thì thời gian kiểm tra hệ thống giảm xuống rất nhiều. Trong dự án A2, thời gian kiểm tra hệ thống nhiều vì dự án này phải được kiểm tra kết hợp với dự án A1.

4.1.2 Motorola Paging Products Group

4.1.2.1 Giới thiệu

Motorola Paging Products Group là một công ty chuyên sản xuất các thiết bị di động hay máy nhắn tin… Trụ sở của công ty đặt tại bãi biển Boynton, Florida.

Ba người quản lý của Motorola và 2 giáo sư từ đại học Embry – Riddle Aeronautical đã giới thiệu PSP vào trong công ty. Cho đến nay thì Motorola đã tổ chức 3 lớp huấn luyện PSP, huấn luyện được 40 kỹ sư và 22 người quản lý. Motorola đánh giá khá cao quy trình này và đã sử dụng khoảng 8 giờ/tuần cho việc huấn luyện và 4 giờ khác cho việc tập luyện cải tiến quy trình. Bên cạnh đó, để khuyến khích các kỹ sư sử dụng PSP, ban giám đốc còn xây dựng logo cho các đội thể thao của công ty mang biểu tượng PSP. Những cá nhân hoàn thành tốt khoá học PSP đều có thưởng về tài chính và được tham gia câu lạc bộ PSP hoạt động vào hàng tháng.

4.1.2.2 Kết quả thu được

Số dự án	Kích thước (LOC)	Số tháng sử dụng	Tổng số sai sót	Sai sót trong quá trình kiểm tra	Sai sót trong quá trình sử dụng
1	463	18	13	5	0
2	5465	NA	69	10	0
3	1571	NA	47	8	0
4	3381	NA	69	22	0
5	5	9	0	0	0
6	22	5	2	0	0
7	1	18	1	0	0
8	2081	10	34	0	1
9	114	8	15	2	0
10	364	NA	29	2	0
11	7	5	0	0	0
12	620	3	12	2	0
13	720	NA	9	2	0
14	3894	NA	20	2	0
15	2075	NA	79	27	0
16	1270	NA	20	1	0
17	467	NA	17	3	0
18	3494	8	139	50	0
Tổng cộng	25114	NA	575	136	1

Bảng 4.1.3 Dữ liệu của 18 dự án trong quá trình thử nghiệm hiệu quả của PSP

Trong dự án 1, thời gian thực tế nhiều hơn thời gian lập kế hoạch, nhưng do thời gian sử dụng trong pha kiểm tra giảm 60% so với những dự án trước (không sử dụng PSP) nên tổng thời gian sử dụng cho toàn dự án ít hơn 44% so với thời gian kế hoạch.

Trong dự án 2, tổng cộng sai sót trong dự án là 69, tuy nhiên phần lớn các sai sót này đều đã được tìm thấy và loại bỏ từ các pha trước. Đến pha kiểm tra thì chỉ còn lại 10 sai sót do đó dự án đã giảm được hơn 80% sai sót trước khi đến pha kiểm tra.

Những giá trị NA trong cột số tháng sử dụng là những số mà Motorola không xác định được.

4.1.3 Union Switch & Signal Inc

4.1.3.1 Giới thiệu

Công ty Union Switch & Signal (US & S) chuyên sản xuất nhiều sản phẩm phần cứng phục vụ cho các ngành công nghiệp vận chuyển. Công ty chuyên phát triển và lắp đặt các hệ thống điều khiển thời gian thực. Trụ sở của công ty đặt tại Pittsburgh và có khoảng 100 nhân viên phần mềm.

US & S đã tổ chức 3 lớp học PSP cho 9 người quản lý và 25 kỹ sư phần mềm.

4.1.3.2 Kết quả thu được

Sau khi huấn luyện PSP, các kỹ sư đã thực hiện 5 dự án và con số thực tế thu thập được sau khi thực hiện 5 dự án này như sau:

Sản phẩm	LOC	Số tháng sử dụng	Số sai sót trong tìm thấy trong pha kiểm tra	Số sai sót tìm thấy trong khi sử dụng
M45	193	9.0	4	0
M10	453	7.5	2	0
M77	6133	4.0	25	0
M54	477	3.5	5	0
M53	1160	1.0	21	0
Tổng cộng	8416	NA	57	0

Bảng 4.1.4 Dữ liệu thực tế của các dự án sau khi kỹ sư được huấn luyện PSP

4.1.4 Một số nhóm phát triển phần mềm khác

Ngoài cuộc khảo sát kết quả sử dụng của 3 công ty trên do sự phối hợp của 3 công ty với viện SEI, nhiều tổ chức đã áp dụng PSP và cũng thu được những kết quả khả quan.

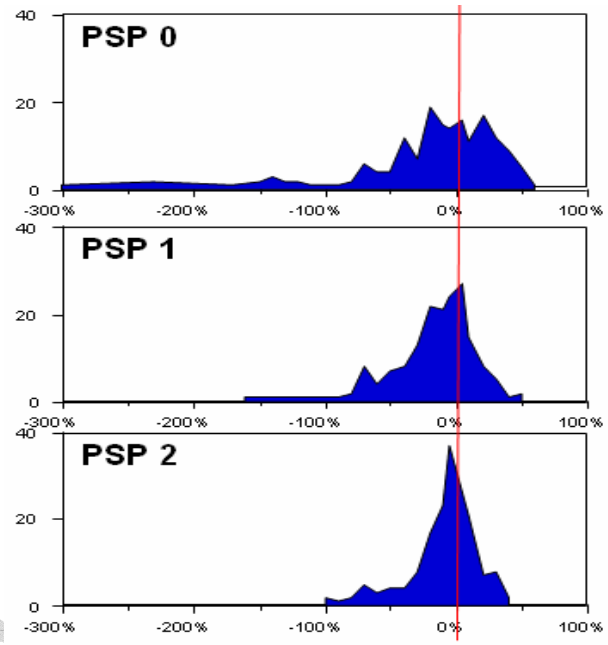
Công ty FIDA: một công ty được chứng nhận ISO 9001 chuyên sản xuất các điều khiển số học. FIDA bắt đầu sử dụng PSP vào tháng 4/1997. Trước khi sử dụng PSP, việc ước lượng trong dự án chủ yếu dựa vào sự tương tự. Nghĩa là, công ty xem xét dự án mới có gần giống với những dự án nào trước đó và tính ra được thời gian và chi phí. Tuy nhiên, sau khi sử dụng PSP, công ty đã sử dụng mối quan hệ giữa kích thước và thời gian để ước lượng. Thêm vào đó, hiệu quả công việc, tỉ lệ sai sót cũng giảm thiểu đáng kể.[6]

Teradyne sử dụng qui trình PSP, xây dựng một hệ thống gồm tổng cộng 90000 LOC. Sau khi hoàn thành, thời gian lúc đầu dự kiến là 77 tuần nhưng khi thực hiện chỉ 71 tuần (sớm hơn 8%), chất lượng phần mềm tăng 100 lần so với những dự án trước đó, mật độ sai sót là 0.02 sai sót/KLOC. [7]

Căn cứ quân sự hàng không Hill đã phát triển một dự án gồm 25820 LOC. Sản phẩm hoàn thành trước thời gian là 1 tháng, chi phí gần bằng với chi phí ước lượng, việc chênh lệch trong lịch biểu giảm từ 22% xuống còn 2.7%. [7]

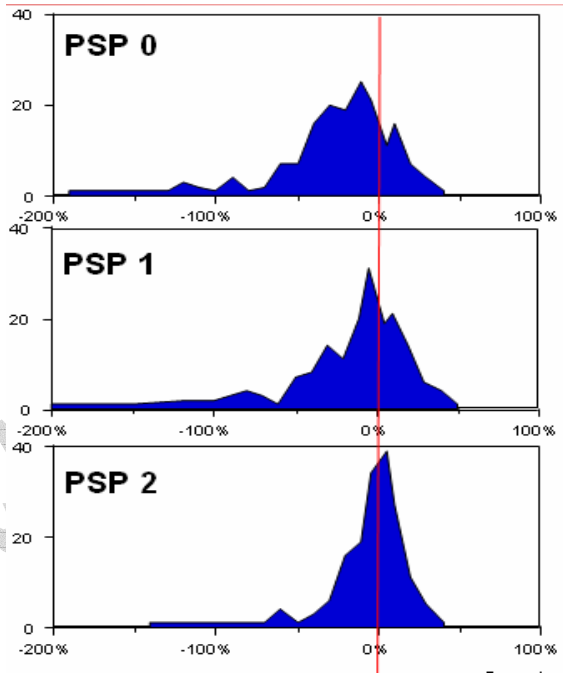
4.2 Trong các trường đại học

Năm 1997, viện SEI tổ chức 1 khoá dạy PSP gồm 298 sinh viên (trong đó có cả kỹ sư). Quy trình huấn luyện cũng đi từ PSP0 đến PSP2. Sau khi kết thúc khoá học, Watt S.Humphrey đã thu thập được dữ liệu từ hơn 30000 chương trình viết có sử dụng PSP:



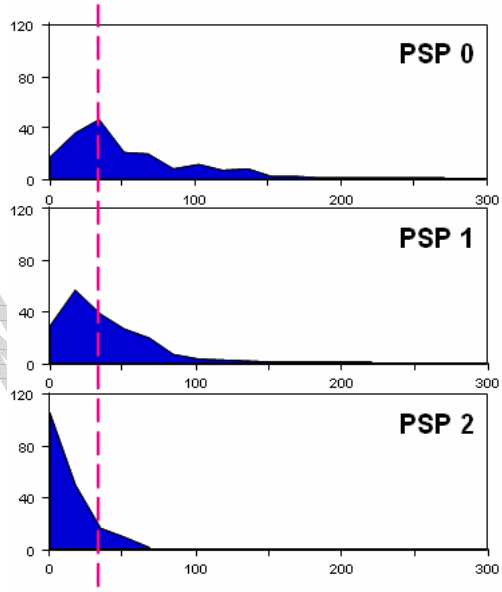
Hình 4.2.1 Độ chính xác trong ước lượng kích thước

Trong hình trên, ở những giai đoạn đầu khi làm quen với PSP, việc ước lượng còn chưa được chính xác. Mức độ chênh lệch giữa ước lượng và thực tế còn cao. Lí do là lúc này các kỹ sư vẫn chưa có nhiều dữ liệu kích thước của chương trình cũ. Công việc ước lượng dựa chủ yếu trên sự phán đoán. Càng ở những giai đoạn về sau (cấp độ PSP sau), các kỹ sư đã có dữ liệu nên khả năng ước lượng chính xác hơn.



