BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

LUẬN VĂN THẠC SỸ KHOA HỌC

ỨNG DỤNG LẬP TRÌNH LINH HOẠT TRONG QUY TRÌNH CỘNG TÁC PHẦN MỀM

NGÀNH: CÔNG NGHỆ THÔNG TIN
MÃ SỐ:

AN VĂN MINH

Người hướng dẫn khoa học: TS. HUỲNH QUYẾT THẮNG

HÀ NỘI - 2006

LỜI CAM ĐOAN

Em xin cam đoan luận văn tốt nghiệp này là kết quả nghiên cứu của bản thân, dưới sự hướng dẫn của thầy giáo, TS.Huỳnh Quyết Thắng. Nếu có gì sai phạm em xin hoàn toàn chịu trách nhiệm.

Người làm cam đoan

An Văn Minh

MỤC LỤC

DANH SÁCH BẢNG

DANH SÁCH CÁC HÌNH VẼ

LỜI CẢM ƠN

Em xin gửi lời cảm ơn sâu sắc tới thầy giáo, TS.Huỳnh Quyết Thắng, đã hướng dẫn, chỉ bảo và giúp đỡ em hết sức tận tình, để em hoàn thành tốt luận văn này. Em xin gửi lời cảm ơn chân thành đến các thầy cô giáo trong khoa Công nghệ thông tin, trường Đại học Bách khoa Hà Nội đã giảng dạy, tạo điều kiện và giúp đỡ em trong suốt quá trình học tập tại trường.

Chân thành cảm ơn các anh, chị và các bạn học viên lớp CNTT-2004, đã động viên và giúp đỡ em rất nhiều trong thời gian học tập và làm luận văn tốt nghiệp, để em có được kết quả tốt.

Em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

Xử lý thông tin là một nhu cầu tất yếu của con người, hoạt động này diễn ra hằng ngày, hằng giờ và trong tất cả các lĩnh vực của đời sống xã hội. Với sự ra đời của máy tính điện tử, một chiếc máy, xử lý thông tin một cách tự động và nhanh chóng. Nó đã giúp con người tiết kiệm được rất nhiều thời gian và công sức trong việc xử lý thông tin. Nhưng ta cũng biết rằng, để máy tính thực hiện xử lý thông tin, người sử dụng phải đưa vào đó một chương trình để điều khiển, và được gọi là phần mềm.

Với sự phát triển mạnh mẽ của nền công nghiệp nói chung, và công nghệ máy tính nói riêng, ngày càng có nhiều tổ chức sử dụng máy tính vào việc xử lý thông tin, nhằm giảm bớt nhân lực, và sự nhàm chán trong công việc. Nhưng ta cũng biết rằng, khối lượng thông tin ngày càng lớn, các thao tác xử lý ngày càng phức tạp. Do vậy, việc xây dựng phần mềm máy tính cũng trở nên rất khó khăn và đòi hỏi phải tuân theo một quy trình làm việc thích hợp. Công nghệ phần mềm ra đời, đã đưa ra các quy trình, giúp cho việc xây dựng phần mềm được thuận lợi, chẳng hạn, quy trình phần mềm dựa trên các cá nhân (PSP).

Tuy nhiên, với hiểu biết ngày càng sâu sắc hơn về công nghệ thông tin. Con người, mà cụ thể là các khách hàng phần mềm, không dừng lại ở nhu cầu cần có một phần mềm máy tính, mà họ còn muốn có nó một cách nhanh chóng. Hơn nữa, phần mềm phải có kích thước vừa phải, các thao tác xử lý nhanh, chính xác, đáp ứng yêu cầu của bài toán, đồng thời phải dễ sửa đổi và nâng cấp. Ngoài ra, họ còn muốn dõi theo quá trình xây dựng phần mềm, để chắc chắn rằng, phần mềm của họ được xây dựng đúng tiến độ, và đạt được hiệu quả mong muốn.

Việc xây dựng một phần mềm theo PSP là khá xa rời khách hàng. Tổ chức phần mềm nhận yêu cầu xây dựng phần mềm, sau một thời gian, giao

phần mềm cho khách hàng. Khách hàng chẳng biết gì về quá trình xây dựng phần mềm và họ không thể tin chắc rằng, phần mềm có thể được xây dựng thành công hay không?. Hơn nữa, việc sử dụng PSP, một tổ chức xây dựng phần mềm giao các nhiệm vụ cần thực hiện cho từng cá nhân. Vì vậy, phần mềm thường có nhiều lỗi, các thao tác xử lý chậm, thiếu chính xác…

Để khắc phục các nhược điểm nói trên, cần có một quy trình làm phần mềm mới, và phương pháp XP ra đời. Với mục tiêu là giao nhanh phần mềm đến tay khách hàng, đồng thời khách hàng có thể dõi theo quá trình xây dựng phần mềm, và tin tưởng vào khả năng phần mềm sẽ được hoàn thiện và có hiệu quả tốt.

Với mong muốn được đóng góp một phần nhỏ bé vào xu thế phát triển ngành công nghệ thông tin, đặc biệt trong giáo dục-đào tạo, cũng như trong việc xây dựng phần mềm ứng dụng, đáp ứng yêu cầu xử lý thông tin ngày càng cao của con người. Luận văn tốt nghiệp đã nghiên cứu đề tài: "ỨNG DỤNG LẬP TRÌNH LINH HOẠT TRONG QUY TRÌNH CỘNG TÁC PHẦN MỀM". NVLV mong rằng nó sẽ góp phần vào việc nâng cao chất lượng đào tạo sinh viên ngành công nghệ thông tin, và giúp các tổ chức phần mềm biết thêm về một quy trình xây dựng phần mềm, khắc phục những nhược điểm của các quy trình cũ. Đáp ứng được những đòi hỏi ngày càng khắt khe của những khách hàng phần mềm.

Luận văn gồm 4 chương:

- *Chương 1:Tổng quan về "Lập trình linh hoạt" và "Quy trình cộng tác phần mềm*", nghiên cứu các khái niệm trong phương pháp XP và quy trình CSP.

- *Chương 2: Các "thông lệ" trong Lập trình linh hoạt*, nghiên cứu các "thông lệ" trong XP, đây là các quy tắc và các bước thực hiện mà người lập trình cần tuân thủ khi xây dựng phần mềm dựa trên XP.

- *Chương 3: Ứng dụng "Lập trình linh hoạt" trong "Quy trình cộng tác phần mềm",* đề xuất một quy trình ứng dụng "Lập trình linh hoạt" trong "Quy trình cộng tác phần mềm", và các bước cần thực hiện để xây dựng phần mềm theo quy trình này.

- *Chương 4: Ứng dụng "Lập trình linh hoạt" trong đào tạo và phát triển phần mềm*, trình bày các thử nghiệm áp dụng phương pháp XP vào giảng dạy một môn học lập trình, ứng dụng "Lập trình linh hoạt" trong "Quy trình cộng tác phần mềm" để phát triển phần mềm "Quản lý nhân sự".

Với khoảng thời gian nghiên cứu đề tài không nhiều và trình độ kiến thức về công nghệ phần mềm còn hạn chế, nên luận văn không thể tránh khỏi những sai sót. Em rất mong được sự đánh giá và góp ý bổ sung của các thầy giáo, cô giáo và các bạn để luận văn được hoàn thiện hơn.

Hà nội, ngày tháng năm 2006

Học viên thực hiện

An Văn Minh

Chương 1. TỔNG QUAN VỀ LẬP TRÌNH "LINH HOẠT" VÀ "QUY TRÌNH CỘNG TÁC PHẦN MỀM"

1.1. PHƯƠNG PHÁP LẬP TRÌNH LINH HOẠT

1.1.1. Lập trình "linh hoạt" là gì?

Lập trình "linh hoạt" (XP) là một tập các giá trị, các quy tắc và các bước thực hiện, để phát triển nhanh một phần mềm chất lượng cao. Đây là một phương pháp phát triển phần mềm rất linh hoạt, nó phù hợp để phát triển các ứng dụng có kích thước vừa phải [5]. Một điểm đặc biệt của XP là trong quá trình phát triển phần mềm, khách hàng tham gia cùng với nhà phát triển. Nhờ đó, nhà phát triển nắm bắt được các thay đổi, các yêu cầu mới, làm giảm chi phí để sửa đổi hệ thống.

XP đạt được thành công bởi nó làm cho khách hàng thoả mãn về tâm lý. Các tổ chức phần mềm sử dụng XP, để phát triển nhanh một phần mềm và giao cho khách hàng khi họ yêu cầu. XP giúp các nhà phát triển đáp ứng được việc thay đổi các yêu cầu của khách hàng, thậm chí là đến cuối chu kỳ phát triển hệ thống.

XP cải tiến một dự án phần mềm bằng 4 phương pháp chính [1]: trao đổi thông tin, tính đơn giản, phản hồi thông tin, và sẵn sàng đón nhận thay đổi yêu cầu. Các lập trình viên XP trao đổi thông tin với khách hàng và với các lập trình viên trong cùng nhóm. Họ tạo ra thiết kế ở mức đơn giản và rõ ràng. Họ nhận thông tin phản hồi bằng cách kiểm tra phần mềm của họ vào thời gian đầu tiên của mỗi ngày làm việc. Họ giao phần mềm cho khách hàng trong thời gian ngắn và thực hiện các thay đổi nếu được đề nghị. Như vậy, các lập trình viên XP có thể đáp ứng được việc thay đổi các yêu cầu từ phía khách hàng và những thay đổi về công nghệ.

XP khác với các phương pháp khác. Nó gồm nhiều thành phần nhỏ, các thành phần riêng biệt không tạo ra được kịch bản, nhưng khi kết hợp chúng lại với nhau ta có thể nhìn thấy một "bức tranh hoàn thiện". Đây là một đặc điểm khác với các phương pháp phát triển phần mềm truyền thống và dẫn đến một thay đổi trong cách lập trình [1].

1.1.2. Tại sao sử dụng XP?

Như đã biết, hầu hết các phương pháp phát triển phần mềm truyền thống đều bao gồm các bước: Nắm bắt các yêu cầu, phân tích, thiết kế, viết mã lệnh, kiểm thử và bảo trì.

Cách tiếp cận này có thuận lợi đó là xác định được sản phẩm cuối cùng trước khi tiến trình xây dựng phần mềm được thực hiện. Tuy nhiên, điều này không phù hợp với các dự án phần mềm hiện đại với các yêu cầu luôn luôn thay đổi. Khách hàng sẽ luôn đưa ra nhiều yêu cầu mới và cần những thông tin phản hồi liên tục để điều chỉnh lại các lựa chọn của họ. Người lập trình cần phải có một phương pháp thực hiện nào đó, để luôn sẵn sàng đón nhận những thay đổi từ người dùng để họ có thể tiếp nhận được các thông tin phản hồi. Nếu bạn làm việc trong một môi trường mà ở đó hệ thống cần phải đáp ứng được các yêu cầu luôn thay đổi và khách hàng sẽ có lợi nếu việc bàn giao phần mềm sớm hơn và thường xuyên thì nên xem xét việc sử dụng XP để phát triển hệ thống. Các nhóm lập trình sử dụng XP nhận thấy rằng, họ sẽ bàn giao các sản phẩm phần mềm chất lượng cao với số lượng rất lớn và nhanh hơn trước đây rất nhiều [1].

1.1.3. Lịch sử phát triển của XP

XP được tạo ra bởi Kent Beck, Ward Cunningham và Ron Jeffries khi họ làm việc cho dự án phát triển hệ thống Chrysler Comprehensive Compensation (C3) [34]. Kent Beck trở thành chủ dự án C3 vào tháng 3 năm 1996 và bắt đầu lựa chọn một phương pháp phát triển để sử dụng vào việc

phát triển dự án. Kent Beck viết một cuốn sách về phương pháp đã sử dụng và vào tháng 10 năm 1999, cuốn sách *Extreme Programming Explained [34]* được xuất bản. Chrysler huỷ bỏ dự án C3 vào năm 2000, nhưng phương pháp thực hiện dự án được ghi nhận trong lĩnh vực công nghệ phần mềm. Đến năm 2006, một số dự án phát triển phần mềm vẫn tiếp tục sử dụng XP.

**** Nguồn gốc của XP***

Việc phát triển phần mềm trong những năm 1990 có hai ảnh hưởng chính: thứ nhất, lập trình hướng đối tượng (OOP) thay thế lập trình thủ tục khi OOP được sự quan tâm ủng hộ của ngành công nghiệp; thứ hai là sự xuất hiện của Internet và các công ty .COM nhấn mạnh thời gian đưa sản phẩm ra thị trường và sự lớn mạnh của công ty là các yếu tố cạnh tranh thương mại. Các yêu cầu thay đổi nhanh chóng làm ngắn lại vòng đời sản phẩm và thường không phù hợp với các phương pháp phát triển phần mềm truyền thống.

Dự án C3 được thực hiện nhằm xác định cách thức tốt nhất để sử dụng công nghệ đối tượng. Dựa trên việc nghiên cứu các hệ thống thanh toán tại Chrysler, sử dụng ngôn ngữ Smalltalk. Dự án của Kent Beck được thực hiện cùng với một lập trình viên giỏi về Smalltalk, để điều khiển quá trình thực hiện của hệ thống và ông phát hiện ra nhiều vấn đề nảy sinh trong quá trình phát triển. Từ đó, đề xuất và thực hiện một số thay đổi trong các cách thực hiện của mình, cùng với một cộng tác viên là Ward Cunningham.

1.1.4. Các mục tiêu của XP

XP là một phương pháp phát triển phần mềm, giúp các tổ chức phần mềm đạt được các mục tiêu [1]:

- Làm thoả mãn nhu cầu của người dùng.
- Đáp ứng những thay đổi mang tính xã hội.
- Đưa ra cách thức cải tiến phần mềm.
- Xác định kiểu phát triển phần mềm.

- Xác định các bước cần thực hiện để phát triển phần mềm.

Ưu điểm chính của XP là làm giảm chi phí cho việc thay đổi các yêu cầu. Trong các phương pháp truyền thống, các yêu cầu của hệ thống được xác định ngay từ khi bắt đầu phát triển dự án và thường được cố định từ thời điểm đó, làm cho việc bảo trì hệ thống rất khó khăn và chí phí lớn.

Việc sử dụng XP có thể giảm được chi phí cho việc thay đổi là nhờ vào các giá trị, các quy tắc và các bước thực hiện. Bằng cách áp dụng XP, một dự án phát triển hệ thống sẽ mềm dẻo hơn trong việc sửa đổi hoặc bổ sung yêu cầu [34].

1.1.5. Các giá trị của XP

XP gồm có 5 giá trị:

- Trao đổi thông tin.
- Tính đơn giản.
- Phản hồi thông tin.
- Phát triển phần mềm theo quan điểm của người dùng.
- Sự quan tâm (respect).

Để phát triển một hệ thống phần mềm, các lập trình viên phải hiểu được những yêu cầu của hệ thống. Trong các phương pháp truyền thống, việc này được thực hiện bằng cách làm tài liệu. Tuy nhiên, XP lại sử dụng các giá trị của nó để phổ biến một cách nhanh chóng các kiến thức đến các thành viên trong một nhóm phát triển. Mục đích là tạo cho tất cả những người phát triển một quan điểm chung về hệ thống, để nó phù hợp với quan điểm của những người sử dụng hệ thống. XP ủng hộ việc tạo ra các thiết kế đơn giản, thông suốt, sự cộng tác của người sử dụng với lập trình viên, việc trao đổi thông tin bằng lời, và việc phản hồi thông tin.