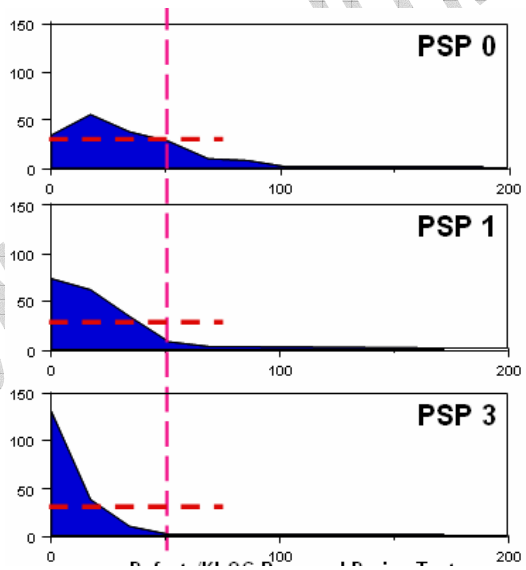
Hình 4.2.2 Độ chính xác trong ước lượng thời gian

Trong hình trên, ở những qui trình PSP0, số các chương trình có phần ước lượng sai sót so với thực tế rất nhiều và sự chênh lệch cũng cao (biểu hiện bằng phần bên trái trục Kích thước nhiều hơn rất nhiều bên phải trục kích thước). Ở cấp độ PSP1, số lượng các chương trình sai sót trong dự đoán kích thước đã giảm xuống và đến cấp độ thì cả hai phần xấp xỉ gần bằng nhau.

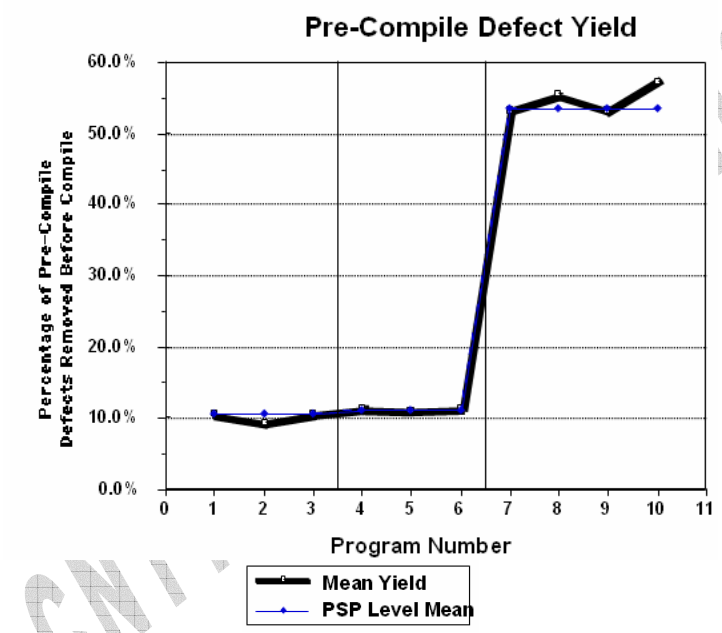


Hình 4.2.3 Số sai sót/KLOC được loại bỏ trong pha biên dịch

Trong hình trên, trục X biểu thị số sai sót/KLOC, còn trục đứng (y) biểu diễn số kỹ sư báo về mật độ sai sót ở một giá trị x. Cũng tương tự như những đánh giá ở trên, về mặt tổng thể càng ở những cấp độ sau thì số sai sót/KLOC được loại bỏ trong pha biên dịch cũng giảm xuống và số kỹ sư phạm phải nhiều sai sót cũng giảm đáng kể.



Hình 4.2.4 Số sai sót/KLOC được loại bỏ trong pha kiểm chứng

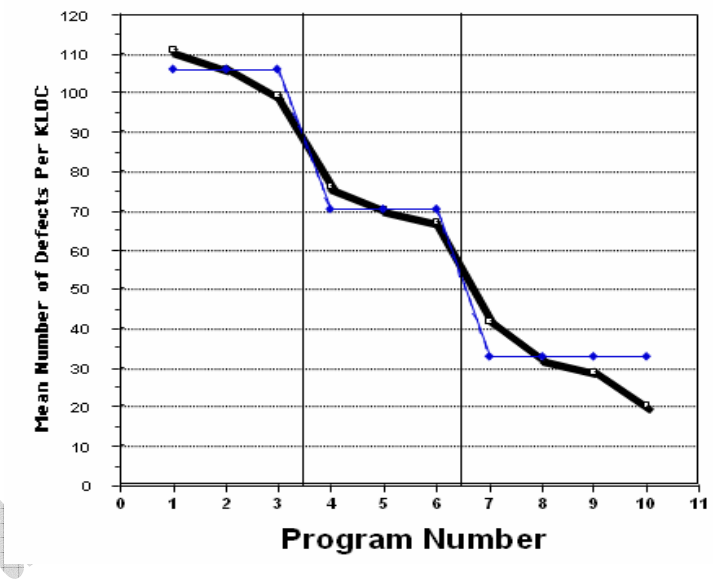


Hình 4.2.5 Chất lượng qui trình

Mean Yield: hiệu suất trung bình với chương trình không sử dụng phương pháp PSP.

PSP Level Yield: hiệu suất của những chương trình có sử dụng PSP.

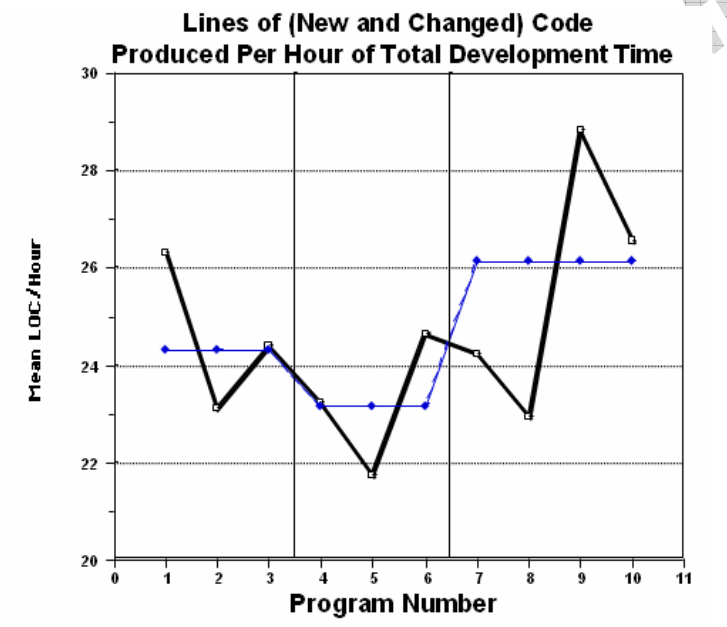
Khi không sử dụng PSP, mặc dù phần trăm số sai sót được loại bỏ trước pha biên dịch có lúc cao lúc thấp không ổn định. Còn với những chương trình có sử dụng PSP, phần trăm tăng đều và tương đối ổn định.



Hình 4.2.6 Chất lượng sản phẩm

Hình trên mô tả số sai sót/KLOC. Những chương trình từ 1 – 3 nằm trong cấp độ PSP0 và PSP0.1, 4 – 6 nằm trong cấp độ PSP1 và PSP1.1, từ 7 – 9 nằm trong PSP2 và

PSP2.1. Các chương trình còn lại là 10 – 11 nằm trong cấp độ PSP3. Như vậy ở những cấp độ đầu thì do chưa quen nên số sai sót còn nhiều. Ở những cấp độ sau, số sai sót/KLOC giảm xuống và có xu hướng đi vào ổn định.



Hình 4.2.7 Hiệu suất công việc

Hình trên đề cập đến hiệu suất làm việc của các kỹ sư (LOC/giờ) trong quá trình thực hiện PSP. Ở những cấp độ đầu PSP0 và PSP0.1, hiệu suất không tăng mà có vẻ ổn định. Ở những cấp độ về sau (từ 7 – 9), số LOC/giờ làm việc tăng và có xu hướng đi vào ổn định. Tuy nhiên, với những chương trình không sử dụng PSP, số LOC/giờ làm việc có thể cao hơn hay thấp hơn nhưng không ổn định và biến động khá nhiều.

Một ví dụ khác về hiệu quả của PSP. Trong những năm 1995 và 1996, SEI đã tổ chức hai khoá dạy PSP (một khoá vào hè 1995 và khoá còn lại vào mùa đông năm 1996). Sau đó viện SEI đã phân tích để đánh giá hiệu quả của PSP. Dữ liệu phân tích được lấy từ những chương trình của 124 sinh viên trong khoá học và dữ liệu của những sinh viên này trong vòng 6 tháng sau khi kết thúc khoá học. Kết quả như sau: [8]

	Số sai sót/KLOC Chương trình 1,2,3 (PSP0 và PSP0.1)	Số sai sót/KLOC Chương trình 8,9,10 (PSP2 và PSP2.1)	% Sự cải tiến
Lớp 1	93	66	29
Lớp 2	50	40	20
Tổng kết	72	52	27.7

Bảng 4.2.1 Kết quả khóa học PSP

Với lớp 1: Ở các chương trình đầu, số sai sót là 93, còn ở các chương trình sau, sai sót giảm xuống còn 66 => % sự cải tiến là: (93 - 66)*100%/93 = 29 %

Với lớp 2: Ở các chương trình đầu, số sai sót là 50, còn ở các chương trình sau, sai sót giảm xuống còn 40 => % sự cải tiến là: (50 - 40)*100%/50 = 20 %

Tổng cộng, ở các chương trình đầu, số sai sót là 72, còn ở các chương trình sau, sai sót giảm xuống còn 52 => % sự cải tiến là: (72 - 52)*100%/72 = 27.7 %

4.3 Kết quả áp dụng PSP của bản thân.

4.3.1 Hướng áp dụng

Để thử nghiệm tính hiệu quả của các phương pháp luận sử dụng trong PSP, chúng tôi đã bắt tay vào ghi nhận thời gian thực hiện của việc: đọc tài liệu. Thời gian tiến hành thử nghiệm bắt đầu từ ngày 1/5/2005 và kết thúc ngày 30/6/2005.

Với phần đọc tài liệu, mục tiêu của chúng tôi là:

- Nghiên cứu các cấp độ của PSP.
- Nghiên cứu phương pháp ước lượng PROBE.
- Nghiên cứu kết quả áp dụng PSP trong thực tiễn.

Với phần đọc tài liệu, chúng tôi sử dụng biểu mẫu sau để ghi nhận thông tin:

Tên: Trương thị Ngọc Phượng **Ngày:**

Công việc #	Ngày	Tiến trình	Ước lượng		Thực tế			Đến ngày				
			Thời gian	Đơn vị	Thời gian	Đơn vị	Tốc độ	Thời gian	Đơn vị	Tốc độ	GTLN	GTNN
1	Mô tả	Đọc file										

Bảng 4.3.1 Bản ghi thời gian

4.3.2 Kết quả thu được

4.3.2.1 Phần đọc tài liệu

Cuối tuần, chúng tôi thực hiện việc đánh giá. Việc đánh giá cho 1 tuần được thực hiện bằng cách tính tỉ lệ chênh lệch giữa thời gian ước lượng và thời gian thực tế. Một ví dụ về đánh giá hiệu quả cho tuần 1 (1/5/2005 đến 7/5/2005)

Tên: Trương thị Ngọc Phượng Ngày 1/5/2005

Công việc #	Tiến trình	Ước lượng		Thực tế			Đến ngày				
		Thời gian	Đơn vị	Thời gian	Đơn vị	Tốc độ	Thời gian	Đơn vị	Tốc độ	GTLN	GTNN
1	Đọc tài liệu	20	25	60	25	2.4	60	25	2.4	2.4	2.4
	Mô tả : Đọc file se_psp.pdf [9]										
2	Đọc tài liệu	12	6	30	6	5	90	31	2.9	5	2.4
	Mô tả : Đọc file : KR.pdf [10]										
3	Đọc tài liệu	228	76	180	76	2.37	270	107	2.52	5	2.37
	Mô tả : Đọc file Empirical Study of PSP.pdf [11]										
4	Đọc tài liệu	140	55	300	55	5.45	570	162	3.52	5.45	2.37
	Mô tả : Đọc file 00tr022.pdf [12]										
5	Đọc tài liệu	120	34	60	34	1.76	630	196	3.21	5.45	1.76
	Mô tả : Đọc file Tutorial Using PSP0.ppt [13]										
6	Đọc tài liệu	48	16	15	16	0.938	645	212	3.04	5.45	0.938
	Mô tả : Đọc file Tutorial Using PSP0.1.ppt [14]										
7	Đọc tài liệu	72	24	30	24	1.25	675	236	2.86	5.45	0.938
	Mô tả : Đọc file Tutorial Using PSP1.ppt [15]										
8	Đọc tài liệu	35	12	30	12	2.5	705	248	2.84	5.45	0.938
	Mô tả : Đọc file Tutorial Using PSP1.1.ppt [16]										
9	Đọc tài liệu	50	18	50	18	2.78	755	266	2.84	5.45	0.938
	Mô tả : Đọc file Tutorial Using PSP2.ppt [17]										
10	Đọc tài liệu	35	12	30	12	2.5	785	278	2.82	5.45	0.938
	Mô tả : Đọc file Tutorial Using PSP2.1.ppt [18]										

Kết quả thực hiện của tuần 1 như sau:

Tài liệu	(Thời gian thực tế - Thời gian ước lượng)/Thời gian thực tế
1	66.67%
2	60%
3	-26.67%
4	53.33%
5	- 100%
6	- 220%
7	-140%
8	-16.67%
9	0%
10	-16.67%

Bảng 4.3.2 Kết quả thực hiện trong 1 tuần

Độ chênh lệch ước lượng trung bình của cả tuần: (Tổng thời gian thực tế - Tổng thời gian ước lượng)/Tổng thời gian ước lượng = (785 – 760)*100%/785 = 3.18%

Kết quả thực hiện sau 8 tuần thực hiện

Tuần	Độ chênh lệch ước lượng trung bình
1	3.18%
2	18.25%
3	2.45%
4	1.32%
5	-1.35%
6	-0.22%
7	- 0.12%
8	-0.03%

Bảng 4.3.3 Kết quả thực hiện sau 8 tuần

Mặc dù độ chênh lệch chưa đạt được giá trị 0% (ước lượng bằng thực tế) nhưng càng ở những tuần sau thì con số ước lượng gần đúng với thực tế hơn. Mặc dù những tỉ lệ âm không phải là tối ưu nhưng ít ra con số đó cũng phản ảnh việc chúng tôi hoàn thành công việc trước thời gian qui định.

4.4 Kết luận về việc sử dụng PSP

Việc áp dụng PSP trong môi trường công nghiệp đem lại hiệu quả khá lớn. Tuy nhiên, tính đến thời điểm hiện nay thì qui trình này vẫn chưa được sử dụng rỗng rãi ở mức toàn cầu. Lý do là vì mức độ nghiêm ngặt của nó. Mặc dù PSP chỉ áp dụng trên mức độ cá nhân, nhưng việc áp dụng PSP sẽ làm thay đổi thói quen làm việc của cá nhân nên trong công nghiệp, việc huấn luyện cùng một lúc số lượng lớn các kỹ sư gặp khó khăn. Thêm vào đó, một yếu tố khác ảnh hưởng đến kết quả thực hiện PSP là mức độ phức tạp của công việc trong thực tế. Ở những khoá học dạy PSP, các kỹ sư (sinh viên) được hướng dẫn cách thực hiện PSP trên những bài tập (chương trình nhỏ). Do đó, trong các pha phân tích, lập kế hoạch, thiết kế dễ kiểm soát. Khi áp dụng vào trong công việc đòi hỏi kỹ sư phải có kinh nghiệm.

Tháng 12/2000, giáo sư Alan R.Hevner kết hợp với viện SEI đã thực hiện một cuộc khảo sát đánh giá việc sử dụng PSP trong thực tế đối với các công ty và kỹ sư đã được huấn luyện PSP. Kết quả thu như sau:[6]

	Giá trị
Số dự án sử dụng PSP	Trung bình 24 Khoảng chênh lệch 0 – 18
PSP được sử dụng như thế nào	
Chỉ những dự án quan trọng	36%
Chỉ những dự án không quan trọng	25%
Những dự án quan trọng và không quan trọng	39%
Phần trăm dự án sử dụng PSP	
0%	10%
1% đến 25%	26%
26% đến 50%	13%
51% đến 75%	14%
Nhiều hơn 75%	37%
Loại dự án sử dụng PSP	
Chỉ chứng minh khái niệm (proof of concept)	31%
Chỉ những dự án nhỏ	10%
Cả những dự án lớn và nhỏ	28%
Chỉ những dự án lớn	7%
Tất cả các dự án	24%
Những pha trong dự án sử dụng PSP	
Pha xác định yêu cầu	32%
Pha phân tích	40%
Pha thiết kế	80%
Pha thực thi chương trình	92%
Pha kiểm chứng	65%
Pha bảo trì	24%

Bảng 4.4.1 Kết quả khảo sát đánh giá việc sử dụng PSP

Đối với cá nhân chúng tôi, sau khi thử áp dụng thực hiện phương pháp quản lý và ước lượng thời gian của PSP, chúng tôi gặp một số khó khăn sau:

Đối với phần ước lượng: Khi đọc các file pdf sử dụng font chữ lớn số lượng trang nhiều nhưng nội dung thì ít. Cũng có những file pdf, mặc dù ít trang nhưng font chữ nhỏ và nội dung có đề cập đến nhiều vấn đế quan trọng. Do đó, con số ước lượng dựa vào công thức (số trang/phút của những tài liệu trước) * số trang của tài liệu hiện tại đôi khi cho kết quả chênh lệch rất nhiều.

Đối với phần theo dõi thời gian thực hiện công việc: PSP sử dụng đơn vị là phút. Trong quá trình theo dõi, đôi khi cũng có lúc quên cập nhật dữ liệu. Đến khi nhớ ra thì lúc này con số ghi nhận chỉ ở mức độ tương đối và có thể là chênh lệch nhiều. Hơn nữa, khi sử dụng đơn vị là phút, chúng tôi thường phải nhẩm xem tương ứng với số phút đó là bao nhiêu giờ thì mới có thể hình dung được độ lớn của khoảng thời gian.

Thời gian đầu khi thực hiện công việc theo dõi, chúng tôi cảm thấy khó khăn. Vấn đề khó khăn ở đây không hẳn chỉ là các con số ghi nhận mà việc ghi nhận là một việc hoàn toàn mới mà trước đây chúng tôi ít dùng. Sau khoảng 2 tuần đầu, khi thực hiện đánh giá kết quả hàng tuần chúng tôi thấy hiệu quả của việc ước lượng có được cải thiện đôi chút và lúc này, việc ghi nhận bước đầu cũng đã dần dần không còn khó thực hiện như trước nữa. Tuy nhiên, cần phải thực hiện trong một khoảng thời gian ít nhất là 3 tháng thì thói quen này mới có thể hình thành được.

Chương 5. Ứng dụng minh họa

5.1 Giới thiệu

Qui trình phần mềm cá nhân ít nhiều đã chứng minh được hiệu quả trong việc cải thiện chất lượng công việc của từng cá nhân.

PSP dựa trên những nguyên lý quản lý về thời gian và sai sót một cách chi tiết và tỉ mỉ. Chính vì thế, nếu sử dụng thành thục PSP, người kỹ sư sẽ ngày càng nâng cao khả năng kiểm soát lịch biểu và công việc của mình.

Với mong muốn có thể áp dụng các nguyên lý của PSP, chúng tôi dự định xây dựng một ứng dụng để hỗ trợ cá nhân quản lý công việc và cũng là để minh họa cho nghiên cứu của mình. Tuy nhiên, để phát triển một chương hoàn chỉnh theo dự định thì phải mất rất nhiều thời gian và công sức. Hơn nữa, thời gian thực hiện của một luận văn lại không cho phép. Do đó, chúng tôi chỉ viết một ứng dụng nhỏ minh họa và hỗ trợ một phần nào đó mà thôi. Cụ thể chúng tôi sẽ xây dựng một ứng dụng hỗ trợ việc lên kế hoạch dự án, ghi lại dữ liệu của việc thực hiện kế hoạch đó để làm dữ liệu lịch sử cho các ước lượng sau. Sau khi kết thúc dự án, ứng dụng sẽ đưa ra một số nhận xét về việc lên kế hoạch của cá nhân đó.

Ứng dụng này được xây dựng trên Web để giúp các thành viên xây dựng dự án có thể làm việc ở bất cứ nơi đâu và dễ dàng trao đổi thông tin về dự án. Hơn nữa, sau này ứng dụng có thể được phát triển tiếp để hỗ trợ nhiều chức năng quản lý phần mềm khác trên Web.

5.2 Yêu cầu

Ứng dụng được xây dựng có những yêu cầu sau:

1. Hệ thống được cài đặt ở một máy chủ trong phòng SELAB. Các thành viên trong phòng khi muốn sử dụng sẽ đăng nhập vào hệ thống.
2. Đối với các thành viên của hệ thống:
   a. Các user chưa có account có thể cung cấp thông tin cụ thể để đăng ký là thành viên hệ thống.
   b. Admin có thể tạo hàng loạt account cho các user.
   c. Hệ thống cho phép quản lý thông tin các thành viên thuộc phòng SELAB và một số thành viên không thuộc phòng SELAB nhưng có vai trò quan trọng trong các dự án của phòng.

d. Hệ thống cho phép quản lý thông tin các nhóm đang tham gia các dự án.

e. Hệ thống chỉ phân 2 quyền: admin, user.

f. Admin có thể tạo các các group cho các dự án và chỉ định project manager.

g. Các user có thể chỉnh sửa những thông tin cá nhân thông qua hệ thống.

h. Tạo mới các template thông tin cho các email gửi cho các account.