XP khuyến khích bắt đầu với giải pháp đơn giản nhất để cho mọi lập trình viên đều hiểu được bài toán. Sự khác nhau giữa quan điểm này với các

phương pháp truyền thống là tập trung vào việc thiết kế và viết mã lệnh cho các yêu cầu hôm nay thay vì làm việc đó cho ngay mai, tuần sau hoặc tháng sau. Những người đề xuất XP chấp nhận điều không thuận lợi là có thể việc cố gắng thay đổi hệ thống là thừa; có những hệ thống không cần sự thay đổi sau khi đã xây dựng. Từ đó không cần phải tính đến chi phí cho việc sửa đổi hệ thống. Nếu tính đến giá trị trao đổi thông tin, thì sự đơn giản trong thiết kế và mã lệnh sẽ cải thiện được "chất lượng" của việc trao đổi. Một thiết kế đơn giản cùng với mã lệnh rất đơn giản, làm cho các lập trình viên trong nhóm dễ hiểu hơn.

Trong XP, thông tin phản hồi có liên quan đến nhiều vấn đề khác nhau của việc phát triển hệ thống:

- Thông tin phản hồi từ hệ thống: bằng cách viết các bộ kiểm tra, hoặc chạy các bộ kiểm tra để kiểm tra việc kết hợp theo định kỳ, các lập trình viên có được các thông tin phản hồi trực tiếp từ trạng thái của hệ thống, sau khi thực hiện các thay đổi.

- Thông tin phản hồi từ người dùng: Các bộ kiểm thử chức năng được viết bởi người dùng và những người thực hiện kiểm thử. Họ sẽ nhận được các thông tin cụ thể về trạng thái hiện tại của hệ thống. Việc kiểm thử này chỉ cần chuẩn bị một lần cho cả hai hay ba tuần, vì vậy, người dùng có thể dễ dàng hướng theo việc phát triển.

- Thông tin phản hồi từ nhóm: khi người dùng đưa ra các yêu cầu mới, nhóm phát triển tiếp nhận yêu cầu và đưa ra một đánh giá trực tiếp về thời gian cần thiết để thực hiện yêu cầu.

Thông tin phản hồi có liên quan mật thiết với việc trao đổi thông tin và tính đơn giản. Các lỗi trong hệ thống có thể dễ dàng được phát hiện bằng cách viết một bộ kiểm tra, để tìm ra đoạn mã lệnh có lỗi. Thông tin phản hồi trực tiếp từ hệ thống cho phép các lập trình viên biết, để ghi nhận lỗi của đoạn mã

lệnh đó. Người dùng cũng có thể kiểm tra hệ thống định kỳ theo các yêu cầu chức năng. Kent Beck nói: "Sự lạc quan là một nguy cơ gây ra lỗi trong lập trình và thông tin phản hồi giúp xử lý việc này".

"Sự quan tâm" có thể biểu thị theo nhiều cách. Thứ nhất, trong XP, các thành viên trong nhóm nên quan tâm lẫn nhau, bởi các lập trình viên không nên uỷ thác các thay đổi dẫn đến làm hỏng việc biên dịch, các bộ kiểm tra đang tồn tại bị hỏng, hoặc làm trễ việc của người cùng nhóm. Thứ hai, các thành viên quan tâm đến công việc của họ bằng cách luôn phấn đấu vì chất lượng cao và tìm ra các thiết kế tốt nhất cho giải pháp ngay khi cải tiến mã lệnh.

1.1.6. Các quy tắc của XP

Các quy tắc của XP định dạng cơ sở của XP được dựa trên các giá trị vừa được mô tả và dự kiến để thúc đẩy việc đưa ra các quyết định trong dự án phát triển hệ thống. Các quy tắc được dự kiến là cụ thể hơn so với các giá trị và dễ dàng biên dịch hơn để đưa ra hướng dẫn trong tình huống đặc biệt.

1.1.6.1. Phản hồi thông tin

Sự phản hồi thông tin có tác dụng nhất nếu nó được thực hiện một cách nhanh chóng. Thời gian giữa một kích hoạt và thông tin phản hồi nhận được là vấn đề then chốt để nghiên cứu và thực hiện các thay đổi. Trong XP, ngoại trừ các phương pháp phát triển hệ thống cũ, việc tiếp xúc với người dùng chỉ trong những khoảng thời gian ngắn. Người dùng hiểu biết rõ ràng về hệ thống đang được phát triển và có thể đưa ra thông tin phản hồi cũng như hướng theo sự phát triển nếu cần thiết.

Các bộ kiểm tra (Unit tests) cũng làm cho sự phản hồi thông tin được thực hiện nhanh hơn. Khi viết mã lệnh, bộ kiểm tra cung cấp thông tin phản hồi trực tiếp để sao cho hệ thống tác động trở lại các thay đổi. Nếu trong trường hợp khẩn cấp, các thay đổi ảnh hưởng đến một phần hệ thống mà

không nằm trong phạm vị của lập trình viên đã tạo ra thành phần đó, lập trình viên đó không cần chú ý đến điều này. Khả năng xuất hiện lỗi này là rất lớn khi hệ thống đang trong thời gian xây dựng.

1.1.6.2. Tính đơn giản

Tính đơn giản là việc xem xét mọi vấn đề nếu như giải pháp của nó là "hết sức đơn giản". Các phương pháp phát triển hệ thống cũ nói đến việc lập kế hoạch cho tương lai và mã hoá cho khả năng sử dụng lại. XP không thực hiện theo ý tưởng này.

Những người ủng hộ XP nói rằng, việc tạo ra tất cả các thay đổi lớn trong một lần là không thể thực hiện được. XP áp dụng các thay đổi mang tính cải tiến: chẳng hạn, cứ sau 3 tuần một hệ thống có thể có những bước hoàn thiện nhỏ. Bằng cách tạo ra nhiều bước hoàn thiện nhỏ, người sử dụng thể tác động nhiều hơn đối với quá trình phát triển và đối với hệ thống đang được phát triển.

1.1.6.3. Đón nhận sự thay đổi

Quy tắc đón nhận sự thay đổi là không chống lại các thay đổi và phải nắm bắt được chúng. Trong trường hợp cần thiết, nếu có yêu cầu thay đổi nào đó đến từ phía người dùng, các lập trình viên sẵn sàng đón nhận yêu cầu đó đồng thời lập kế hoạch cho các yêu cầu sắp tới.

1.1.7. Các hoạt động theo XP

XP mô tả 4 hoạt động chủ yếu được thực hiện trong quá trình phát triển phần mềm [34] gồm: Viết mã lệnh, kiểm thử, nhận định các tác nhân hệ thống và thiết kế.

1.1.7.1. Viết mã lệnh

Những người ủng hộ XP cho rằng, sản phẩm thực sự quan trọng của quá trình phát triển hệ thống là mã lệnh. Không có mã lệnh bạn chẳng có gì hết.

Việc viết mã lệnh có thể là việc vẽ các sơ đồ từ đó sẽ phát sinh mã lệnh, việc viết kịch bản cho hệ thống dựa trên web hoặc viết lệnh cho chương trình cần thiết phải được biên dịch.

Việc viết mã lệnh cũng có thể được dùng để chỉ ra giải pháp phù hợp nhất cho bài toán. Trong trường hợp này, XP tán thành việc đối mặt với nhiều lựa chọn cho một vấn đề lập trình, một lựa chọn đơn giản cho việc viết mã lệnh cho tất cả các giải pháp và xác định với các bộ kiểm tra tự động cho giải pháp phù hợp nhất.

Việc viết mã lệnh cũng giúp trao đổi những suy nghĩ về các vấn đề lập trình. Một lập trình viên thực hiện một bài toán lập trình phức tạp và thấy rằng khó có thể giải thích giải pháp đó cho các lập trình viên khác, có thể viết mã lệnh cho nó và sử dụng mã lệnh để giải thích ý định của mình. Các lập trình viên khác có thể giải thích vấn đề này bằng cách viết mã lệnh theo suy nghĩ của họ.

1.1.7.2. Kiểm thử

Một lập trình viên không thể chắc chắn về bất kỳ điều gì trừ khi anh ta đã kiểm tra nó. Việc kiểm thử không phải là để nhận biết được vấn đề mà là yêu cầu về tính chính xác của người dùng. Trong phát triển phần mềm, XP cho rằng điều này có nghĩa là một người không thể chắc chắn rằng một chức năng sẽ hoạt động trừ khi anh ta đã kiểm tra nó. Từ đó, đặt ra câu hỏi là cần phải xác định điều gì mà anh ta chưa chắc chắn về nó.

- Các lập trình viên có thể không chắc chắn về mã lệnh mà họ đã viết so với mã lệnh mà họ nên viết. Để kiểm tra điều này, XP sử dụng các bộ kiểm tra (Unit Tests). Có các bộ kiểm tra, thực hiện việc kiểm tra mã lệnh một cách tự động. Các lập trình viên sẽ cố gắng viết mã lệnh, nhưng họ có thể nghĩ rằng các bộ kiểm tra có thể làm hỏng mã lệnh mà họ đang viết; tuy nhiên, nếu tất

cả các bộ kiểm tra đều thực hiện thành công thì việc viết mã lệnh là hoàn thiện.

- Các lập trình viên có thể không chắc chắn về điều mà họ đã làm so với điều mà đáng ra họ nên làm. Để kiểm tra việc này, XP sử dụng các bộ kiểm tra chuyên biệt (acceptance tests) dựa trên các yêu cầu được đưa ra bởi khách hàng trong giai đoạn tìm hiểu việc lập kế hoạch theo từng bước (release planning).

1.1.7.3. Nhận định các tác nhân của hệ thống

Các lập trình viên không cần biết bất kỳ điều gì về mặt nghiệp vụ của hệ thống trong quá trình phát triển. Nhưng chức năng của hệ thống lại được xác định từ nghiệp vụ của hệ thống. Để các lập trình viên tìm ra được những gì nên thuộc về chức năng của hệ thống, họ cần phải tìm hiểu nghiệp vụ của hệ thống.

Các lập trình viên phải tìm hiểu trong phạm vi rộng: họ phải tìm hiểu những gì mà người dùng yêu cầu. Họ phải cố hiểu vấn đề nghiệp vụ của hệ thống, và gửi thông tin phản hồi cho người dùng về vấn đề của anh ta, để cải thiện sự hiểu biết của anh ta về bài toán.

Việc trao đổi giữa khách hàng và lập trình viên được trình bày trong phần “Planning game”.

1.1.7.4. Thiết kế

Từ quan điểm về tính đơn giản, một người có thể nói rằng việc phát triển hệ thống không cần thêm điều gì ngoài việc viết mã lệnh, kiểm tra và tìm hiểu các yếu tố tác động đến hệ thống. Nếu các hoạt động này được thực hiện tốt, kết quả sẽ luôn là một hệ thống làm việc được. Trong thực tế, hệ thống này sẽ không làm việc. Một người có thể thực hiện công việc theo một cách nào đó mà không cần thiết kế, nhưng cuối cùng sẽ bị mắc kẹt. hệ thống sẽ trở nên quá phức tạp, và những ràng buộc trong hệ thống sẽ ngày càng rõ ràng.

Có thể tránh điều này bằng cách tạo ra một cấu trúc thiết kế được tổ chức một cách logic bên trong hệ thống. Một thiết kế tốt sẽ tránh được nhiều phục thuộc trong hệ thống; điều này có nghĩa là việc thay đổi một phần hệ thống không ảnh hưởng đến những phần khác của hệ thống.

1.2. QUY TRÌNH CỘNG TÁC PHẦN MỀM

Trong công nghiệp người ta đã chứng minh rằng hai lập trình viên làm việc bên cạnh nhau, trên cùng một máy tính, cùng thiết kế thuật toán, cùng viết mã lệnh, hay cùng kiểm tra chương trình, về căn bản, hiệu quả hơn so với hai người làm việc độc lập[35]. Điều này cho thấy rằng, các lập trình viên sẽ làm việc tốt hơn khi cùng thực hiện một quá trình đã được xác định, người này xem xét lại công việc của người kia. Từ việc đưa hai ý tưởng đến cùng giải quyết một công việc, người ta tạo ra quy trình cộng tác phần mềm (CSP). CSP là một quy trình phát triển phần mềm được xác định, có tính chất lặp lại, với hai lập trình viên cộng tác làm việc. CSP là sự mở rộng của quá trình phần mềm độc lập (PSP) và nó dựa trên nền tảng của PSP.

Để chứng minh hiệu quả của CSP, một thí nghiệm được tiến hành năm 1999, với gần 40 sinh viên cuối khoá, ngành khoa học máy tính tại đại học Utah. Tất cả các sinh viên đều được học CSP và PSP [35]. Có 3 nhóm, mỗi nhóm hai sinh viên cùng làm việc trên một máy tính để phát triển các nhiệm vụ lập trình được giao. Các sinh viên khác làm việc độc lập và cũng thực hiện những nhiệm vụ tương tự. Nghiên cứu đã đóng góp vào việc tạo ra một quá trình làm phần mềm xác định, có tính chất lặp lại, quy trình cộng tác phần mềm, với các cặp lập trình viên cộng tác. Thí nghiệm đã đưa ra một số nhận định sau về các nhóm lập trình cộng tác, sử dụng CSP [35]:

1. Các cặp lập trình cộng tác mất thời gian nhiều hơn so với lập trình độc lập là 15%. Tuy nhiên, thời gian này là không đáng kể.

2. Sản phẩm do các cặp lập trình cộng tác tạo ra có chất lượng tốt hơn đáng kể. Đồng thời tiết kiệm được 15% thời gian mã hoá.

3. Nếu tính đến việc trợ giúp trong thời gian lâu dài, để có các sản phẩm lập trình chất lượng cao hơn thì lập trình cộng tác chi phí ít hơn lập trình độc lập.

4. 95% số người lập trình cộng tác khẳng định rằng họ thích và tin tưởng vào công việc hơn so với khi họ làm việc một mình.

Ngoài ra nghiên cứu còn đưa ra nhiều khẳng định về lập trình cộng tác. Đáng chú ý nhất là hiệu quả rõ ràng trong việc gia tăng các kỹ năng giải quyết bài toán, các thiết kế được làm tốt hơn, tăng khả năng học hỏi, và thúc đẩy việc tạo ra nhóm các cặp lập trình cộng tác. Các tổ chức mà ở đó các kỹ sư phần mềm luân chuyển một cách phù hợp các thành viên, cũng như chú ý làm tăng việc trao đổi thông tin, nâng cao khả năng làm việc nhóm từ đó làm giảm lỗi của sản phẩm.

1.2.1. Giới thiệu quá trình cộng tác phần mềm

1.2.1.1. Động lực nghiên cứu

Ngày nay trong tin học, hầu hết việc sản xuất phần mềm được thực hiện bởi các nhà khoa học, họ cố gắng giải quyết các bài toán cụ thể, kích thước vừa phải. Mô hình lập trình hiện nay được gọi là mô hình code-and-fix... đây là một mô hình không những được xây dựng một cách chính xác mà còn được điều khiển một cách cẩn thận [2].Ghezzi và cộng sự mô tả mô hình lập trình code-and-fix gồm 2 bước [35]:

Bước 1: Viết mã lệnh.