3. Đối với các dự án:

a. Quyền hạn admin:

- Tạo mới các project kèm theo các thông tin hỗ trợ cho việc thực hiện dự án.
- Phân công việc cho các thành viên trong dự án.
- Khi gặp một dự án có một số chức năng giống như những chương trình trước, admin có thể xem lại thông tin như thời gian thực tế, thời gian ước lượng đầu tiên, user thực hiện để lên kế hoạch cho dự án mới.
- Khi bắt đầu thực hiện dự án, admin sẽ phân dự án thành các công việc nhỏ hơn và giao cho các thành viên. Admin cũng đưa ra những con số ước lượng về thời gian thực hiện cho từng công việc nhỏ.
- Trong quá trình thực hiện dự án, admin có thể cập nhật lại con số ước lượng.
- Khi kết thúc dự án, admin sẽ được cung cấp những thông tin: thời gian thực tế đã sử dụng cho mỗi công việc nhỏ, số lần chỉnh sửa ước lượng. Từ những đối chiếu này, người kỹ sư sẽ có cái nhìn tổng quan hơn về hiệu quả của việc ước lượng công việc của mình. Và qua đó cũng sẽ nâng dần khả năng dự đóan chính xác cho những dự án sau.
- Bên cạnh đó, khi kết thúc dự án, admin có thể đưa ra các nhận xét, các ý kiến để cải tiến hơn hiệu quả qui trình. Phần này sẽ có lợi cho bản thân và là sưu liệu cho các dự án sau.

b. Quyền hạn user.

- Có thể xem các thông tin về dự án sẽ tham gia (không có phần phân công việc)
- Xem thông tin các dự án đã tham gia, thông tin chi tiết các công việc nhỏ hơn do bản thân các kỹ sư đã phân ra để thực hiện dự án, kèm theo các thông tin về thời gian thực tế và ước lượng.
- Khi nhận được công việc từ project manager, user sẽ lên danh sách các công việc nhỏ hơn để thực hiện kèm theo các con số ước lượng. Các thông tin này sẽ được ghi nhận vào hệ thống.

- Trong quá trình thực hiện, user có thể biết được tổng thời gian đã sử dụng cho mỗi công việc tính đền thời điểm đang xét.
- Trong quá trình thực hiện, user có thể chỉnh sửa các con số ước lượng đã lưu trong hệ thống.
- Khi kết thúc dự án, user sẽ được hệ thống đưa ra những con số thực tế đã sử dụng cho mỗi công việc nhỏ, số lần chỉnh sửa ước lượng.

c. Ngoài ra, cả user và admin đều có thể trao đổi thắc mắc và khó khăn thông qua forum (mỗi project sẽ có một forum).

4. Đối với việc quản lý thời gian

a. Các user, admin đều có thể lên lịch biểu cho các công việc hàng tuần của mình. Loại công việc gồm có 3 loại: Fixtime (những công việc bắt buộc phải thực hiện đúng giờ và thường là ít không thay đổi như ăn cơm, tham gia một lớp học trong một học kỳ…), thời gian trong một dự án nào đó, và thời gian để thực hiện các loại công việc khác (như xin visa, đám cưới ….)

b. Mỗi ngày, các user, admin sẽ ghi nhận lại chi tiết thời gian sử dụng vào trong hệ thống.

c. Các user, admin có thể xem lại lịch làm việc đã lên trong ngày, trong tuần, trong tháng.

d. Hàng tuần, user, admin có thể xem lại bảng thống kết quỹ thời gian dùng cho các công việc là bao nhiêu để biết được tình hình làm việc của bản thân.

5. Các chức năng khác

a. Vì việc ghi nhận thời gian là liên tục trong ngày, nếu mỗi lần ghi nhận phải truy cập vào hệ thống thì tốn khá nhiều thời gian. Do đó, user, admin sẽ ghi thông tin vào một ứng dụng trên Window. Sau đó sẽ Export ra XML và import vào trong server.

b. Đối với thông tin các dự án đang được thực hiện thì cũng được export từ server ra XML và import vào trong ứng dụng Window

c. Ngoài ta các thành viên có thể trao đổi những tin nhắn với nhau qua hệ thống.

5.3 Bảng chú giải

5.3.1 Giới thiệu

Tài liệu này được dùng để định nghĩa các thuật ngữ đặc thù trong lĩnh vực của bài toán, giải thích các từ ngữ có thể không quen thuộc đối với người đọc trong các mô tả use case hoặc các tài liệu khác của dự án. Thường thì tài liệu này có thể được dùng như một từ điển dữ liệu không chính thức, ghi lại các định nghĩa dữ liệu để các mô tả use case và các tài liệu khác.

5.3.2 Các định nghĩa

Bảng chú giải này bao gồm các định nghĩa cho các khái niệm chính trong Hệ thống.

5.3.2.1 User

Người tham gia vào hệ thống và có khả năng thực hiện một số chức năng của hệ thống.

5.3.2.2 Admin

Cũng là người tham gia vào trong hệ thống. Ngoài những chức năng mà user có, admin còn có thêm một số chức năng tạo mới các account hay chấp nhận account tham gia hệ thống.

5.3.2.3 Project manager

Cũng là người tham gia vào trong hệ thống. Ngoài những chức năng mà user có, project manager còn có thêm một số chức năng liên quan đến việc tạo và phân bố công việc trong các dự án cho các user.

5.3.2.4 Time recording log

Bảng ghi nhận thời gian thực hiện cho các công việc. Các công việc này có thể là các công việc thuộc một dự án hay là một công việc mà người dùng cần thực hiện.

5.3.2.5 Weekly Activity Summary

Bảng tổng kết hoạt động trong tuần, bao gồm danh sách các công việc đã lập kế hoạch và thời gian thực hiện được ghi nhận trong bảng Time recording log.

5.3.2.6 Task

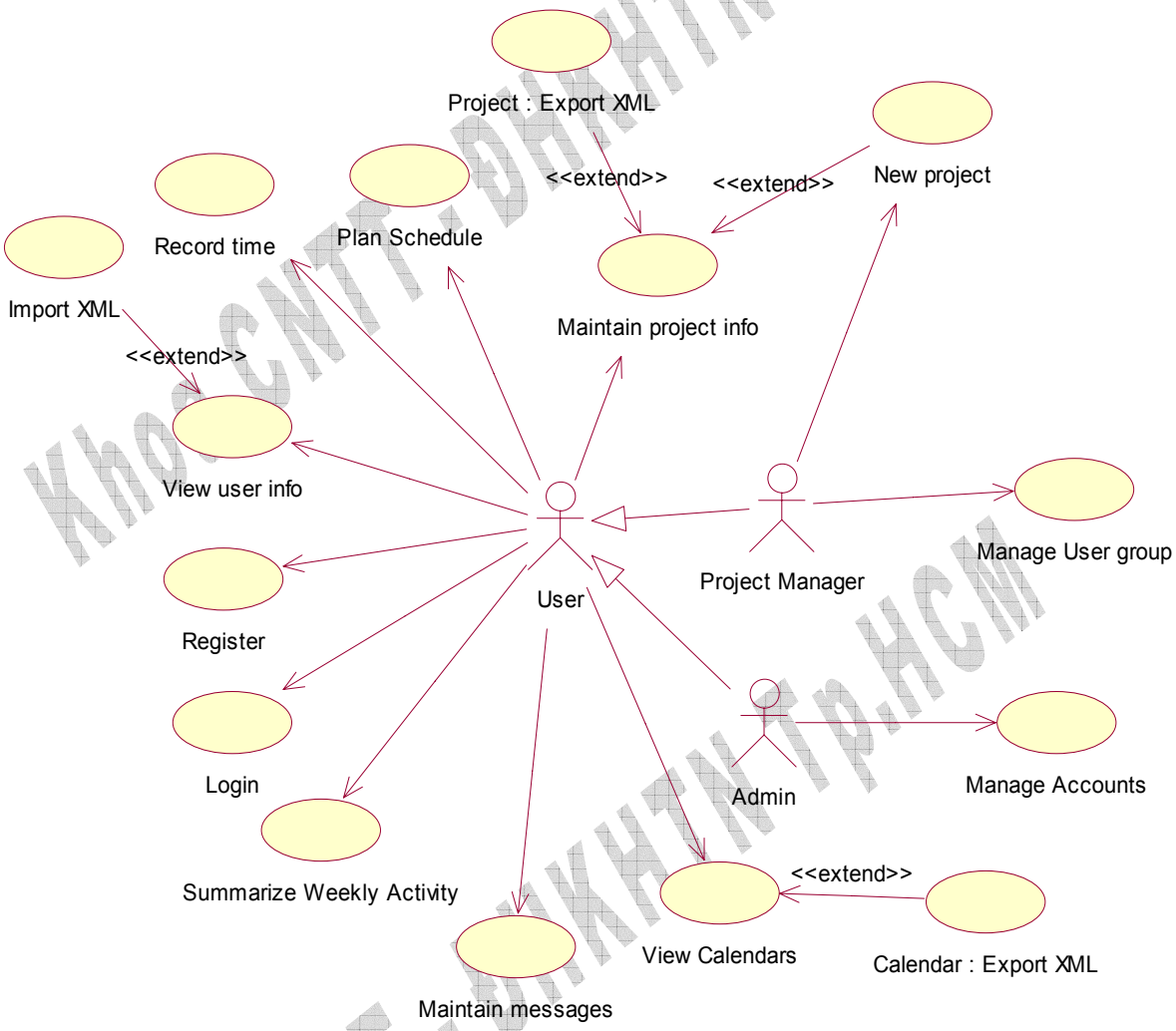
Với mỗi dự án (project), project manager sẽ phân ra thành các module nhỏ hơn và giao cho các user. Các module này được xem là các task đối với các user.

5.3.2.7 To do

Khi user nhận được các task thì họ sẽ phân ra thành các công việc nhỏ hơn cần thực hiện để hoàn thành task. Các công việc nhỏ hơn này là To do

5.4 Thiết kế

5.4.1 Use case

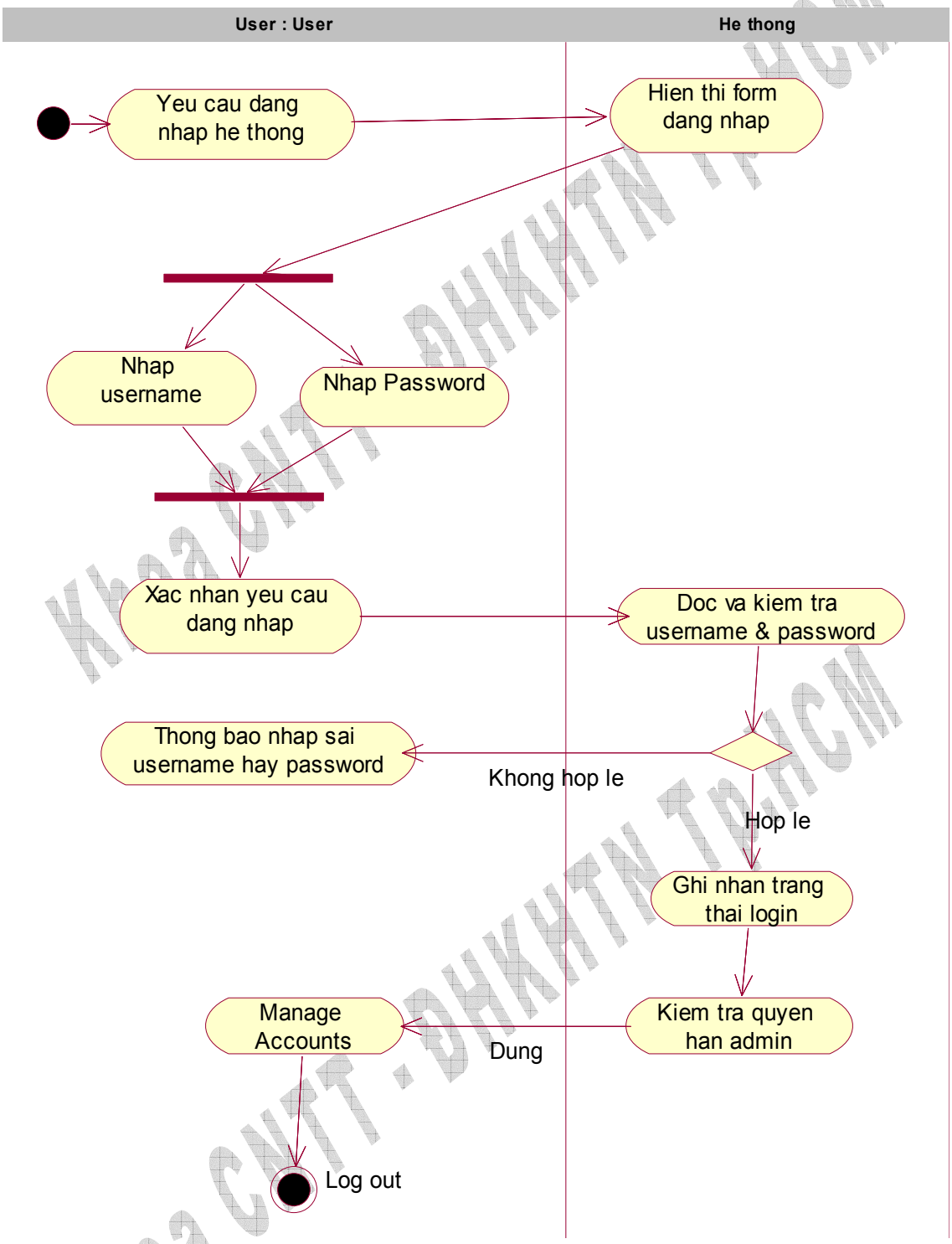


Hình 5.4.1 Mô hình use case của ứng dụng

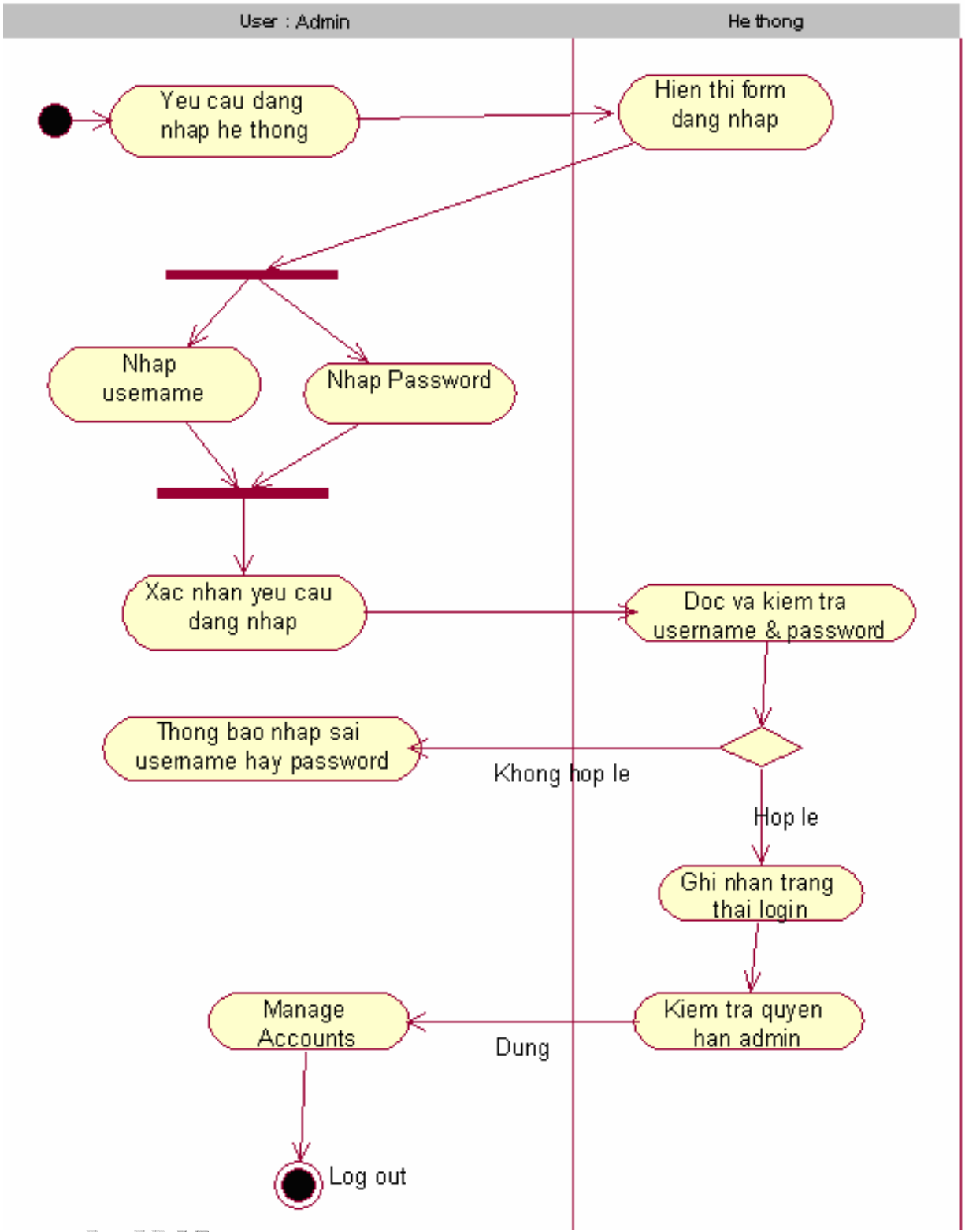
5.4.2 Đặc tả bổ sung

Xem tập tin đính kèm: Baocaochinh\Dactabosung.doc

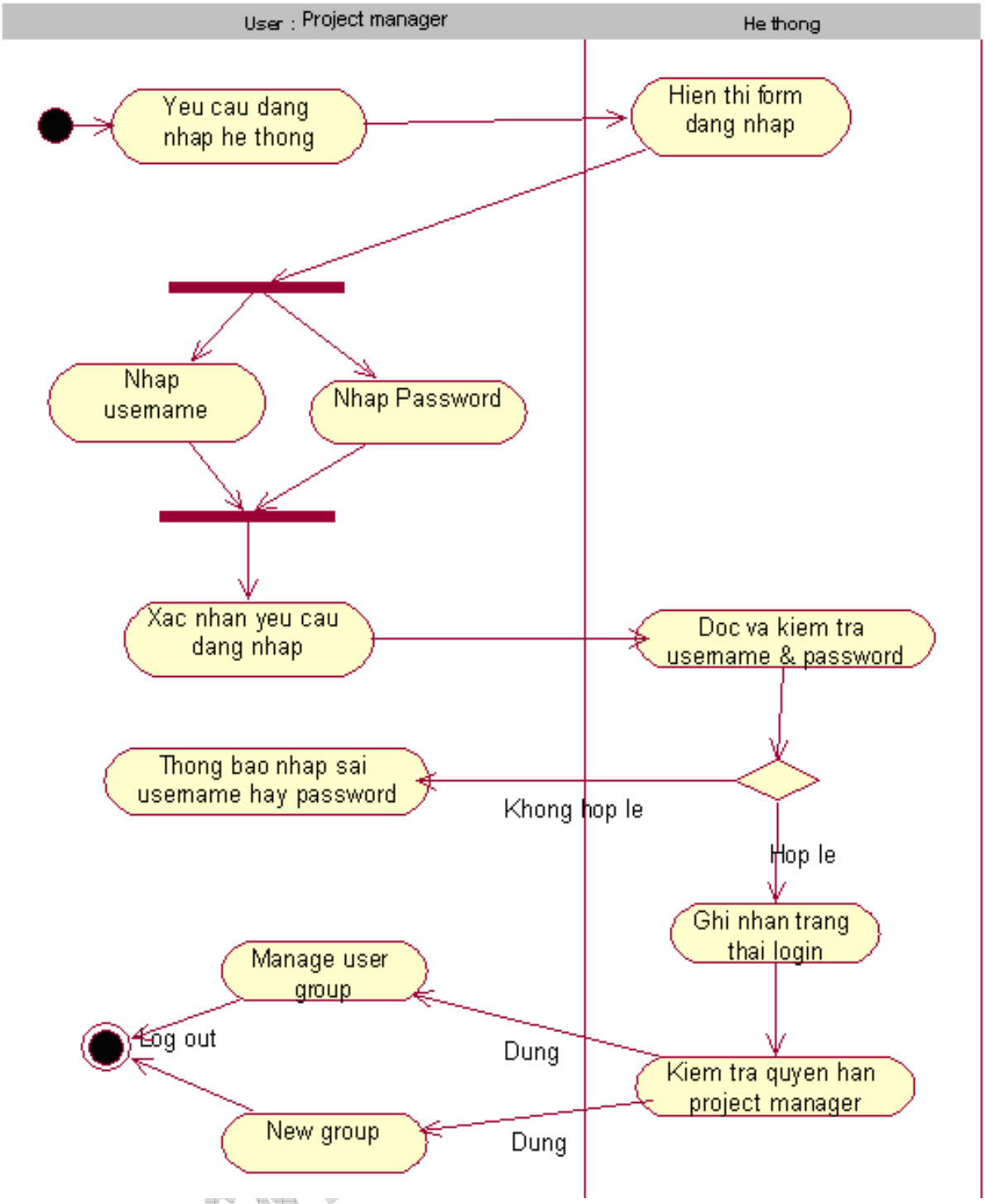
5.4.3 Các activity diagram chính trong ứng dụng