Bước 2: Cố định mã lệnh để loại bỏ các lỗi, nâng cao chức năng hiện tại hoặc thêm đặc trưng mới.

Theo thời gian, máy tính trở nên rẻ hơn và phổ biến hơn. Ngày càng có nhiều người sử dụng máy tính để giải quyết các bài toán lớn hơn, nhưng vẫn sử dụng và phát triển mô hình lập trình truyền thống.

Ngày nay mô hình code-and-fix vẫn được sử dụng, nó không thích hợp để thực hiện các bài toán phức tạp ứng với việc phát triển phần mềm có tỉ trọng lớn. Cách đây khoảng 40 năm thuật ngữ "khủng hoảng phần mềm" được đưa ra để mô tả sự bất lực của ngành công nghiệp phần mềm với việc cung cấp cho khách hàng những sản phẩm chất lượng cao đạt yêu cầu. Những dự án càng lớn thì hiệu quả nhìn chung càng kém hơn. Và có khoảng 3/4 trong số các hệ thống lớn "thất bại trong việc vận hành", không thực hiện được chức năng như mong đợi hoặc không vận hành được [3].

Đáng chú ý là sự tiến triển trong hơn 40 năm của tình trạng khủng hoảng phần mềm, của chính các nhà khoa học và các nhà phát triển phần mềm và nói chung là tất cả mọi người. Ngày nay, phần mềm đang thâm nhập vào hầu hết các lĩnh vực của cuộc sống, bao gồm các hệ thống hỗ trợ quyết định ảnh hưởng đến cả sức khoẻ và hạnh phúc của con người [4]. Tất nhiên, sự cố Y2K là yếu tố đầu tiên ảnh hưởng đến các vấn đề phần mềm.

Thật không may, sự tiến bộ trong các kỹ thuật phát triển phần mềm đã bị cản trở bởi sự tăng theo hàm mũ của quy mô và tính phức tạp của phần mềm. Thách thức này được khắc phục và đem lại các kỹ thuật được vận dụng thành công để giải quyết sự phức tạp tăng lên hàng ngày. Một mục đích khác của nghiên cứu này là nhằm khắc phục sự khủng hoảng phần mềm, giúp ngành công nghiệp phần mềm sản suất được những phần mềm có chất lượng cao.

1.2.1.2. Lập trình cặp (Pair-Programming)

Với các ứng dụng phần mềm lớn hơn và phức tạp hơn; các ứng dụng này được sử dụng trong rất nhiều tổ chức với các mục đích khác nhau. Có lẽ, tốt hơn hết các ứng dụng phức tạp nên được giải quyết bởi hai người tại một thời

điểm. Ý tưởng cặp lập trình, hai người lập trình làm việc cộng tác, cùng thiết kế, cùng đưa ra thuật toán, cùng viết mã lệnh, hoặc cùng kiểm tra, đã được đưa ra nhiều lần từ cách đây khoảng 10 năm. Thói quen lập trình cặp có nhiều lợi ích, trước hết với sự ra đời của phương pháp lập trình "linh hoạt" [5].

Hai người cộng tác ngồi cạnh nhau, tại một máy tính trong suốt quá trình phát triển phần mềm. Một người được xem là người điều khiển. Người này có quyền điều khiển chuột, bàn phím hoặc dụng cụ để viết, và tích cực trong việc tạo ra thiết kế, mã lệnh và kiểm thử. Người còn lại sẽ quan sát công việc của người điều khiển và phát hiện các sai sót trong công việc của họ. Theo thống kê [7] người ta thấy rằng, các cặp lập trình tạo ra mã lệnh có chất lượng cao hơn, nhanh hơn so với những người lập trình độc lập.

1.2.1.3. Cách tiếp cận

CSP đạt được yêu cầu về việc chấp nhận một phương pháp phát triển phần mềm đã được xây dựng hoàn thiện, kết hợp với lập trình cặp khi phát triển phần mềm. Một quá trình phát triển phần mềm mới, quá trình cộng tác phần mềm là một phương pháp được xác định, có tính chất lặp lại đối với hai kỹ sư cộng tác phần mềm để phát triển phần mềm.

Trong CSP, các bước trong mỗi giai đoạn của quá trình phát triển phần mềm từ giai đoạn phân tích đến giai đoạn kiểm thử được hướng dẫn bởi các kịch bản chi tiết, các biểu mẫu (templates) các khuôn dạng (forms). Các thông tin từ các biểu mẫu và các khuôn dạng được phân tích và dùng làm cơ sở để đánh giá hiệu quả của cặp cộng tác. Cặp cộng tác sử dụng các thông tin này nhằm cải thiện hiệu quả của quá trình cộng tác của chính họ.

Nghiên cứu cho thấy, các cặp cộng tác sử dụng CSP làm việc tốt hơn các kỹ sư phần mềm làm việc độc lập, sử dụng PSP [8]. Một số thước đo cụ thể, sử dụng để so sánh giữa PSP và CSP là thời gian quay vòng, năng lực sản xuất và chất lượng phần mềm. Để chứng minh tính đúng đắn của giả thuyết,

một thí nghiệm có tính cấu trúc được thực hiện tại đại học Utah với một lớp học thiết kế phần mềm mức cao. Trong thí nghiệm này, các cặp cộng tác và các sinh viên độc lập đã hoàn thành cùng một nhiệm vụ. Thước đo trên được sử dụng để so sánh quá trình thực hiện của tất cả các sinh viên.

1.2.2. Các yếu tố liên quan đến CSP

CSP chính thức được công bố sau khi điều tra những đóng góp và những thành công trong nhiều lĩnh vực của công nghệ phần mềm và khoa học tri thức.

1.2.2.1. Quá trình phần mềm độc lập (PSP)

Về mặt cấu trúc, ảnh hưởng lớn nhất đối với CSP là từ PSP [8], được đưa ra bởi Watts S.Humphrey. PSP định nghĩa một mô hình phát triển phần mềm, bao gồm các thao tác đã được định nghĩa hoặc các tiến trình con và các kỹ thuật phân tích và đánh giá để giúp các kỹ sư phần mềm hiểu được các kỹ năng của chính họ, nhằm cải thiện năng lực của cá nhân mình. Mỗi tiến trình con có một tập các kịch bản, thể hiện các bước cụ thể kèm theo và một tập các biểu mẫu hoặc các khuôn dạng để điền thông tin vào đó, đảm bảo tính đầy đủ và thu thập dữ liệu tạo thành các thông tin đánh giá cơ sở. Các thông tin đánh giá cơ sở này cho phép các lập trình viên đánh giá công việc của mình, phân vùng bài toán, thiết lập và thực hiện các mục tiêu. Ví dụ: các lập trình viên ghi các thông tin về tất cả các lỗi được loại bỏ trong các chương trình của mình. Họ có thể sử dụng các thông tin đã được tổng hợp về các lỗi mà họ gặp phải để tránh việc lặp lại lỗi đó. Thêm vào đó, họ có thể kiểm tra các xu hướng mà họ mắc lỗi trong một nghìn dòng lệnh và có thể nhận thấy khi tạo ra cải tiến thực sự.

PSP có nhiều lý luận vững chắc. Thứ nhất, một lỗi phần mềm có thể duy trì được lâu hơn trong một sản phẩm, để tìm ra và loại bỏ nó là rất tốn kém. Bởi vậy, việc xem xét lại thiết kế và mã lệnh đã được thực hiện trước đó để

loại bỏ lỗi một cách hiệu quả nhất. Thứ hai, việc ngăn chặn lỗi là hiệu quả hơn so với việc loại bỏ lỗi. Các thiết kế thận trọng đã được phát triển và các dữ liệu được lựa chọn để bổ sung cho đầu vào nhằm điều chỉnh các tiến trình phần mềm độc lập của chính họ, từ đó ngăn chặn các lỗi trong tương lai. Cuối cùng, PSP dừng lại ở khái niệm được đánh giá là tốt nhất và vì vậy đạt được các kết quả tốt nhất, từ việc ghi nhận lại các lỗi, người ta tạo ra một cơ sở dữ liệu tỉ lệ lỗi. Dữ liệu về lỗi các sản phẩm trước đó được giữ lại để sử dụng cùng với các thủ tục đánh giá cơ sở dùng cho việc đánh giá hiệu quả của phần mềm. Các tiến trình và các triết lý này làm việc cùng với nhau để tạo ra các kết quả tốt hơn.

Dữ liệu của viện công nghệ phần mềm về 104 kỹ sư phần mềm chỉ ra rằng [35], PSP đào tạo để làm giảm các lỗi trong đánh giá quy mô là 25.8% và các lỗi đánh giá thời gian là 40%. Số dòng lệnh viết trong một giờ tăng trung bình là 20.8% và chi phí thời gian phát triển của mỗi kỹ sư dành cho việc biên dịch giảm được 81.7%, thời gian kiểm thử giảm 43.3%, tổng các lỗi là 59.8% và kiểm tra lỗi là 73.2%[9].

PSP là một tiến trình được định rõ, có tính chất lặp lại đối với một kỹ sư phần mềm làm việc độc lập; CSP được định rõ, có tính chất lặp lại đối với hai lập trình viên làm việc cộng tác. CSP là một mở rộng của PSP và dựa vào nền tảng của PSP.

1.2.2.2. Lập trình linh hoạt

CSP cũng chịu ảnh hưởng lớn bởi những yếu tố thành công của phương pháp XP[5], được phát triển chủ yếu bởi hãng Smalltalk và cố vấn Kent Beck cùng với các đồng nghiệp là Ward Cunningham và Ron Jeffries. XP không có được bằng chứng định lượng giống như PSP để chứng minh tính hiệu quả của mình. Bằng chứng về sự thành công của XP mang tính giai đoạn, nhưng rất ấn tượng trong công nghệ phần mềm. Một ví dụ lớn nhất về thành tựu của XP là

hệ thống thanh toán Chrysler khá lớn được khai trương vào tháng 5 năm 1997. Hệ thống thực hiện việc trả lương hàng tháng cho khoảng 10000 nhân viên, có 2000 lớp và 30000 phương thức [10]. Một ví dụ khác là các lập trình viên tại công ty ô tô Ford mất 4 năm không thành công trong việc thử xây dựng hệ thống VCAPS (Vehicle Cost and Profit System) bằng việc sử dụng phương pháp thác nước truyền thống. Sau đó họ đã sao chép lại hệ thống đó và đã thành công với thời gian ít hơn một năm, bằng sử dụng phương pháp XP [11]. XP có những thế mạnh nhờ sử dụng lập trình cặp. Việc thiết kế và viết mã lệnh cho chương trình được thực hiện bởi hai người, việc mở rộng hệ thống và thậm chí việc tạo các biểu mẫu cũng được thực hiện với sự cộng tác của hai người. Làm việc cùng nhau, các lập trình viên thực hiện việc xem xét mã kế tiếp nhau. "Bạn có thể ngạc nhiên vì có nhiều lỗi do bạn tạo ra mà người bên cạnh bạn phát hiện được". Có lẽ triết lý cuối cùng của PSP là ngăn chặn các lỗi và loại bỏ chúng một cách hiệu quả.

Việc tập hợp các yêu cầu, việc xác định tài nguyên và các hoạt động thiết kế của XP cũng là điểm xuất phát cơ bản của hầu hết các phương pháp đã được thừa nhận, chẳng hạn PSP hoặc quá trình được thống nhất một cách hợp lý [12]. Các yêu cầu được viết trong các tài liệu yêu cầu người dùng, một bản đánh giá sơ bộ về tài nguyên yêu cầu được đánh dấu trên các thẻ, các tài nguyên này được phân chia cho các cặp lập trình và họ bắt đầu viết mã lệnh. Với việc không có các thủ tục hoặc các bản thiết kế có sẵn trong kế hoạch phát triển hoặc kiến trúc hệ thống, cặp lập trình xác định mã trong cơ sở mã cần thiết để thêm vào hoặc làm thay đổi nó mà không cần phải có sự cho phép của bất kỳ người nào. Việc này đòi hỏi sử dụng "quyền sở hữu chung mã lệnh" do đó bất kỳ cặp lập trình nào cũng có thể sửa hoặc thêm vào mã bất kỳ trong cơ sở mã lệnh, kể cả những người mới biết lập trình cơ bản.

Các cặp lập trình làm mới cơ sở mã lệnh bằng cách tiếp tục thay đổi và bổ sung. Họ xem xét mã lệnh khi tự phát triển thiết kế - họ không mất thời gian làm tài liệu thiết kế, vì vậy, có được tài liệu về kiểu mã và các nguyên tắc giải thích một chặt chẽ. XP cũng có các thủ tục kiểm tra rất đặc biệt. Bao gồm các công cụ kiểm tra được viết và được làm tự động trước cho đến các thay đổi mã thực sự. Các kết quả của việc thực hiện các thủ tục kiểm tra này lại tự động tạo ra các kiểm tra và thứ tự ưu tiên mới, các công cụ kiểm tra ngược xác định xem có sự thay đổi/nâng cao hay không để thực hiện theo "tài liệu yêu cầu người dùng" một cách đúng đắn mà không tổn hại đến các hoạt động phát triển truyền thống, mang tính giai đoạn, sự xuất hiện của XP là rất hiệu quả. Thêm vào đó các lập trình viên cũng nói rằng, việc phát triển với các hoạt động của XP cũng rất thú vị và thú vị hơn so với các tiến trình truyền thống. Từ XP, CSP kết hợp thành công việc sử dụng lập trình cặp và việc phát sinh và thực hiện công cụ kiểm tra tự động.

1.2.2.3. Tri thức phân tán

Năm 1991, Nick Flor một học giả về khoa học tri thức, đã nghiên cứu về tri thức phân tán trong cặp lập trình cộng tác. Trong nghiên cứu của mình, ông ghi lại những trao đổi của hai lập trình viên làm việc cùng với nhau trên một nhiệm vụ bảo trì phần mềm. Ông ghi lại cả hình ảnh và âm thanh và đặt riêng các hành vi có lời thoại và không có lời thoại trong nghiên cứu về hai người dựa theo lý thuyết về tri thức phân tán và ông nhận thấy [35]:

1. Việc chia sẻ các mục tiêu và các kế hoạch: Các lập trình viên cộng tác cố gắng duy trì một tập các mục tiêu và các kế hoạch chung trong quá trình thực hiện.

2. Khả năng trao đổi thông tin: Các chi tiết đàm thoại không đủ để xác định, vì vậy, việc làm giảm thiểu các cuộc thảo luận đòi hỏi giải mã những gì phải được trao đổi.

3. Lựa chọn trong không gian lựa chọn lớn hơn: một hệ thống với các nhân viên đa chức năng có tiềm năng lớn hơn cho việc phát sinh thêm các kế hoạch khác nhau đối với ít nhất 3 lý do: (1) các nhân viên mang các kinh nghiệm khác nhau để thực hiện nhiệm vụ; (2) họ có thể có truy cập khác nhau đến thông tin liên quan với nhiệm vụ; (3) họ đứng trong các mối quan hệ khác nhau đối với bài toán bởi ưu điểm về các vai trò chức năng của họ. Một kết quả quan trọng của việc cố gắng chia sẻ các mục tiêu và các kế hoạch là khi họ xung đột, các lập trình viên phải đàm phán một cách công khai một tiến trình hoạt động chung. Làm như vậy họ tìm được một số lượng lớn hơn các lựa chọn so với lập trình viên đơn lẻ có thể làm. Điều này làm giảm các cơ hội lựa chọn một kế hoạch tồi.