Hình 5.4.2 Activity Diagram - Các chức năng cho user



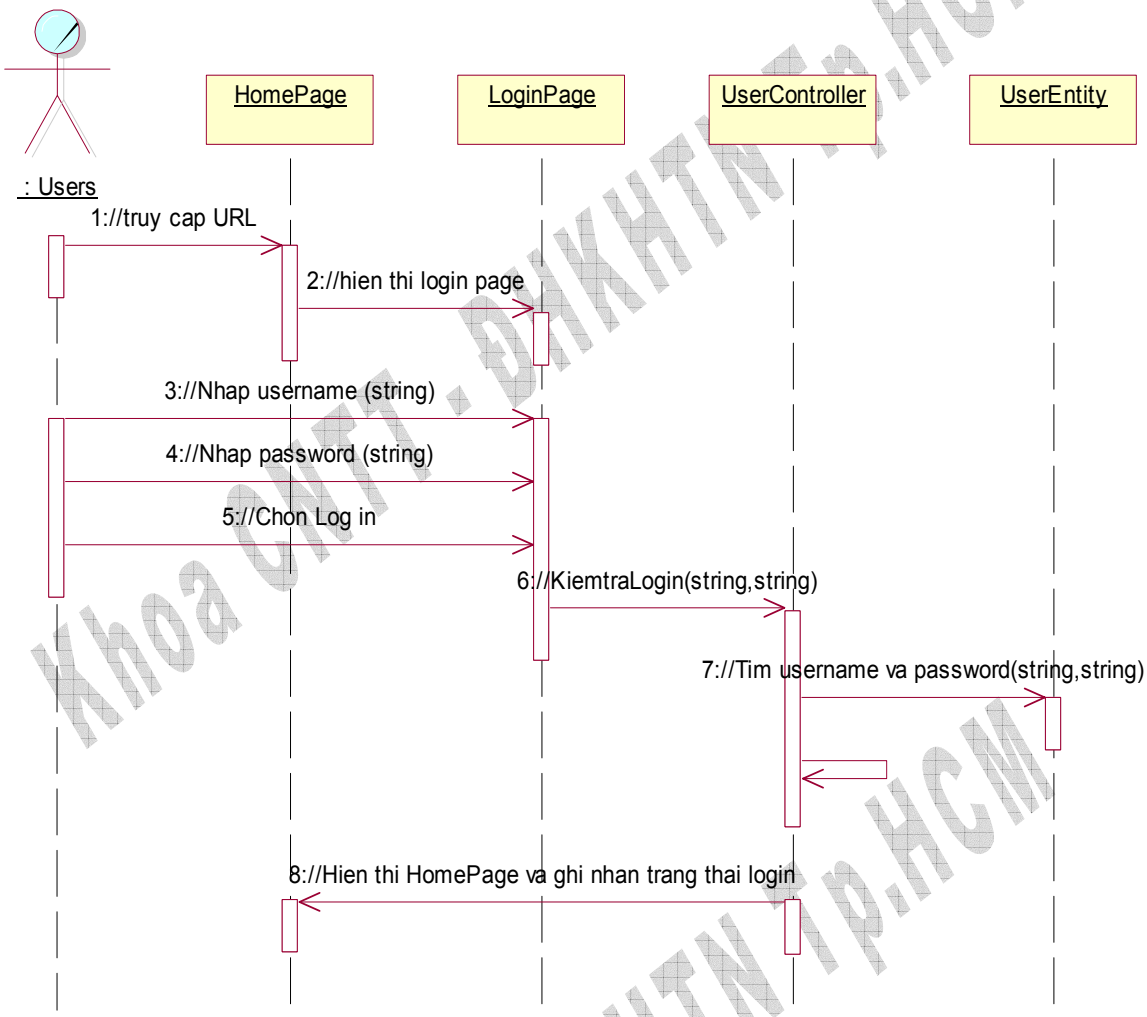
Hình 5.4.3 Activity Diagram - Chức năng cho admin



Hình 5.4.4 Activity Diagram - Chức năng cho project manager

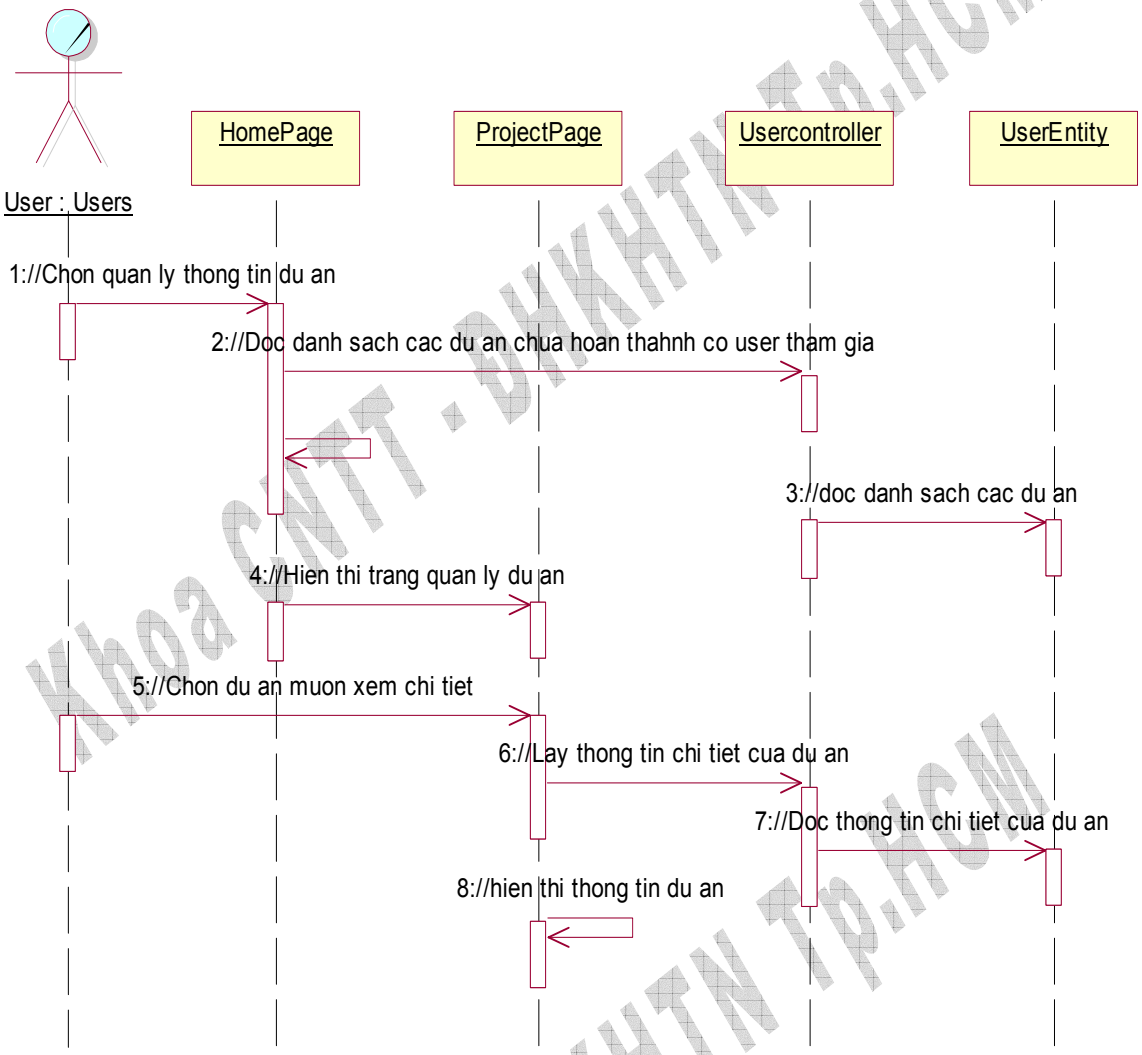
5.4.4 Các sequence diagram chính trong ứng dụng

5.4.4.1 Log in



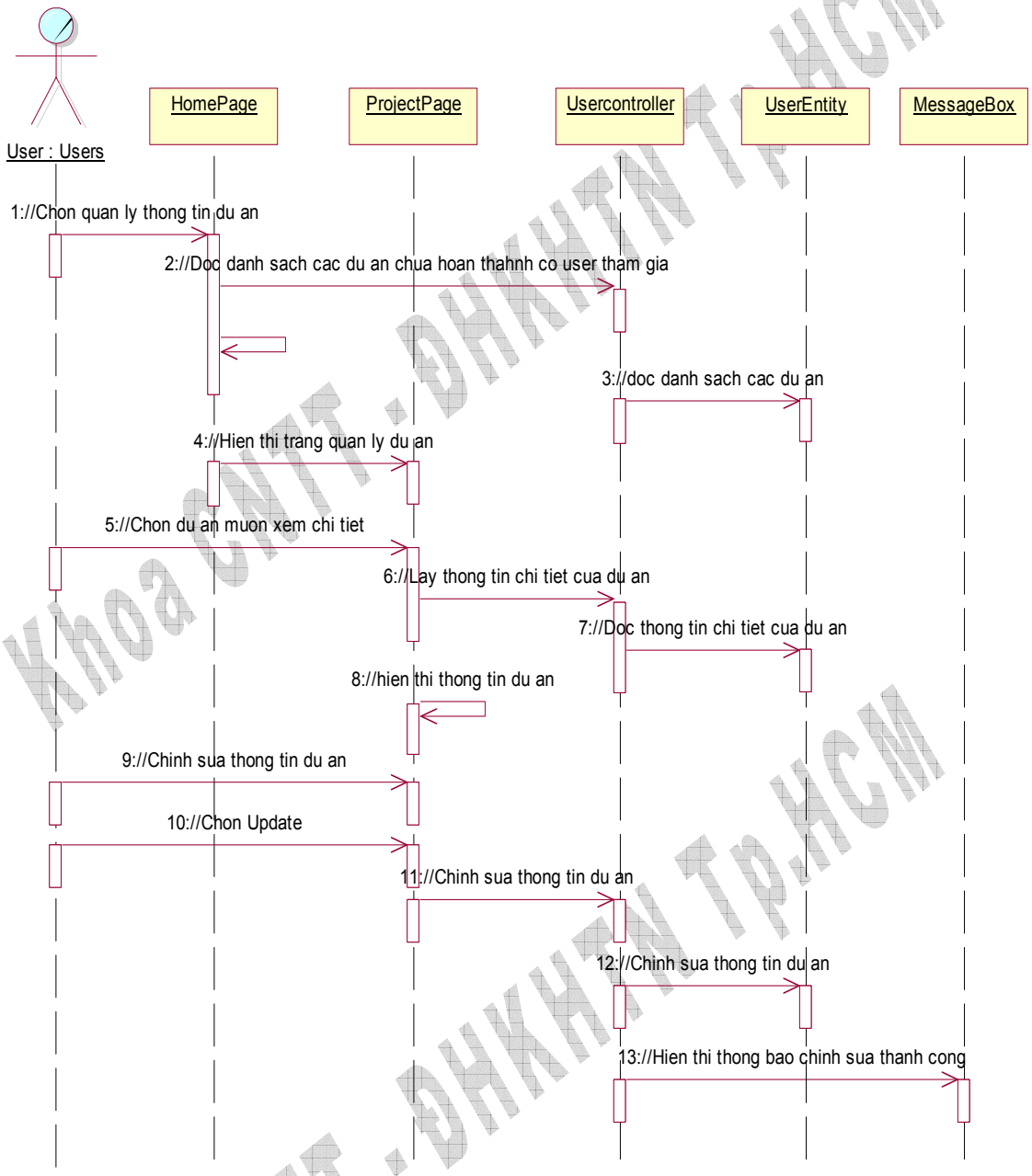
Hình 5.4.5 Sequence Diagram - Log in

5.4.4.2 View Project info



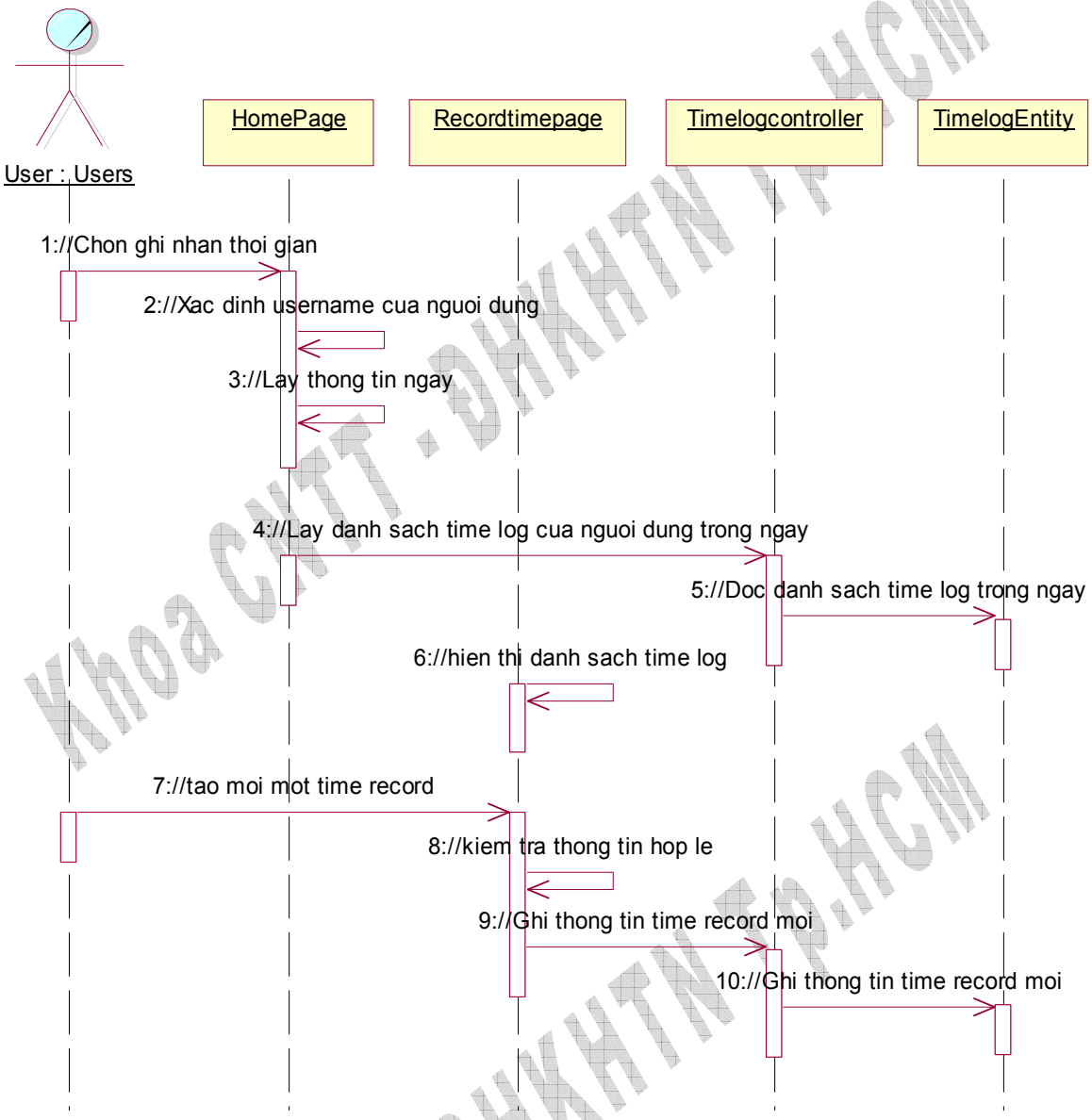
Hình 5.4.6 Sequence Diagram - View Project Info

5.4.4.3 Chỉnh sửa thông tin dự án



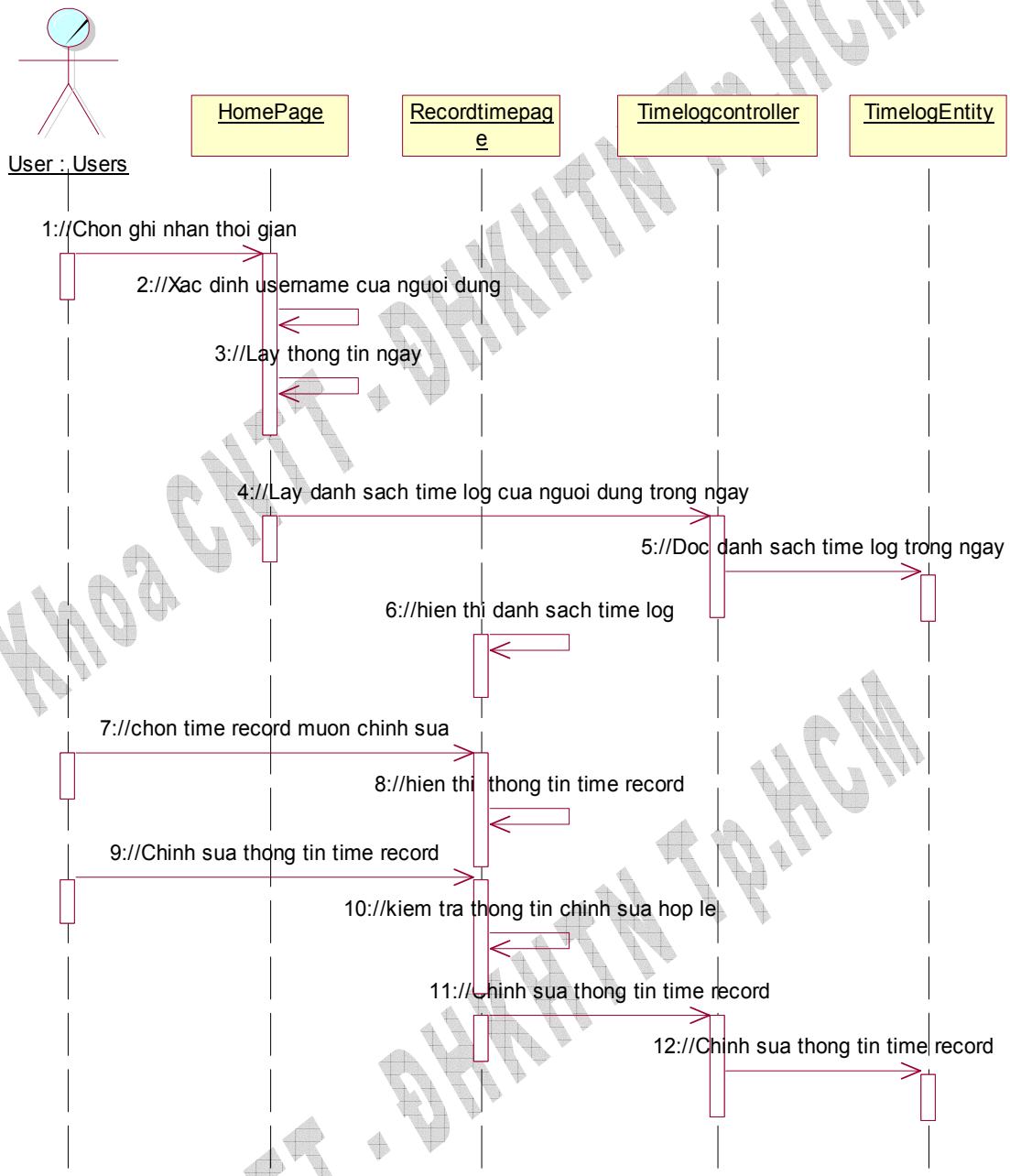
Hình 5.4.7 Sequence Diagram - Chỉnh sửa thông tin dự án