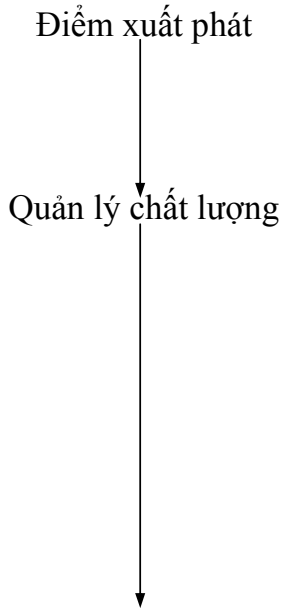
4. Chia sẻ bộ nhớ cho các kế hoạch cũ: một bộ nhớ cho kế hoạch đã được lựa chọn là hữu ích trong các tình huống nơi mà các chủ thể đang tìm ra một tiến trình hoạt động, quyết định trên nó là không phát sinh và phải quay đến một trong các kế hoạch có thể, các kế hoạch cũ hơn. Một lập trình viên làm việc độc lập có thể quên một trong các kế hoạch lựa chọn này [14].

1.2.3. Các yếu tố cơ bản

Viện công nghệ phần mềm đã hướng dẫn các tổ chức phần mềm định ra mô hình hoàn thiện theo khả năng (CMM-Capability Maturity Model) đối với phần mềm [15]. Mục đích của CMM là cung cấp "một cách thức có thứ tự cho các tổ chức để họ xác định các khả năng của quy trình hiện tại của mình để thiết lập các ưu thế cho việc cải tiến. Việc này được làm bằng cách thiết lập và định nghĩa 5 mức tăng dần về khả năng hoàn thiện hơn nữa quá trình [8]. Một quy trình hoàn thiện hơn được định nghĩa tăng dần lên, có tính chất lặp lại và được điều khiển và có thích hợp hơn để có thể dự đoán việc sản xuất phần mềm có chất lượng cao. Năm mức độ tăng dần tính hoàn thiện là: khởi tạo, lặp lại, định nghĩa, quản lý và đánh giá. Mỗi bước này đều định nghĩa các

vùng tiến trình chính (KPA-Key Process Areas). CMM cung cấp các mục tiêu và các hoạt động làm mẫu cho các KPA này, để hướng dẫn các tổ chức trong việc đạt được các mức cao hơn của sự hoàn thiện của quá trình. Các tổ chức có các tiến trình bên trong hoặc bên ngoài, xem xét để xác định mức hoàn thiện của quá trình hiện tại và để xây dựng các kế hoạch cải tiến mức độ hoàn thiện.

Mức	Quá trình cộng tác phần mềm
0.0	Điểm xuất phát
0.1	Chuẩn mã lệnh Đánh giá kích thước Lập kế hoạch cho quá trìnhcải tiến
1.0	Phân tích (mô hình ca sử dụng) Ghi các kết quả phân tích vào thẻ CRC Thiết kế
1.1	Xem xét mã lệnh Xem xét thiết kế Kiểm thử Đánh giá
2.0	Đánh giá kích thước Đánh giá tài nguyên
2.1	Lập kế hoach cho các nhiệm vụ Lập kế hoạch làm việc



Hình 1.1: Mô hình mức tăng trưởng của CSP

CSP cũng sử dụng mô hình mức "tăng trưởng" và có 6 mức. Các mức này được mô tả trong hình 1 và được định nghĩa dưới đây (phần 2.3.2). CSP bao gồm 45 kịch bản, định dạng, mẫu biểu, tiêu chuẩn và danh sách kiểm tra.

Nhìn chung, chúng cũng được dựa trên PSP, nhưng được sửa lại cho đơn giản và chú ý đến sự điều khiển và phân tích của 1 cặp lập trình viên. Thêm vào đó, CSP thay đổi nhiều thành phần của PSP, đặc biệt là trong giai đoạn phân tích và thiết kế. CSP cũng kết hợp chặt chẽ 7 tập chỉ thị và biểu mẫu một cách trực tiếp mà không thay đổi so với PSP, CSP đưa ra một chương trình làm việc cho cặp cộng tác để giúp tổ chức của họ đạt được một mức hoàn thiện cao hơn.

1.2.4. Định nghĩa quá trình cộng tác phần mềm

1.2.4.1. CSP mức 0: Điểm xuất phát.

a. CSP mức 0.0

CSP mức 0.0 không áp đặt và giới thiệu thêm bất kỳ một bước tiến trình nào; các kỹ sư phần mềm sử dụng tiến trình "tự nhiên". Họ có thể dựa vào tài liệu tiến trình, kế hoạch, và các kịch bản phát triển để thực hiện các bước. Tuy nhiên, đây là các bước chung và bất kỳ nhà thiết kế phần mềm nào cũng phải tuân theo. Chẳng hạn các bước này là: 1) tạo ra một thiết kế để nắm bắt các yêu cầu; 2) thực thi theo thiết kế; 3) biên dịch chương trình.

Mục đích của mức này là cung cấp các đánh giá ban đầu, từ đó so sánh các kết quả trong các cải tiến quá trình tiếp theo. Do vậy, chỉ cần thêm một tiến trình tự nhiên để ghi lại dữ liệu về thời gian và lỗi về trong quá trình phát triển của họ. Những người thực hiện phải ghi đúng tổng thời gian mà họ phải làm trong mỗi tiến trình phát triển và ghi lại thông tin về các lỗi mà họ đã loại bỏ được trong các giai đoạn xem xét, dịch và kiểm thử.

Kịch bản cuối cùng quy định việc hoàn thành kế hoạch dự án. Trước khi bắt đầu phát triển, cặp cộng tác đưa ra dự đoán về tổng thời gian phải làm để hoàn thiện sản phẩm. Dữ liệu về thời gian, về lỗi đã ghi lại được tổng hợp trong kế hoạch dự án để sử dụng trong phân tích tiến trình của cặp cộng tác và để đưa ra các dự đoán trong tương lai. Việc sử dụng bảng sẽ dễ dàng cho việc

so sánh giữa thời gian dự đoán hoàn thiện sản phẩm và chi phí thời gian thực sự để hoàn thiện sản phẩm.

b. CSP mức 0.1

Tại mức 0.1 có nhiều cải tiến quy trình nhỏ được thực hiện. Những người thực hiện bắt đầu tuân theo một tiêu chuẩn mã hoá. Đây là một thuận lợi cho các cặp lập trình cộng tác khi mỗi người quay lại bổ sung và xem xét mã lệnh của cộng sự. Đó cũng là một thuận lợi cho việc bảo trì phần mềm khi những người bảo trì phải đọc và hiểu mã lệnh của nhiều lập trình viên khác nhau. Những người thực hiện cũng đếm và ghi lại số dòng lệnh của họ, để làm đơn vị đo kích thước của phần mềm. Các dòng lệnh có thể là một đơn vị đo không hoàn hảo nhưng nó cần thiết để nắm bắt được các mục tiêu của CSP. Khi được sử dụng cùng với tiêu chuẩn mã hoá, các cặp cộng tác đặc biệt có thể sử dụng đơn vị đo là dòng lệnh để so sánh kích thước tương đối của các chương trình khác nhau. Kế hoạch dự án được cập nhật để kết hợp việc ghi lại đơn vị đo bằng dòng lệnh. Thêm vào đó đánh giá về thời gian phát triển được đưa vào mức giai đoạn quy trình (lập kế hoạch, thiết kế, viết mã lệnh, biên dịch, kiểm thử, xem xét lại). Các cặp đánh giá tỷ lệ thời gian phát triển của từng giai đoạn, bằng cách xem xét dữ liệu mà họ ghi được từ khi sử dụng CSP mức 0. Kế hoạch dự án và việc kiểm tra lại được điều chính sao cho phù hợp với nhau.

Cuối cùng, sau mỗi chương trình, các cặp đối chiếu tiến trình của họ xem những gì là tốt và những gì chưa tốt trong quá trình phát triển phần mềm mà họ đã sử dụng thực sự cho chương trình đó và ghi lại những điều đã quan sát được vào kế hoạch cải tiến quy trình (PIP-Process Improvement Proposal). Mục đích của tài liệu này là để đánh dấu những gì mà một cặp nên làm hay không nên làm trong tương lai nhằm đạt được hiệu quả tốt nhất.

1.2.4.2. CSP mức 1: Quản lý chất lượng cộng tác

Các lập trình viên cộng tác, được làm việc theo cặp và sử dụng một số tiêu chí đánh giá rất cơ bản trong CSP mức 0. Các lập trình viên hướng vào các hoạt động đặc biệt, để cải tiến chất lượng sản phẩm trong mức 1. "Mục tiêu của quản lý chất lượng trong PSP là để tìm ra và loại bỏ tất cả các lỗi trước lần biên dịch đầu tiên" [16]. CSP chấp nhận mục tiêu đáng quý này.

a. CSP mức 1.0

Trong mức 1.0, tập trung vào giai đoạn đầu tiên của quá trình phát triển, phân tích và thiết kế. Giai đoạn phân tích đề cập đến vấn đề hiểu bài toán, các mục tiêu và các ràng buộc của chương trình [17]. Các mục tiêu phân tích cần thực hiện bao gồm:

- Hiểu được bài toán mà hệ thống phần mềm phải giải quyết.
- Đưa ra các yêu cầu thích hợp về bài toán và hệ thống.
- Cung cấp cơ sở cho việc trả lời các câu hỏi về các thuộc tính đặc trưng của bài toán và hệ thống.
- Quyết định những gì hệ thống cần làm.
- Quyết định những gì hệ thống không nên làm.
- Xác định những gì hệ thống cần làm để thoả mãn các nhu cầu của người dùng và định nghĩa các tiêu chuẩn chấp nhận được của người dùng.
- Cung cấp cơ sở cho việc phát triển hệ thống.

Trong CSP, quá trình phân tích được thực hiện thông qua việc phát triển các ca sử dụng [18], dựa trên các yêu cầu của người dùng. Các ca sử dụng được thể hiện bằng cách sử dụng mô hình ca sử dụng của UML [19]. Bước đầu tiên phát triển các ca sử dụng là nhận biết các tác nhân; con người hoặc các hệ thống bên ngoài có tác động hoặc hoạt động cùng với hệ thống đang xây dựng. Qua đó có thể tìm ra các ca sử dụng. Một ca sử dụng là một trình tự các hành động được thực hiện bởi một hệ thống, nó cung cấp một kết quả cụ

thể đối với một tác nhân. Một ca sử dụng biểu diễn từ đầu đến cuối một chức năng chính. Thông qua việc xác định các ca sử dụng, các kịch bản, được sử dụng về sau trong quá trình, chúng được nhận biết. "Một ca sử dụng là sự trừu tượng hoá, nó mô tả tất cả các kịch bản có thể được, bằng cách thực hiện chức năng đã được mô tả. Một kịch bản là một thực thể của một ca sử dụng, nó mô tả một tập các sự kiện cụ thể… Các kịch bản được sử dụng làm mẫu để minh hoạ các trường hợp chung, tập trung chủ yếu vào việc làm nổi bật vấn đề hiểu được. Các ca sử dụng được dùng để mô tả tất cả các trường hợp có thể thực hiện được-nó tập trung chủ yếu vào tính hoàn thiện [20].

Trong CSP mỗi ca sử dụng nhận được bằng cách hoàn thiện một luồng sự kiện [21]. Việc hoàn thiện luồng các sự kiện giúp những người thiết kế đưa ra ý tưởng về những yêu cầu hệ thống nên làm và không nên làm. Mô hình ca sử dụng và luồng các sự kiện đều dễ đọc và dễ hiểu đối với người dùng. Vì vậy, chúng có thể cho người dùng biết những gì hệ thống nắm bắt được từ các yêu cầu. Việc phát triển các công cụ này làm cho giai đoạn thiết kế trở nên dễ dàng hơn, khi các mối quan hệ giữa các ca sử dụng được thiết lập. Vì thế, các mục tiêu phân tích, như đã nói ở trên, đạt được thông qua việc tạo ra mô hình ca sử dụng và luồng các sự kiện của ca sử dụng.

Luồng các sự kiện của ca sử dụng cũng rất có lợi cho việc phát triển công cụ kiểm tra hộp đen. Các kỹ sư có thể xác định nhiều con đường thông qua luồng các sự kiện để phát minh ra các công cụ kiểm tra để công nhận tính đúng đắn của chương trình đối với tập các trạng thái. "Các luồng được lựa chọn đã được xác định trong luồng các sự kiện là rất có ích cho việc xác định các trạng thái lỗi cần phải được điều chỉnh trong chương trình và phải được kiểm tra để đảm bảo công việc được tiến hành đúng.

Khi việc phân tích được hoàn thành trong CSP, một thẻ CRC ghi kết quả hoạt động của nhóm được lựa chọn cho thiết kế mức cao (CRC: Class,

Responsibility, and Collaborator). Nội dung trên thẻ CRC cho phép xác định các đối tượng của hệ thống và các giao diện chung của chúng [22]. Các thẻ có chỉ số được được sử dụng để xác định các lớp, nhiệm vụ của các lớp, và các lớp khác mà chúng phải kết hợp để thực hiện các dịch vụ của chúng. Một dạng thẻ CRC điển hình được chỉ ra ở hình 2 (thông thường các thuộc tính của lớp được viết phía sau của thẻ này).

<table>
<tr><td colspan="2">Tên lớp</td></tr>
<tr><td colspan="2">Nhiệm vụ chính của lớp</td></tr>
<tr><td>Các nhiệm vụ khác</td><td>Các lớp cộng tác</td></tr>
<tr><td>…</td><td>…</td></tr>
</table>

Hình 1.2: Thẻ CRC

Các kịch bản được xác định bởi các ca sử dụng, đóng vai trò sử dụng các thẻ để đảm bảo các lớp thực hiện các dịch vụ cần thiết để hoàn thiện mỗi kịch bản. Tốt nhất nên chọn các ca sử dụng có đề cập đến tập các lớp có quan hệ với nhau. Tập các kịch bản có đề cập đến tập các lớp khác nhau nên được áp dụng với thẻ CRC của chính nó. Phần này cho phép có thêm các phiên sử dụng kết quả hoạt động của nhóm.

Một phiên của thẻ CRC tiến hành như sau: Một kịch bản đặc biệt được lựa chọn từ một ca sử dụng (chẳng hạn một luồng đặc biệt thông qua luồng các sự kiện của ca sử dụng). Các kỹ sư phần mềm biểu diễn yêu cầu mà mã lệnh cần thực hiện phù hợp với kịch bản để hoàn thiện một cách thành công. Khi một lớp cần được tạo ra để thực hiện các yêu cầu của một kịch bản, một thẻ trắng được chọn. Tên và mục đích của lớp được viết trên thẻ đó. Khi các lớp đã được xác định, các nhiệm vụ của lớp cũng được xác định là một thành phần của các kịch bản và được viết lên thẻ của lớp. Nếu lớp phải cộng tác với

các lớp khác nhằm hoàn thiện nhiệm vụ của nó, lớp đó phải được ghi ngang hàng với nhiệm vụ trong cột của lớp cộng tác. Một tập các kịch bản điển hình phải đóng vai trò tham gia. Với mỗi kịch bản có vai trò tham gia, người tham gia phải chỉ ra hoặc nhặt ra các thẻ sẽ được dùng để điều khiển nhiệm vụ, nếu các lớp và các nhiệm vụ đã được định nghĩa rồi và/hoặc khởi tạo các lớp và các nhiệm vụ mới.