5.4.4.4 Thêm mới time record



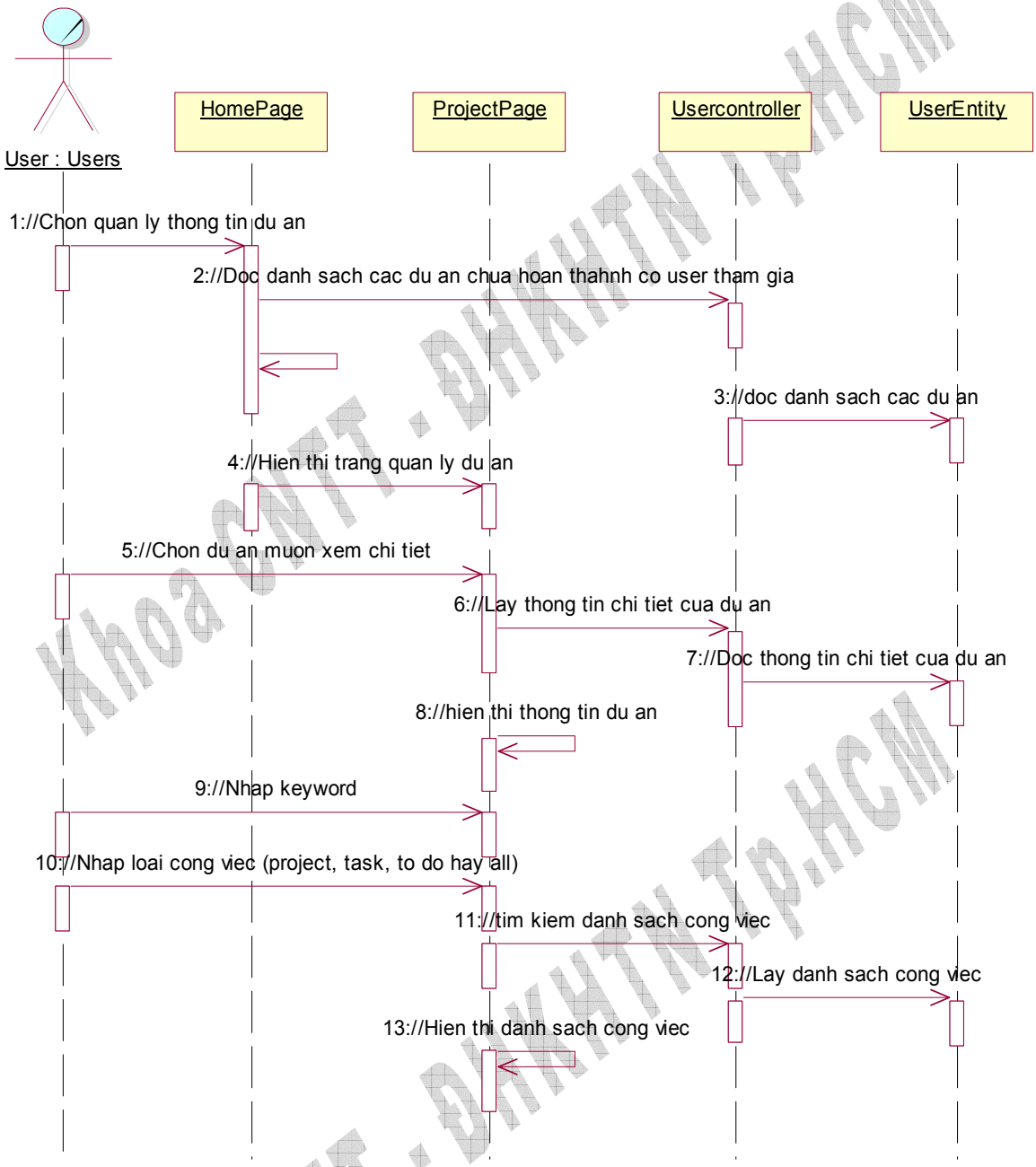
Hình 5.4.8 Sequence Diagram - Thêm mới record

5.4.4.5 Chỉnh sửa thông tin time record



Hình 5.4.9 Sequence Diagram - Chỉnh sửa thông tin time record

5.4.4.6 Tìm kiếm thông tin dự án

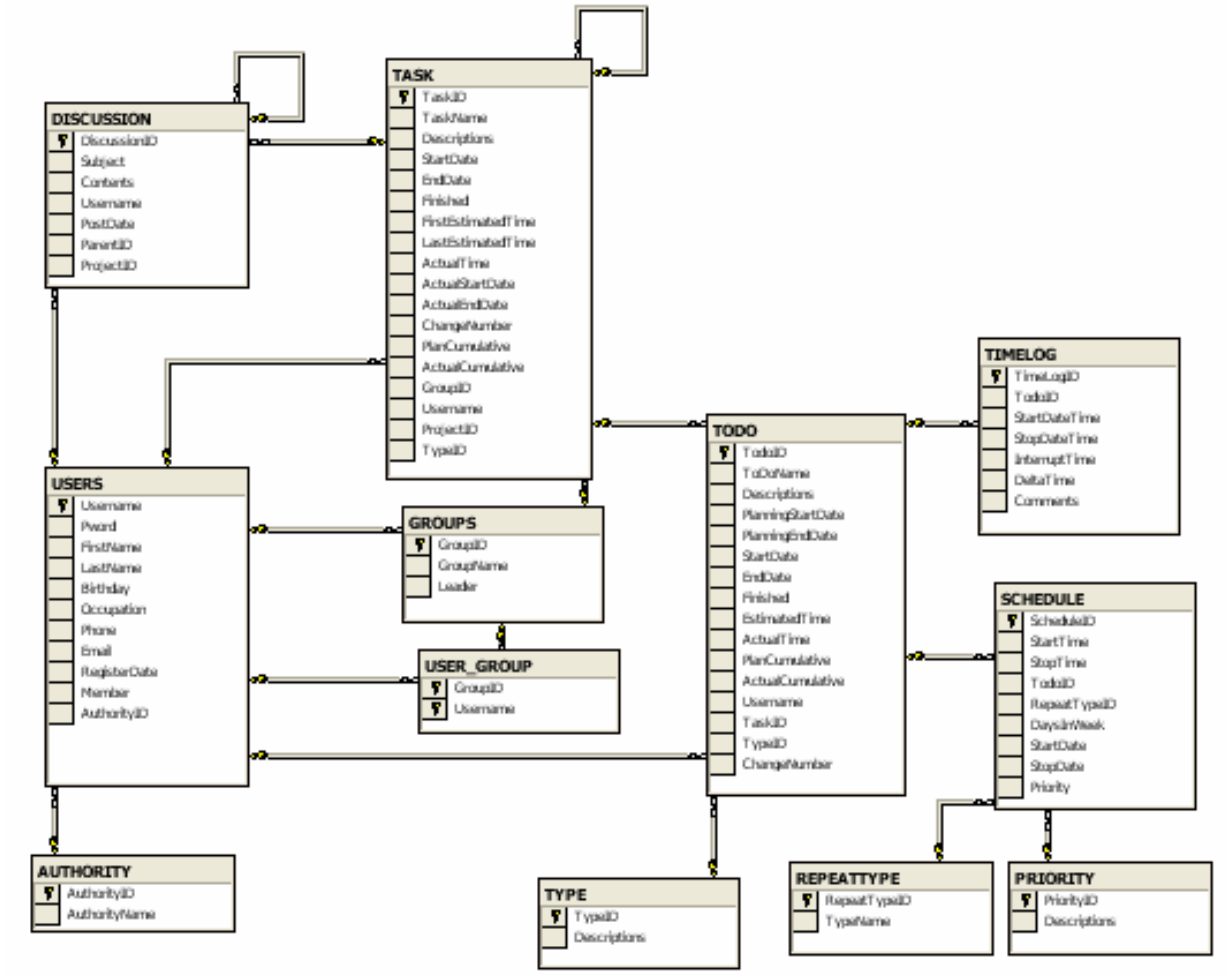


Hình 5.4.10 Sequence Diagram - Tìm kiếm thông tin dự án

5.4.4.7 Các sequence diagram khác:

Xem tập tin đính kèm: baocaochinh\diagrams.mdl

5.4.5 Mô hình thực thể kết hợp



Hình 5.4.11Mô hình thực thể kết hợp của ứng dụng

Chương 6. Một số kết luận và hướng phát triển

6.1 Kết quả đạt được:

6.1.1 Về mặt lý thuyết

- Tài liệu tổng hợp, phân tích PSP theo hai hướng tiếp cận: tiếp cận phương pháp luận và tiếp cận thực hiện qui trình.
- Trình bày và phân tích chi tiết hai phương pháp luận chính sử dụng trong PSP:
  - Quản lý thời gian và kế hoạch – quy trình lên kế hoạch
  - Quản lý chất lượng sản phẩm – quy trình quản lý sai sót
- Trình bày và hướng dẫn cách áp dụng PSP cho từng cá nhân.
- Nêu ra một số kết quả của việc áp dụng PSP vào thực tế.

6.1.2 Về mặt ứng dụng

Xây dựng ứng dụng web sử dụng nguyên lý quản lý thời gian và kế hoạch của PSP để hỗ trợ:

- Các nhóm phát triển phần mềm lập kế hoạch thời gian cho dự án.
- Các cá nhân theo dõi tiến độ thực hiện công việc.
- Cá nhân đánh giá độ chính xác việc ước lượng của bản thân.

6.2 Hướng phát triển

- Phát triển thêm các chức năng quản lý sai sót.
- Tăng thêm tính tiện dụng cho người dùng.
- Xây dựng công cụ ghi nhận thời gian trên Desktop, máy Palm, PDA và các cập nhật thông tin này vào ứng dụng Web bằng cách đồng bộ các thông tin về dự án theo format XML.

Tài liệu tham khảo

[1] Watts S. Humphrey, *The Personal Software Process*^{SM} *(PSP*^{SM}*),* Carnegie Mellon – Software Engineering Institute, 11/2000, pp. 17

[2] Menegos – Stavros, *Process Improvement Experiment Final Report,* Computer Logic S.A, 28/9/1998

[3] Stephen M. Shook, *Personal Software Process*^{SM} *& Team Software Process*^{SM}, Illinois State University, 03/11/2004

[4] Watts S. Humphrey, *Introduction to the Personal Software Process*^{SM}*,* Addison Wesley Longman, Inc., 1999

[5] Pat Ferguson – Watts S. Humphrey – Soheil Khajenoori – Susan Macke, *Results of Applying the Personal Software Process*

[6] Gina C. Green - Alan R. Hevner, The Successful Diffusion of Innovations: Guidance for Software Development Organizations

[7] John C. Donoghue, Personal Software Process (PSP)^{sm}, 14/12/2000

[8] Iraj Hirmanpour – Soheil Khajenoori, *Personal Software Process Technology: An Experiential Report,* Department of Computing and Mathematics – Embry-Riddle Aeronautical University

[9] Mike Grasso, Ph.D, *The Personal Software Process*, University of Maryland Baltimore County

[10] Elina Koivumaki Heli Rasilainen, Personal Software Process (PSP) and quality

[11] Will Hayes & James W.Over, The Personal Software Process^{SM}: An Empirical Study of the Impact of PSP on Individual Engineers, Carnegie Mellon University, 12/1997

[12] Watt S.Humphrey, *The personal Software Process*, Carnegie Mellon University, 12/2000

[13] Watt S.Humphrey, *Tutorial using PSP0*, Carnegie Mellon University, 2/2005

[14] Watt S.Humphrey, *Tutorial using PSP0.1*, Carnegie Mellon University, 2/2005

[15] Watt S.Humphrey, *Tutorial using PSP1*, Carnegie Mellon University, 2/2005

[16] Watt S.Humphrey, *Tutorial using PSP1.1*, Carnegie Mellon University, 2/2005

[17] Watt S.Humphrey, *Tutorial using PSP2*, Carnegie Mellon University, 2/2005

[18] Watt S.Humphrey, *Tutorial using PSP2.1*, Carnegie Mellon University, 2/2005