Có thể xảy ra trường hợp, các lớp sẽ được xác định và bị huỷ bỏ trong phiên sử dụng kết quả hoạt động của nhóm. Việc ghi kết quả hoạt động nhóm cho phép có nhiều lựa chọn thiết kế tại một thời điểm. Các lớp không cần thiết được đưa ra chỗ khác, nhưng không được loại bỏ chúng. “Một thiết kế ban đầu không cần thiết nhưng sau đó có thể sẽ cần đến, hoặc có thể thiết kế cuối cùng là một thay đổi nhỏ của một thiết kế bị hỏng vào lúc khởi tạo” [23].

Qua quá trình này, các lập trình viên đảm bảo rằng các lớp được xây dựng là tốt và chúng thể hiện được chức năng (qua các phương thức của chúng) và các dữ liệu (thông qua các thuộc tính) để điều khiển tập các kịch bản tiêu biểu.

Từ nội dung trên thẻ CRC, việc thiết kế lớp mức cao/UML hầu hết đạt kết quả, bởi vì các lớp đã được xác định cũng như các phương thức và các thuộc tính được yêu cầu.

Việc xây dựng mô hình đối tượng và kết hợp với lược đồ tương tác phải làm giống như mẫu thông tin trên thẻ CRC và phục vụ như một thiết kế mức cao. Cuối cùng, việc phát triển theo chu kỳ được khuyến khích. Người ta đề nghị rằng một khi thiết kế mức cao và việc xem xét đã hoàn thành, cặp cộng tác dừng dự án của họ lại một chút lúc có khoảng từ 100 đến 300 dòng lệnh. Cặp cộng tác nên thực hiện lại thiết kế, mã lệnh, xem xét, biên dịch và kiểm tra mức thấp đối với mỗi sự gia tăng.

b. CSP mức 1.1

CSP mức 1.1, tập trung vào việc kiểm tra thiết kế và mã lệnh. Ở đây các cặp lập trình viên xem xét sản phẩm mà họ tạo ra.

Cặp cộng tác xem xét lại công việc của họ, để quy định danh sách kiểm tra thiết kế và mã lệnh. Đôi khi với các cặp lập trình cộng tác, công việc sẽ chỉ có một người làm bởi một trong hai người không thể tham gia (ốm, bận, hoặc làm việc khác). Có thể nhiều cặp lập trình thấy rằng có những đoạn chương trình được viết một cách hiệu quả mà chỉ cần một người. Do vậy, CSP có hai kiểu danh sách kiểm tra thiết kế và hai kiểu danh sách kiểm tra mã lệnh. Một kiểu cho dùng cho những người làm việc độc lập, một kiểu dùng cho các cặp cộng tác. Việc cần làm là mã hoá cơ sở. Vì vậy, các danh sách kiểm cho làm việc một mình phải được làm rất cẩn thận. Tuy nhiên, khi cả hai người cùng làm thì danh sách kiểm tra không cần phải chi tiết quá, bởi việc kiểm tra xem xét là thường xuyên trong khi thực hiện. Việc xem xét đối với cặp cộng tác chủ yếu tập trung vào các yếu tố quan trọng nhất như "có phải thiết kế tất cả các mục trong một bảng chi tiết không?" hoặc "ta đã hoàn thành thiết kế chưa?". Việc xem xét của làm việc độc lập cũng phải kiểm tra lỗi cú pháp và lỗi logic mức thấp.

Tuy nhiên, cần nhấn mạnh rằng, các cặp cộng tác phải thường xuyên quan tâm đến các danh sách kiểm tra. Nếu cặp cộng tác không phạm một lỗi nghiêm trọng nào, mục này nên được loại bỏ khỏi danh sách.

1.2.4.3. CSP mức 2: Quản lý dự án cộng tác

Mức 2 đề cập đến việc bổ sung các hoạt động quản lý dự án khả thi vào quy trình của nhóm cộng tác. "Quản lý dự án bao gồm các hoạt động giám sát để đảm bảo sự phân chia hệ thống chất lượng cao theo thời gian và chi phí phù hợp[20]. Các kỹ thuật quản lý dự án trong PSP không cần phải thay đổi

trong CSP. Những người làm việc cộng tác có thể dễ dàng ứng dụng các kỹ thuật này giống như những người làm việc độc lập.

a. CSP Level 2.0

Thông thường kích thước sản phẩm và các đánh giá tài nguyên được phát triển thông qua các dự đoán và có thể dựa vào cảm giác. Tuy nhiên, việc sử dụng các phương thức ở mức 2.0, một người có thể trả lời một cách hệ thống các câu hỏi của người quản lý phát triển phần mềm, chẳng hạn: "Bạn có thể hoàn thành dự án này vào cuối tháng 3 không? Khách hàng muốn có nó vào tháng 3". Nó nâng khả năng trả lời câu hỏi của các kỹ sư từ một câu trả lời "có thể" đến câu trả lời chẳng hạn "Tôi có thể nói chính xác với bạn đến 90% là dự án này khoảng 4000 đến 4500 dòng lệnh. Dựa vào dữ liệu được ghi nhận của chính tôi, điều này làm tôi mất 100 giờ - tôi cảm thấy rất tiện khi cam kết vào tháng 3". Tóm lại, phương thức giúp các kỹ sư đưa ra lời cam kết mà họ có thể gặp.

Bước đầu tiên trong việc xây dựng lời cam kết là việc phát triển thiết kế khái niệm mức cao. Thiết kế này thiết lập một cách tiếp cận thiết kế sơ bộ và đặt tên cho các đối tượng sản phẩm được mong đợi và chức năng của chúng[8]. Ý tưởng là không tốn quá nhiều thời gian vào thiết kế khái niệm – cân bằng nhu cầu để yêu cầu các đối tượng sẽ cần đến mà không cần đưa ra thiết kế mức cao. Thiết kế khái niệm này sẽ được dùng để xác định đánh giá/lời cam kết về tài nguyên yêu cầu để xây dựng sản phẩm. Các quyết định có tiếp tục hay không với sản phẩm sẽ dựa vào các đánh giá và các lời cam kết này. Các hoạt động phân tích và thiết kế tiếp theo sẽ cải tiến thiết kế này nếu quyết định phát triển sản phẩm.

b. CSP Level 2.1

Việc thực hiện các hoạt động đặt ra ở mức 2.1, cho các kỹ sư một kế hoạch có thứ tự, để thực hiện các nhiệm vụ đã yêu cầu để hoàn thành một dự

án và một chương trình làm việc để xác định và trao đổi tình trạng công việc của họ. Các kỹ sư thực thi nhiệm vụ, lập kế hoạch và điều chỉnh theo phương thức giá trị thực sự.

Giá trị thực sự của một nhiệm vụ đặc biệt, được dựa trên tỷ lệ phần trăm của tổng hiệu quả của dự án theo kế hoạch mà nhiệm vụ sẽ thực hiện. Khi các nhiệm vụ được hoàn thiện, giá trị thực sự được sử dụng để chỉ ra tỷ lệ phần trăm công việc được hoàn thiện. Khi kéo dài qua các tuần, giá trị thực sự của dự án có thể được so sánh với giá trị theo kế hoạch của nó, để xác định tình trạng, để đánh giá tốc độ của tiến trình, và để dự đoán ngày hoàn thiện dự án[16].

Đối với việc lập kế hoạch cho nhiệm vụ, các kỹ sư liệt kê một danh sách các nhiệm vụ cần thiết để hoàn thành dự án. Sau đó, họ dự đoán về khoảng thời gian cần thiết để hoàn thành nhiệm vụ (thông thường một tỷ lệ phần trăm tổng đánh giá tài nguyên được phát triển trong mức 2.0). Mỗi nhiệm vụ được đánh dấu bởi một giá trị thực sự, dựa trên tỷ lệ phần trăm tổng thời gian nhiệm vụ được dự đoán để thực hiện. Các kỹ sư có được giá trị này cho tới khi hoàn thiện nhiệm vụ. Các kỹ sư có thể dễ dàng trao đổi trạng thái hoàn thiện dựa trên các kết quả lập kế hoạch nhiệm vụ.

Việc lập lịch được sử dụng để xác định xem mất bao nhiêu thời gian cho dự án. Việc đánh giá tài nguyên ở mức 2.0 được chia cho các cam kết thời gian cho mỗi tuần của dự án.

Trên đây là mô hình mức tăng trưởng của CSP, được áp dụng trong quá trình phát triển phần mềm. Về mặt cấu trúc, CSP giống với PSP, nhưng thời gian để hoàn thiện một phần mềm lại nhiều hơn. Để khắc phục nhược điểm này, khi phát triển phần mềm theo CSP, cần kết hợp với một phương pháp để có thể giảm bớt thời gian thực hiện. XP một phương pháp cho phép phát triển nhanh một phần mềm là một sự lựa chọn phù hợp. Dưới đây tôi xin trình bày

việc kết hợp CSP với XP để tạo ra một quy trình phát triển phần mềm với kích thước lớn, chất lượng cao với thời gian vừa phải.

1.3. KẾT HỢP XP TRONG CSP ĐỂ PHÁT TRIỂN PHẦN MỀM

Hiện nay, việc phát triển nhanh một phần mềm và tạo được niềm tin cho khách hàng là vấn đề được các tổ chức phần mềm rất quan tâm, vì đây là một trong những lợi thế cạnh tranh để nhận được các dự án phần mềm. Để giải quyết vấn đề này, các tổ chức phần mềm có thể sử dụng phương pháp XP. Tuy nhiên, XP chỉ phù hợp với các dự án vừa và nhỏ, không thể áp dụng vào việc phát triển các dự án phần mềm lớn. Để khắc phục nhược điểm này, các tổ chức phần mềm có thể lựa chọn CSP, và áp dụng các kỹ thuật của XP vào quy trình thực hiện, nhằm thúc đẩy nhanh quá trình phát triển.

Giữa CSP và XP có một điểm tương đồng là các nhiệm vụ được thực hiện bởi các cặp lập trình, nên việc kết hợp chúng sẽ không mấy khó khăn. CSP được thực hiện theo mô hình mức tăng trưởng, gồm có 6 mức. Trong mỗi mức ta kết hợp với các kỹ thuật của XP để làm tăng tốc độ của tiến trình thực hiện.

Trong mức 0 của CSP, ta áp dụng các kỹ thuật của XP để lập một kế hoạch thực hiện. Từ việc trao đổi với khách hàng để nắm bắt các yêu cầu của hệ thống, chuyển các yêu cầu thành các nhiệm vụ. Giao các nhiệm vụ cho các cặp lập trình. Các cặp tạo ra một thiết kế đơn giản cho nhiệm vụ, thực hiện cài đặt và cuối cùng là kiểm tra lại những công việc đã làm. Việc cài đặt được thực hiện theo một chuẩn mã lệnh. Khi các cặp hoàn thành nhiệm của mình, họ kết hợp các kết quả với nhau để có được chương trình hoàn chỉnh. Chương trình được, kiểm thử, biên dịch rồi giao cho khách hàng. Từ các thông tin phản hồi của khách hàng, và các kết quả đánh giá, các lập trình viên cải tiến chương trình để có được chương trình hoàn thiện hơn.

Mức 1 của CSP, tập trung vào việc phân tích và thiết kế hệ thống. Trong mức này, ta đưa vào giá trị trao đổi thông tin và phản hồi thông tin của XP. Các lập trình viên thường xuyên trao đổi với khách hàng, để nắm bắt các thay đổi và các yêu cầu mới. Khách hàng được tham gia vào quá trình phát triển để biết rằng các yêu cầu của họ được thực hiện, và tin tưởng vào tính khả thi của dự án.

Mức 2 của CSP là việc quản lý dự án, việc này được thực hiện trong mọi quy trình, và CSP cũng được thực hiện giống như PSP. Mức này không có sự kết hợp của XP.

1.4. KẾT LUẬN

Trong chương này, luận văn nghiên cứu ba vấn đề:

- XP, một phương pháp lập trình mới, rất linh hoạt, được ứng dụng để phát triển nhanh một phần mềm, với các yêu cầu luôn thay đổi, kích thước vừa phải. Bao gồm các khái niệm về XP, các giá trị, các quy tắc và các hoạt động của XP được sử dụng để phát triển phần mềm.

- CSP, một quá trình được áp dụng để phát triển các phần mềm có kích thước lớn. Gồm các khái niệm về CSP, cách tiếp cận của CSP và mô hình mức tăng trưởng của CSP áp dụng trong quá trình phát triển phần mềm.

- Đưa ra khả năng ứng dụng XP trong CSP để phát triển phần mềm.

Chương 2. CÁC "THÔNG LỆ" TRONG XP

2.1. TỔNG QUAN VỀ CÁC THÔNG LỆ TRONG XP

XP gồm có 12 "thông lệ", được chia thành 4 nhóm, các bước thực hiện này nhận được từ các bước thực hiện tốt nhất được đưa ra trong công nghệ phần mềm.

**** Nhóm các "thông lệ" với sự phản hồi thông tin liên tục gồm:***

- Lập trình theo cặp
- Lập kế hoạch thực hiện
- Phát triển hướng vào việc kiểm tra
- Làm việc theo nhóm

**** Nhóm các "thông lệ" là quá trình liên tục***

- Kết hợp thường xuyên
- Cải tiến thiết kế
- Hoàn thiện theo từng bước nhỏ

**** Nhóm các "thông lệ" thực hiện với sự hiểu biết chung của nhóm lập trình:***

- Tiêu chuẩn mã hoá
- Sở hữu chung mã lệnh
- Thiết kế được làm đơn giản
- Hệ thống trong suốt

**** Nhóm các "thông lệ" thể hiện lợi ích cho các lập trình viên***

- Tốc độ làm việc vừa phải

2.2. CÁC THÔNG LỆ TRONG XP

2.2.1. Tiêu chuẩn mã hoá

Tiêu chuẩn mã hoá được chấp nhận dựa trên một tập các luật, mà toàn bộ nhóm phát triển đồng ý thực hiện theo đó trong cả dự án. Tiêu chuẩn mã hoá xác định một kiểu và một định dạng thích hợp cho mã nguồn, trong phạm vi ngôn ngữ lập trình đã được lựa chọn. Tiêu chuẩn mã hoá có thể là các quy ước chuẩn được chỉ rõ bởi ngôn ngữ lập trình (ví dụ: các quy ước về mã lệnh đối với ngôn ngữ lập trình Java), hoặc được lựa chọn theo thói quen của nhóm phát triển.

2.2.2. Sở hữu chung mã lệnh

Sở hữu chung mã lệnh nghĩa là mọi người chịu trách nhiệm chung về mã lệnh được tạo ra, mọi người trong nhóm lạp trình đều được phép sửa đổi một đoạn mã lệnh bất kỳ hay bổ sung vào một đoạn mã lệnh mới. Hoạt động này được đưa ra bởi việc lập trình theo cặp; khi làm việc luân chuyển trong các cặp khác nhau, các lập trình viên hiểu được toàn bộ mã lệnh của chương trình. Thuật lợi chủ yếu của việc sở hữu chung mã lệnh là làm tăng tốc độ của quá trình phát triển, bởi nếu có lỗi xảy ra trong mã lệnh thì bất kỳ một lập trình viên nào cũng có thể khắc phục nó.

Việc cho các lập trình viên quyền sửa đổi mã lệnh, các lỗi sẽ được tìm ra bởi các lập trình viên biết được những gì mình đang làm, nhưng không biết trước được những sự phụ thuộc tạo ra lỗi. Tuy nhiên, trường hợp này đã có các bộ kiểm đủ tốt để xử lý: nếu không biết trước được những phụ thuộc tạo ra các lỗi, thì chạy các bộ kiểm tra, chúng sẽ chỉ ra các chỗ bị lỗi.

2.2.3. Sự kết hợp thường xuyên

Nhóm phát triển nên luôn luôn làm việc trên phiên bản mới nhất của phần mềm. Từ các thành viên trong các nhóm khác nhau có thể có các phiên bản đã lưu lại những sửa đổi và cải tiến khác nhau, họ cố gắng xem xét mã

lệnh trong phiên bản chương trình hiện tại trong thời gian khoảng vài giờ đồng hồ, hoặc khi một tín hiệu lỗi xuất hiện. Sự kết hợp thường xuyên sẽ tránh được sự chậm trễ sau chu kỳ dự án, gây ra bởi lần kết hợp.

2.2.4. Cải tiến thiết kế

Bởi XP chỉ ủng hộ việc lập trình cho những vấn đề cần thiết ở thời điểm hiện tại, và việc thực hiện việc đó sao cho càng đơn giản càng tốt. Đôi khi điều này sẽ có kết quả đối với một hệ thống đang bị đình trệ. Một trong những điều đáng chú ý của vấn đề này là yêu cầu đối với việc bảo trì: các sửa đổi về chức năng đòi hỏi sửa đổi nhiều bản sao chép mã lệnh. Một vấn đề đáng chú ý khác là những sửa đổi trong một phần của mã lệnh ảnh hưởng đến nhiều thành phần khác. XP cho rằng khi xảy ra điều này, hệ thống sẽ cho bạn thấy để refactor (phân tích lại) mã lệnh bằng cách sửa đổi cấu trúc, làm cho nó đơn giản hơn và phổ dụng hơn.

2.2.5. Thiết kế đơn giản

Các lập trình viên nên theo cách tiếp cận "đơn giản là tốt nhất" để thực hiện thiết kế phần mềm. Bất cứ khi nào một phần mã lệnh mới được viết, lập trình viên nên tự hỏi mình "có cách nào đơn giản hơn vẫn cho kết quả tương tự?". Nếu câu trả lời là có, thì cách thức đơn giản hơn nên được lựa chọn. Cải tiến mã lệnh (sẽ được trình bày ở phần sau) cũng nên được sử dụng, để làm cho mã lệnh phức tạp trở nên đơn giản hơn.

2.2.6. Các bước hoàn thiện nhỏ

Việc giao phần mềm được thực hiện bởi các bước được quyết định từ trước. Kế hoạch từng bước được xác định khi bắt đầu thực hiện dự án. Thông thường mỗi bước là một công đoạn nhỏ của quá trình phần mềm, nó có thể chạy mà không phụ thuộc vào các thành phần sẽ được thực hiện sau. Các bước hoàn thiện nhỏ làm cho khách hàng tin tưởng vào lợi ích của sự tiến triển của dự án.

2.2.7. Tốc độ làm việc vừa phải

Là tiến độ thực hiện phù hợp với khả năng của lập trình viên. Khái niệm này cho biết các lập trình viên và các nhà phát triển phần mềm không nên làm việc hơn 40 giờ một tuần, và nếu có thì cũng không nên quá 1 tuần. Từ khi các chu kỳ phát triển là các chu kỳ ngắn được kết hợp thường xuyên, dẫn đến toàn bộ chu kỳ phát triển là thường xuyên hơn, các dự án trong XP không tuân theo thời gian đặc biệt nào mà các dự án khác yêu cầu.

Ở đây cũng đề cập đến vấn đề con người sẽ thực hiện tốt nhất và sáng tạo nhất nếu được nghỉ ngơi một cách hợp lý.

2.2.8. Hệ thống trong suốt

Hệ thống trong suốt là một khái niệm, trong đó các lớp và các phương thức cần được làm đơn giản, sao cho các thành viên nhóm dự đoán được chức năng của một lớp hay một phương thức đặc biệt, mà chỉ cần nhìn vào tên của nó. Chẳng hạn, hệ thống thư viện có thể tạo lớp loan_records cho lớp borrowers, và nếu một mục trở nên quá hạn nó có thể thực hiện thao tác make_overdue trên lớp catalogue. Với mỗi lớp hoặc mỗi thao tác chức năng của nó phải dễ hiểu đối với toàn nhóm.

2.2.9. Lập trình theo cặp

Lập trình theo cặp nghĩa là toàn bộ mã lệnh được tạo ra bởi hai người lập trình cho một nhiệm vụ và trên một trạm làm việc. Một lập trình viên điều khiển trạm làm việc và nghĩ ra hầu hết mã lệnh một cách chi tiết. Người kia tập trung nhiều hơn vào toàn bộ công việc, và thường xuyên xem xét mã lệnh do người thứ nhất tạo ra. Các lập trình viên thường xuyên thay đổi vai trò cho nhau.

Các cặp không cố định: điều này có nghĩa là ngoài việc các lập trình viên cung cặp thay đổi vai trò cho nhau, thì các lập trình viên cũng nên luân chuyển các lập trình viên giữa các cặp khác nhau, điều này được thực hiện

càng nhiều càng tốt. Thực hiện tốt việc này sẽ làm cho tất cả các lập trình viên đều hiểu biết những gì họ làm, và từ đó hiểu biết rõ toàn bộ hệ thống.

2.2.10. Lập kế hoạch dự án

Quá trình lập kế hoạch cơ bản trong XP là lập kế hoạch dự án. Phần này sẽ giải thích quá trình lập kế hoạch dự án bằng cách sử dụng các mô hình tiến trình.

Quá trình lập kế hoạch được chia làm 2 giai đoạn:

Lập kế hoạch từng bước:

Giai đoạn này tập trung vào việc xác định các yêu cầu trong một bước và khi nó đang được phân chia. Người dùng và nhà phát triển hệ thống cùng tham gia vào giai đoạn này. Lập kế hoạch từng bước lại bao gồm 3 giai đoạn nhỏ:

- Giai đoạn tìm hiểu: Trong giai đoạn này người dùng đưa ra các yêu cầu của họ đối với hệ thống. Các yêu cầu này sẽ được ghi vào phiếu yêu cầu người dùng.

- Giai đoạn chuyển giao: trong giai đoạn này nghiệp vụ và phát triển sẽ tự chuyển giao chức năng của nó và kỳ hạn của bước tiếp theo.

- Giai đoạn điều chỉnh: Trong giai đoạn điều chỉnh kế hoạch có thể được điều chỉnh cho phù hợp, các yêu cầu mới có thể được bổ sung hoặc các yêu cầu đang tồn tại có thể được sửa đổi hoặc loại bỏ cho phù hợp.

Lặp lại việc lập kế hoạch:

Giai đoạn này chuẩn bị các hoạt động và các nhiệm vụ cho nhà phát triển. Trong giai đoạn này không cần sự tham gia của người dùng. Lặp lại việc lập kế hoạch cũng được chia làm 3 giai đoạn nhỏ:

- Giai đoạn tìm hiểu: Giai đoạn này các yêu cầu được chuyển thành các nhiệm vụ khác nhau. Các nhiệm vụ được ghi vào các phiếu ghi nhiệm vụ.

- Giai đoạn chuyển giao: Các nhiệm vụ sẽ được phân công cho các lập trình viên, và thời gian hoàn thành nhiệm vụ cũng được đánh giá.

- Giai đoạn điều chỉnh: các nhiệm vụ được thực hiện và kết quả cuối cùng được so sánh với các yêu cầu ban đầu của người dùng. Nếu nó chưa thực sự phù hợp thì sẽ được điều chỉnh.

2.2.10.1. Lập kế hoạch từng bước

a. Giai đoạn tìm hiểu

Đây là quá trình lặp lại việc thu thập các yêu cầu và đánh giá thực tế công việc của các yêu cầu đó.

- Nắm bắt các yêu cầu từ người dùng: nghiệp vụ chỉ có nếu bài toán được đặt ra; khi trao đổi với người dùng, nhà phát triển sẽ cố gắng để nắm bắt được mục đích của nó và xác định các yêu cầu thực sự của bài toán.

- Viết các yêu cầu: Dựa trên bài toán nghiệp vụ, nhà phát triển viết một bản các yêu cầu. Việc này được thực hiện bởi nghiệp vụ hệ thống, đó là những gì họ muốn hệ thống thực hiện. Một điều quan trọng là việc phát triển không làm ảnh hưởng đến bản ghi yêu cầu này. Bản ghi các yêu cầu được viết trên phiếu yêu cầu người dùng.

- Tách yêu cầu: trong khi thực hiện, nếu không đánh giá được các yêu cầu, nó cần phải được tách ra và được viết lại. Điều này không ảnh hưởng đến các yêu cầu nghiệp vụ mà người dùng đã đưa ra.

- Đánh giá yêu cầu: Việc phát triển phải đánh giá thời gian cần thiết để thực hiện một yêu cầu được cung cấp bởi phiếu ghi yêu cầu người dùng.

Khi việc không còn yêu cầu nghiệp vụ nào thêm, thì bước sang giai đoạn chuyển giao.

b. Giai đoạn chuyển giao

Giai đoạn này giải quyết các vấn đề về xác định các chi phí, lợi nhuận, và kế hoạch làm việc thực tế. Nó gồm 4 thành phần:

- Phân loại theo giá trị: phân loại các yêu cầu khách hàng theo giá trị công việc cần thực hiện.

- Phân loại rủi ro: phân loại các yêu cầu theo sự rủi ro.

- Thiết lập tiến độ thực hiện: xác định tiến độ mà họ có thể thực hiện dự án.

- Lựa chọn phạm vi: Các yêu cầu người dùng cuối cùng trong bước tiếp theo sẽ được chọn ra. Dựa trên các yêu cầu người dùng, người ta xác định thời gian thực hiện của một bước.

**** Phân loại theo giá trị***

Phân loại các yêu cầu khách hàng bởi giá trị công việc được thực hiện. Họ sẽ sắp đặt chúng trong 3 cột:

- Giá trị không có ý nghĩa: các yêu cầu không có chức năng trong hệ thống hoặc không có ý nghĩa.

- Giá trị có ý nghĩa: các yêu cầu người dùng có giá trị thực hiện.

- Tính hấp dẫn: các yêu cầu người dùng không có ý nghĩa về mặt giá trị trao đổi; chẳng hạn có thể là một cải tiến trong cách dùng hoặc việc trình diễn.

**** Phân loai theo rủi ro***

Các nhà phát triển phân loại các yêu cầu người dùng dựa vào sự rủi ro. Họ cũng phân loại thành 3 cột: các yêu cầu người dùng có mức rủi ro thấp, trung bình và cao.

Xác định chỉ số rủi ro: Đặt cho mỗi yêu cầu người dùng một chỉ số từ 0 đến 2, dựa trên mỗi yếu tố sau đây:

- Tính hoàn thiện

    + Hoàn thiện (chỉ số là 0)

    + Không hoàn thiện (1)

    + Không xác định được (2)

- Tính không ổn định
    + Thấp (0)
    + Trung bình (1)
    + Cao (2)
- Tính phức tạp
    + Đơn giản (0)
    + Chuẩn (1)
    + Phức tạp (2)

Với các chỉ số cho một yêu cầu người dùng, người ta chia các yêu cầu người dùng với chỉ số rủi ro là: thấp (0-1), trung bình (2-4), cao (5-6).

c. Giai đoạn điều chỉnh

Trong giai đoạn điều chỉnh các lập trình viên và người dùng có thể điều chỉnh quá trình. Nghĩa là họ có thể thực hiện các sửa đổi cần thiết. Các yêu cầu người dùng độc lập, hoặc các quyền ưu tiên của các yêu cầu người dùng khác nhau, có thể thay đổi, do các đánh giá có thể sai. Đây là cơ hội để điều chỉnh lại kế hoạch sao cho phù hợp.

2.2.10.2. Lặp lại việc lập kế hoạch

a. Giai đoạn tìm hiểu

Giai đoạn tìm hiểu của việc lặp lại kế hoạch bao gồm việc tạo ra các nhiệm vụ và việc đánh giá thời gian thực hiện.

- Tập hợp các yêu cầu người dùng: tập hợp và viết tất cả các yêu cầu người dùng cho bước tiếp theo.

- Kết hợp/tách nhiệm vụ: nếu lập trình viên không thể đánh giá được nhiệm vụ bởi nó quá lớn hoặc quá nhỏ, lập trình viên cần kết hợp hoặc tách các nhiệm vụ để sao cho có thể thữch hiện được.

- Đánh giá nhiệm vụ: đánh giá thời gian cần thiết để thực hiện nhiệm vụ.

b. Giai đoạn chuyển giao

Trong giai đoạn chuyển giao của việc lặp lại kế hoạch, các lập trình viên được phân công các nhiệm vụ tương ứng với các yêu cầu người dùng khác nhau.

- Nhận một nhiệm vụ: Mỗi lập trình viên nhận một nhiệm vụ thích hợp với mình để thực hiện có hiệu quả nhất.

- Đánh giá nhiệm vụ: do lập trình viên đã nhận được nhiệm vụ thích hợp, anh ta nên đưa ra đánh giá cuối cùng về nhiệm vụ cần thực hiện.

- Thiết lập yếu tố tải: yếu tố tải biểu diễn thời gian thực hiện phát triển lý tưởng với từng lập trình viên trong giai đoạn lặp lại kế hoạch. Chẳng hạn, trong một tuần với 40 giờ làm việc, với 5 giờ thực hiện trao đổi, việc này không quá 35 giờ.

- Sự cân bằng: khi tất cả các lập trình viên trong nhóm được phân công các nhiệm vụ, người ta so sánh thời gian thực hiện nhiệm vụ đã được đánh giá và yếu tố tải. Sau đó các nhiệm vụ được cân bằng giữa các lập trình viên. Nếu một lập trình viên bị quá tải thì các lập trình viên khác cần giúp đỡ thêm.

c. Giai đoạn điều chỉnh

Giai đoạn này thực điều chỉnh các nhiệm vụ nếu nó chưa thực sự phù hợp với yêu cầu đặt ra.

- Nhận phiếu yêu cầu nhiệm vụ: Lập trình viên nhận một trong các phiếu yêu cầu nhiệm vụ mà anh ta được chuyển giao.

- Tìm người cộng sự: Một lập trình viên sẽ thực hiện nhiệm vụ cùng với một lập trình viên khác. Điều này được trình bày trong nội dung về lập trình theo cặp.

- Thiết kế nhiệm vụ: nếu cần thiết, các lập trình viên sẽ thiết kế theo chức năng của nhiệm vụ, để thực hiện đạt kết quả mong muốn.

- Viết các bộ kiểm tra: trước khi viết mã lệnh, các lập trình viên phải viết các bộ kiểm tra tự động.

- Viết mã lệnh: Các lập trình viên thực hiện viết mã lệnh cho yêu cầu.

- Thực hiện kiểm tra: Các bộ kiểm tra được kích hoạt để kiểm tra mã lệnh.

- Cải tiến mã lệnh: Loại bỏ các lỗi được tìm thấy trong mã lệnh, cải tiến các đoạn mã lệnh tồi, để có mã lệnh chất lượng cao hơn.

- Kiểm tra chức năng: các bộ kiểm tra chức năng (dựa trên các yêu cầu được kết hợp bởi yêu cầu người dùng và phiếu yêu cầu nhiệm vụ) được thực hiện.

2.2.11. Phát triển hướng vào việc kiểm tra

Các bộ kiểm tra tự động được chạy để kiểm tra chức năng của các đoạn mã lệnh (ví dụ: các lớp, các phương thức). Trong XP các bộ kiểm tra được viết trước khi viết mã lệnh. Cách tiếp cận này được dùng để mô phỏng việc lập trình viên nghĩ về các điều kiện trong đó mã lệnh do anh ta viết có thể có lỗi. XP cho rằng, các lập trình viên chỉ nên kết thúc công việc của mình khi trong mã lệnh do anh ta viết (có thể) không còn lỗi.

2.2.12. Làm việc theo nhóm

Trong XP, người dùng không phải là người chịu toàn bộ chi phí xây dựng hệ thống, nhưng thực sự là người sử dụng hệ thống. XP cho rằng, người dùng nên quan tâm đến việc xây dựng hệ thống ở mọi thời điểm và luôn đặt sẵn các câu hỏi. Trong trường hợp này, nhóm phát triển một hệ thống quản lý tài chính nên có một người quản lý tài chính trong nhóm.

Ngoài các “thông lệ” nêu trên, XP cũng đưa ra các kỹ thuật cải tiến nhằm làm tăng hiệu quả của mã lệnh có sẵn mà không làm thay đổi mục đích chung của hệ thống. Các kỹ thuật cải tiến mã lệnh, cho phép nhóm lập trình sử dụng các bộ kiểm tra tự động để tìm ra các lỗi và xử lý chúng một cách

hiệu quả. Dưới đây sẽ trình bày khái niệm "cải tiến mã lệnh" (refactor) và các kỹ thuật sử dụng trong để cải tiến mã lệnh.

2.3. CẢI TIẾN MÃ LỆNH

2.3.1. Giới thiệu về "cải tiến mã lệnh"

Các phương pháp lập trình truyền thống cho rằng: "nếu mã lệnh đang chạy được, không cần phải sửa đổi". Tuy nhiên, trong nhiều trường hợp một hệ thống phần mềm đang hoạt động, vẫn cần phải sửa đổi. Chẳng hạn, khi muốn bổ sung thêm một chức năng mới hoặc chuyển hệ thống sang một môi trường khác, nhưng hệ thống không đủ mềm dẻo để có thể làm được việc đó. Trong trường hợp như vậy, hầu hết các lập trình viên đều biết rằng, họ cần phải làm gì. Họ chỉ cần cấu trúc lại hệ thống là có thể thực hiện các sửa đổi cần thiết. Tuy nhiên, mỗi sửa đổi đều tiềm ẩn trong nó những rủi ro. Mỗi lần sửa đổi hệ thống, đều có thể xuất hiện những lỗi mới. Đó là lý do tại sao bạn muốn có một cách để đảm bảo rằng các thay đổi của bạn không tạo các lỗi mới.

Với việc tạo ra phương pháp phát triển chương trình theo hướng đối tượng rất linh hoạt và có tính chất lặp, khái niệm "cải tiến mã lệnh" lần đầu tiên được đưa ra vào những năm 1990. Ý tưởng cơ bản của cải tiến mã lệnh là sửa đổi mã lệnh bằng một cách thức sao cho giảm thiểu được các lỗi trong mã lệnh. Đúng hơn là cải tiến mã lệnh là việc làm thay đổi cấu trúc bên trong của phần mềm, làm cho nó dễ hiểu hơn và chi phí sửa đổi ít hơn mà không làm thay đổi các hành vi của hệ thống. Ở đây, ta trình bày cách thức làm thế nào để kiểm chứng tính hiệu quả cải tiến mã lệnh và làm thế nào để chúng tác động đến quá trình phát triển phần mềm. Đồng thời, ta cũng thảo luận các lý do tại sao cần sử dụng cải tiến mã lệnh và các kỹ thuật cơ sở kèm theo để thực hiện cải tiến mã lệnh. Với việc dựng lên các vấn đề liên quan đến các chi tiết của cải tiến mã lệnh và cách thức để tự động hoá chúng [27].

2.3.2. Làm tài liệu cải tiến mã lệnh

Cải tiến mã lệnh là một phương pháp nằm trong các tài liệu được tổ chức có tính hệ thống, nó mô tả các kỹ thuật được chứng minh, để nâng cao tính an toàn của phần mềm [31]. Chúng cung cấp các nguy cơ gây lỗi và dự đoán các cách để tránh các nguy cơ đó. Trong [27], Martin Fowler giới thiệu một danh mục các cải tiến mã lệnh nơi họ dùng một định dạng chuẩn để biểu diễn tuần tự hơn 70 cải tiến mã lệnh cần thiết [32]. Mỗi cải tiến mã lệnh gồm 5 phần:

Tên: Xác định kỹ thuật cải tiến mã lệnh và giúp việc xây dựng từ điển chung cho các nhà phát triển phần mềm.

Tác dụng: cho biết khi nào và ở đâu bạn cần cải tiến mã lệnh và nó dùng để làm gì. Mục tác dụng cũng giúp bạn tìm ra các cải tiến mã lệnh thích hợp trong một tình huống được đặt ra. Nó cũng bao gồm những đoạn mã nguồn hoặc các lược đồ UML để chỉ ra một kịch bản đơn giản trước hoặc sau đó.

Lý do sử dụng: diễn tả tại sao cải tiến mã lệnh nên được làm bằng cách liệt kê các trường hợp không nên sử dụng.

Cách thực hiện: là thành phần cung cấp từng bước mô tả việc thực hiện cải tiến mã lệnh như thế nào. Các bước càng ngắn gọn càng tốt để có thể làm theo nó một cách dễ dàng.

Ví dụ: Minh hoạ cải tiến mã lệnh được sử dụng như thế nào trong chương trình thực sự.

Hầu hết các kỹ thuật cải tiến mã lệnh rất đơn giản và tên của chúng có ý nghĩa. Nhìn chung, cải tiến mã lệnh thường là các bước rất nhỏ, để chuyển đổi từ các thiết kế thủ tục truyền thống sang các thiết kế hướng đối tượng. Những thảo luận chi tiết là cốt lõi chính của dạnh mục, để thêm vào các chỉ thị cụ thể, cho biết làm thế nào để cải tiến mã lệnh trong các tình huống khác nhau.

Khi tên của các kỹ thuật cải tiến mã lệnh được đặt ra, các kỹ thuật cải tiến mã lệnh có liên quan mật thiết với các mẫu thiết kế [28]. Một mẫu thiết

kế cho biết làm thế nào để giải quyết một bài toán thiết kế theo chu kỳ, với một cách thức có tính kỷ luật trong ngữ cảnh đặt ra. Cải tiến mã lệnh, nói theo cách khác, hướng dẫn bạn cách nâng cao hiệu suất làm việc hiện tại, vì vậy nó mang lại một thiết kế tốt hơn. Thông thường thiết kế này là thiết kế mẫu. Trong nhiều trường hợp, bạn kết thúc việc áp dụng một tập các cải tiến mã lệnhs để quay lại một đoạn mã lệnh đặc biệt của một đối tượng trong một mẫu thiết kế.

Cải tiến mã lệnh rất hữu ích trong việc phát triển các chương trình ứng dụng có tính hiệu quả và mềm dẻo và chúng phù hợp với quá trình phát triển chương trình lặp. Cải tiến mã lệnh cũng là một thành phần kỹ thuật chủ yếu trong XP [24,34].

2.3.3. Các đoạn mã lệnh tồi

Những phẩm chất nào làm cho một phần mềm được xem là tốt? Người ta đề nghị rằng, nên phát triển các chương trình dễ hiểu, có tất cả lập luận được xác định một cách tập trung, cho phép sửa đổi mà không làm thay đổi các hành vi đang tồn tại, và lập luận có điều kiện được biểu diễn càng dễ càng tốt. Các chương trình không có các phẩm chất đó là chương trình tồi (một thuật ngữ được đúc kết bởi Kent Beck). Trong tài liệu [27], Kent Beck việc đặt tên và mô tả một số các đoạn mã lệnh tồi và đề nghị sử dụng các kỹ thuật cải tiến mã lệnh để loại bỏ chúng. Nguồn gốc của các lỗi trong lập trình là mã lệnh được sao chép. Có thể dễ dàng thấy tại sao cần phải bảo trì phần mềm.

Tuy nhiên bạn có thể gặp khó khăn khi cần thực hiện các sửa đổi tương tự ở nhiều đoạn mã lệnh khác nhau và rất khó biết được khi nào bạn thực hiện xong các sửa đổi đó. Đương nhiên, việc sao chép mã lệnh cũng làm tăng số lượng mã lệnh và làm cho các hệ thống khó hiểu hơn và khó bảo trì hơn.

Một nguyên nhân khác cho các đoạn mã lệnh tồi là việc tổ chức các lớp và các phương thức chưa phù hợp. Chúng có thể quá lớn và quá phức tạp hoặc

quá nhỏ và quá đơn giản. Việc thiếu các phẩm chất tốt do sự kết hợp không chặt chẽ giữa các cấu trúc và sự liên kết bên trong chúng cũng có thể gây ra các đoạn mã lệnh tồi. Các nguyên nhân khác dẫn đến các đoạn mã lệnh tồi bao gồm việc sử dụng quá nhiều hoặc quá ít sự uỷ quyền và việc sử dụng các điều khiển và các cấu trúc dữ liệu không phải hướng đối tượng.

2.3.4. Các kỹ thuật cơ bản sử dụng để cải tiến mã lệnh

Làm thế nào để ta chỉ ra được các phẩm chất tốt trong phần mềm và loại bỏ những đoạn mã lệnh tồi. Một trong các thủ tục cơ sở của cải tiến mã lệnh là việc bổ sung gián tiếp.

Gián tiếp hầu hết các dạng của nó có ý nghĩa là xác định các cấu trúc (chẳng hạn, các lớp, các phương thức) và đặt tên cho chúng. Sử dụng các cấu trúc được đặt tên làm cho mã lệnh dễ hiểu, bởi vì nó cho bạn cách thức để giải thích mục đích (dựa vào tên của các lớp và các phương thức) và chức năng (cấu trúc các lớp và thân các phương thức) của các lớp và các phương thức một cách rõ ràng. Kỹ thuật tương tự cho phép sự phân chia logic (chẳng hạn, các phương thức được gọi ở những nơi khác nhau hoặc các phương thức trong lớp cha được chia sẻ bởi các lớp con). Sự phân chia logic, ngược lại, giúp bạn quản lý sự thay đổi trong các hệ thống. Cuối cùng, tính đa hình (một dạng khác của gián tiếp) cung cấp một cách mềm dẻo, không tường minh để biểu diễn lập luận có điều kiện.

Giống như hầu hết các kỹ thuật khác, chìa khoá của thành công với kỹ thuật gián tiếp là chỉ đặt một số lượng vừa phải của nó ở nơi có thể thực hiện được. Có quá nhiều các kết quả gián tiếp được đưa vào hoặc các kết quả gián tiếp tồi được đưa vào trong một hệ thống chưa được hoàn thiện là điều không nên làm. Kỹ thuật gián tiếp không có tác dụng thường được tìm thấy trong một thành phần được dùng chung hoặc được dùng ở nhiều dạng, nhưng không làm thay đổi thêm gì nữa trong quá trình phát triển. Một danh mục cải tiến mã

lệnh cho bạn biết thời điểm bắt đầu để quyết định kỹ thuật gián tiếp nên được sử dụng ở đâu và sử dụng như thế nào.Về căn bản, nếu bạn gặp kỹ thuật gián tiếp và thấy rằng nó không có lợi, bạn cần loại bỏ nó.

2.3.5. Cải tiến mã lệnh trong quá trình phát triển phần mềm

Cải tiến mã lệnh có thể là một ý tưởng lựa chọn thiết kế trước một cách cẩn thận. Loại thiết kế có tính suy đoán này là một sự cố gắng để đưa tất cả các phẩm chất tốt vào hệ thống trước khi thực hiện viết mã lệnh. Điều quan trọng của quá trình này là rất dễ để dự đoán sự nhầm lẫn. Đôi khi XP được xem như một lược đồ tác động trở lại sự quan sát đó bằng cách giữ lại toàn bộ giai đoạn thiết kế.

Tuy nhiên, cải tiến mã lệnh cũng có thể được sử dụng theo một cách thức dễ hơn. Thay vì loại bỏ giai đoạn thiết kế hoàn thiện, bạn chuyển từ thiết kế phức tạp sang một thiết kế đơn giản hơn. Điều này có thể nhận thấy bởi bạn không cần phải thấy trước tất cả những sửa đổi trong khi cải tiến. Với cải tiến mã lệnh, bạn có thể làm đơn giản thiết kế bởi các sửa đổi về mặt thiết kế không tốn kém.

Khi phát triển phần mềm sử dụng các kỹ thuật cải tiến mã lệnh, bạn phân chia thời gian giữa hai hoạt động riêng biệt: cải tiến mã lệnh và bổ sung chức năng mới. Khi bổ sung chức năng mới, không nên sửa đổi mã lệnh đang tồn tại. Nếu có một diễn biến khó khăn trong quá trình phát triển, bởi việc thực hiện không hỗ trợ tốt đặc trưng mới, bạn cần cải tiến mã lệnh của hệ thống lần đầu tiên để xem xét việc sửa đổi.

Sau khi bổ sung một chức năng mới, bạn phải thực hiện kiểm tra bổ sung để thấy rằng chức năng được thực hiện đúng [26]. Bạn cũng có thể xem xét các kiểm tra bằng cách kiểm chứng tài liệu yêu cầu đối với đặc trưng và viết chúng trước khi bổ sung. Việc sử dụng các kiểm tra sẵn có để đảm bảo rằng việc cải tiến mã lệnh không làm thay đổi hành vi hay tạo ra các lỗi mới. Việc

kiểm tra được thực hiện càng đơn giản càng tốt. Đó là lý do tất cả các kiểm tra nên được thực hiện tự động và nên để tự chúng kiểm tra các kết quả. Điều này cho phép bạn thực hiện một cách thường xuyên khi biên dịch. Khi gặp lỗi, bạn có thể tập trung việc gỡ lỗi trong một đoạn mã lệnh nhỏ mà bạn bổ sung lần cuối cùng trước các kiểm tra thành công.

Kỹ thuật kiểm tra làm việc tốt nhất với cải tiến mã lệnh là kiểm tra hộp trắng mức lớp [25]. Cách thức tốt để viết các bộ kiểm tra như vậy, cho một hệ thống hướng đối tượng, có một lớp kiểm tra cho mỗi lớp khởi tạo lớp quan trọng. Bạn cũng có thể sử dụng chương trình kiểm tra (chẳng hạn Junit [30] để quản lý việc kiểm tra và đưa ra cách thức kiểm tra được chuẩn hoá. Chương trình kiểm tra nên cung cấp các cách thức mềm dẻo để kết hợp các kiểm tra cho đến khi chúng được thực hiện theo trình tự bất kỳ.

Việc kiểm tra nên hướng vào các rủi ro. Có nghĩa là bạn kiểm tra các thành phần lớp phức tạp và có giá trị nhất. Nhớ tập trung vào việc kiểm tra xung quanh phạm vi đầu vào và các điều kiện đặc biệt khác. Đừng cố kiểm tra tất cả mọi thứ hoặc không kiểm tra gì cả.

Bên cạnh việc kiểm tra, việc xem xét mã lệnh cũng rất quan trọng đối với qua trình xác định chất lượng phần mềm. Quá trình cải tiến mã lệnh có thể là một phần không thể thiếu của việc xem xét, đặc biệt nếu chúng được quản lý trong các nhóm nhỏ. Cải tiến mã lệnh cho bạn cơ hội để thấy hiệu quả toàn diện của những sửa đổi được đưa ra.

2.3.6. Lợi ích của cải tiến mã lệnh

Cải tiến mã lệnh có thể được sử dụng với nhiều mục đích khác nhau. Đầu tiên, cải tiến mã lệnh giúp mã lệnh giữ nguyên được khuôn mẫu của nó. Nếu không thực hiện cải tiến mã lệnh việc thiết kế chương trình sẽ không thể thực hiện một cách hoàn hảo. Khi sửa đổi mã lệnh (thường không hiểu biết đầy đủ việc thiết kế các đối tượng để thực hiện) dần dần làm mất tính cấu trúc

của nó. Khi mất tính cấu trúc, mã lệnh trở nên khó hiểu và làm tăng khả năng mất tính cấu trúc của thiết kế.

Cải tiến mã lệnh làm cho mã lệnh trở nên dễ hiểu hơn. Điều này là cần thiết cho việc chuyển tải ý tưởng về mã lệnh đến những người khác. Nó cũng làm cho mã lệnh dễ hiểu hơn cho chính người viết ra nó. Điều đó cũng quan trọng bởi vì các lập trình viên không thể khẳng định họ có thể nhớ các ý tưởng của họ sau vài tuần nữa.

Các lập trình viên cũng có thể sử dụng cải tiến mã lệnh để hiểu được ý tưởng của những đoạn mã lệnh mà họ mới đọc lần đầu. Khi xem xét một đoạn mã lệnh lập trình viên cố gắng tìm hiểu xem đoạn mã lệnh đó làm gì. Khi hiểu được đoạn mã lệnh đó, các lập trình viên cải tiến để nó phản ánh tốt hơn sự hiểu biết của họ về mục đích của nó. Sau đó kiểm tra xem hệ thống có thực hiện theo đúng chức năng của nó hay không. Nếu mọi việc vẫn tốt, thì nghĩa là các lập trình viên đã hiểu biết và có cách xử lý đúng về một phần hệ thống mà họ xem xét. Ngược lại, các lập trình viên nên tìm hiểu thêm đoạn mã lệnh mà họ đang xem xét.

Nó có thể hơi khác thường, nhưng Fowler khẳng định rằng thuận lợi thực sự của cải tiến mã lệnh là nó giúp các lập trình viên phát triển phần mềm nhanh hơn [27]. Có thể tin rằng cải tiến mã lệnh là cải tiến chất lượng và làm cho mã lệnh dễ hiểu hơn, nhưng làm thế nào để tăng tốc độ phát triển? Các lập trình viên nghĩ rằng tất cả các sửa đổi và bản chất lặp của cải tiến mã lệnh, không đề cập đến hiệu quả lớn đặt vào việc kiểm tra, sẽ làm cho tốc độ phát triển chậm đi.

Bí quyết là việc bổ sung các chức năng mới và việc tìm ra các lỗi rất hiệu quả khi các lập trình viên làm việc trên một hệ thống với một thiết kế chặt chẽ mà họ có thể hiểu được cặn kẽ. Nếu không thực hiện cải tiến mã lệnh, hệ thống của họ sẽ sụp đổ ngay từ đầu. Đó là điều tại sao nó không

mang lại lợi ích lâu dài, nếu giữ nguyên cải tiến mã lệnh cùng với thiết kế, làm quá tải toàn bộ cải tiến mã lệnh và việc kiểm tra được kết hợp. Nếu các lập trình viên bỏ qua việc cải tiến mã lệnh, họ sẽ kết thúc rất sớm trong một tình huống rắc rối với việc chèn vào các chức năng mới, bởi các hiệu quả không mong muốn. Tương tự các sửa đổi sẽ trở nên khó hơn bởi các lập trình viên bắt đầu có sự sao chép mã lệnh.

2.3.7. Các vấn đề cần lưu ý khi cải tiến mã lệnh

Ngay khi có lợi ích trong nhiều trường hợp, cải tiến mã lệnh không luôn dễ hoặc thậm chí không có ích trong nhiều trường hợp khác. Chẳng hạn, các hệ thống dựa trên hệ quản trị cơ sở dữ liệu Access là rất khó sửa đổi. Hầu hết các ứng dụng cho doanh nghiệp liên quan rất mật thiết với lược đồ cơ sở dữ liệu hỗ trợ chúng, làm cho cơ sở dữ liệu và thậm chí hệ thống xây dựng trên đó rất khó sửa đổi. Nếu các lập trình viên có khả năng phân biệt lớp các đối tượng trong cơ sở dữ liệu, họ vẫn bị ép buộc phải chuyển đổi dữ liệu khi lược đồ cơ sở dữ liệu thay đổi. Việc chuyển đổi dữ liệu cũng là vấn đề rất khó khăn đối với việc sửa đổi một ứng dụng [29].

Việc sửa đổi một giao diện trong một hệ thống hướng đối tượng luôn là một sửa đổi chính khi so sánh với việc sửa đổi sự thực thi. Chẳng may hầu hết các cải tiến mã lệnh đều làm thay đổi giao diện. Vấn đề này dễ xử lý nếu các lập trình viên truy cập đến mã lệnh tương ứng với các giao diện đã được sửa đổi - họ cũng chỉ có thể sửa đổi những phần đó.

Vấn đề nghiêm trọng hơn nữa với các giao diện được sử dụng phổ biến (ví dụ, các giao diện đã được sử dụng bởi những người bên ngoài tổ chức hoặc bởi mã lệnh mà các lập trình viên không sửa đổi). Nếu các lập trình viên phải sửa lại các giao diện đã được sử dụng phổ biến, họ nên giữ lại cả phiên bản mới và phiên bản cũ và phiên bản cũ nên gắn với phần đã được sửa đổi.

Những rắc rối này cho thấy, không nên đưa ra sử dụng những giao diện quá sớm.

Đôi khi các lập trình viên không nên cải tiến mã lệnh một hệ thống ngay từ đầu. Lý do là hệ thống đó cần được viết lại từ đầu. Điều này thường cho thấy rằng hệ thống không làm việc một cách đơn giản và vẫn còn rất nhiều lỗi không dễ gì có thể sửa được.

2. 4. KẾT LUẬN

Mỗi phương pháp phát triển phần mềm đều có một tập các bước để điều khiển quá trình thực hiện. Trên đây là một tập các "thông lệ" được sử dụng để điều khiển quá trình phát triển phần mềm theo XP. Việc nắm được các thông lệ này, cho phép người lập trình xác định được các bước cần thực hiện và các tiêu chuẩn cần tuân theo khi sử dụng XP.

Ngoài ra, với mục đích tạo ra một quá trình phát triển phần mềm kết hợp giữa CSP và XP, việc nghiên cứu và nắm được các thông lệ trong XP cũng là việc rất cần thiết. Trong chương 3, người viết luận văn xin trình bày quá trình phát triển phần mềm theo CSP kết hợp với XP.

Chương 3. ỨNG DỤNG LẬP TRÌNH LINH HOẠT TRONG QUY TRÌNH CỘNG TÁC PHẦN MỀM

Chương này trình bày việc ứng dụng XP trong CSP, mục đích là để đưa ra một quá trình phần mềm nhằm phát triển các ứng dụng chất lượng cao, trong một thời gian vừa phải. Quá trình phát triển được thực hiện theo mô hình mức tăng trưởng của CSP, trong mỗi mức ta áp dụng các giá trị, các quy tắc, các "thông lệ" trong XP nhằm giảm thời gian cần thiết để hoàn thiện mỗi mức này.

3.1. Ý TƯỞNG LẬP TRÌNH LINH HOẠT TRONG QUY TRÌNH CỘNG TÁC PHẦN MỀM

Hiện nay, việc phát triển nhanh một phần mềm và tạo được niềm tin cho khách hàng là vấn đề được các tổ chức phần mềm rất quan tâm, vì đây là một trong những lợi thế cạnh tranh để nhận được các dự án phần mềm. Để giải quyết vấn đề này, các tổ chức phần mềm có thể sử dụng phương pháp XP. Tuy nhiên, XP chỉ phù hợp với các dự án vừa và nhỏ, không thể áp dụng vào việc phát triển các dự án phần mềm lớn. Để khắc phục nhược điểm này, các tổ chức phần mềm có thể lựa chọn CSP, và áp dụng các kỹ thuật của XP vào quy trình thực hiện, nhằm thúc đẩy nhanh quá trình phát triển.

3.2. QUY TRÌNH PHÁT TRIỂN PHẦN MỀM ỨNG DỤNG XP TRONG CSP

3.2.1. Mức 0: Điểm xuất phát

Với mục tiêu giao phần mềm trong thời gian ngắn, và tạo sự tin tưởng cho khách hàng. Ban đầu, các nhà phát triển sẽ nhanh chóng tạo ra một phần mềm để đáp ứng mục tiêu này. Tuy nhiên, phần mềm mới chỉ ở mức đơn giản, và trong quá trình này các nhà phát triển tập trung vào việc đánh giá quá trình thực hiện, và căn cứ vào đó để cải tiến phần mềm trong các bước tiếp

theo để có được phần mềm chất lượng cao. Trong CSP đây là phần mềm mức 0, mức này việc phát triển ứng dụng được thực hiện theo XP.

Quá trình này được thực hiện như sau:

3.2.1.1. Mức 0.0

a. Phát triển chương trình

Quá trình này có sự tham gia của cả người dùng và các lập trình viên. Các lập trình viên trao đổi thông tin với người dùng, để nhận định các yêu cầu của hệ thống và thảo luận với nhau, để xác định những gì hệ thống nên làm và những gì hệ thống không nên làm. Sau khi xác định được các yêu cầu mà hệ thống cần thực hiện, các lập trình viên trao đổi với người dùng để điều chỉnh lại các yêu cầu cho phù hợp. Các bước được thực hiện như sau:

❖ **Lập kế hoạch từng bước**

**** Tìm hiểu bài toán***

Trao đổi với người dùng về hệ thống cần được xây dựng, ghi các yêu cầu vào phiếu yêu cầu.

**** Chuyển giao yêu cầu***

Tìm hiểu hoạt động nghiệp vụ của hệ thống, môi trường hoạt động, trao đổi với người dùng để chuyển các yêu cầu thành các chức năng mà hệ thống cần thực hiện và chuẩn bị cho bước tiếp theo.

**** Điều chỉnh yêu cầu***

Trao đổi với người dùng về các chức năng, để điều chỉnh lại cho phù hợp, và nắm bắt thêm các chức năng mới của hệ thống.

❖ **Lặp lại kế hoạch**

Bước này chuẩn bị các hoạt động và các nhiệm vụ cho nhà phát triển, gồm 3 giai đoạn.

**** Giai đoạn tìm hiểu***

Chuyển các chức năng thành các nhiệm vụ, ghi các nhiệm vụ vào phiếu nhiệm vụ.

**** Giai đoạn chuyển giao***

Hình thành các cặp lập trình, phân công các nhiệm vụ cho các cặp lập trình, đồng thời đánh giá thời gian cần thiết để hoàn thành nhiệm vụ.

**** Giai đoạn điều chỉnh***

Các cặp thực hiện nhiệm vụ được giao, kết quả được đánh giá dựa vào yêu cầu ban đầu. Nếu chưa phù hợp, thì cần phải điều chỉnh lại.

Sau khi các nhiệm vụ được giao cho các cặp, họ tiến hành thực hiện nhiệm vụ của mình như sau:

- Thiết kế: Cặp lập trình tạo ra thiết kế cho chức năng hệ thống mà họ thực hiện. Tuy nhiên, thiết kế được làm sao cho đơn giản nhất có thể được.

Trước khi viết mã lệnh, các cặp lập trình kết hợp các kết quả thiết kế để có bản thiết kế tổng thể. Tiếp theo họ phải thảo luận để đưa ra chuẩn mã lệnh.

- Viết mã lệnh: Các cặp lập trình viên nhận nhiệm vụ, viết mã lệnh cho nhiệm vụ theo chuẩn mã lệnh đã đưa ra.

- Cải tiến mã lệnh: Sử dụng các kỹ thuật cải tiến mã lệnh, để tìm và sửa lỗi trong mã lệnh nhằm nâng cao chất lượng của mã lệnh. Việc này có thể làm thay đổi cấu trúc của mã lệnh, để phù hợp với yêu cầu đặt ra. Tuy nhiên, không làm thay đổi chức năng chung của hệ thống.

- Biên dịch chương trình.

- Giao chương trình cho người dùng và nhận các thông tin phản hồi.

b. Đánh giá chương trình

Mục đích của quá trình này là đưa ra các đánh giá ban đầu, từ đó so sánh các kết quả đạt được với kết quả trong các cải tiến tiếp theo.

- Các lập trình viên đánh giá hiệu quả của quá trình phát triển, bằng cách so sánh thời gian mà họ sử dụng để hoàn thiện nhiệm vụ với thời gian dự kiến. Số lỗi mà họ tìm thấy và loại bỏ trong mã lệnh.

- Dựa vào thông tin phản hồi từ người dùng, các lập trình viên đánh giá hiệu quả của chương trình. Từ đó, xác định những việc nên làm, không nên làm, các yêu cầu được thay đổi và các yêu cầu mới.

Ngoài ra, trong quá trình này cũng đưa ra đánh giá hiệu quả của các cặp lập trình, từ đó đưa ra những điều chỉnh phù hợp để làm tăng hiệu quả làm việc, hoặc thực hiện luân chuyển các lập trình viên giữa các cặp.

3.2.1.2. Mức 0.1

Mức này các lập trình viên thực hiện các công việc sau:

- Cải tiến cải tiến chương trình
- Đánh giá kích thước của chương trình
- Đánh giá thời gian thực hiện cải tiến

a. Cải tiến chương trình

Giai đoạn này, các cặp lập trình thực hiện cải tiến nhiệm vụ mà họ đã tạo ra trong mức 0.0. Việc cải tiến chương trình gồm:

- Cải tiến mã lệnh để được mã lệnh có chất lượng cao hơn.
- Sửa đổi mã lệnh nếu yêu cầu tương ứng bị thay đổi.
- Bổ sung mã lệnh mới nếu một yêu cầu mới được đưa ra.

b. Đánh giá kích thước của chương trình

Kích thước của chương trình được tính bằng số dòng lệnh. Vì vậy, sau khi thực hiện cải tiến chương trình, các cặp lập trình đếm số dòng lệnh mà họ đã viết. So sánh với kích thước của chương trình đã tạo ra ở mức 0.0.

c. Đánh giá thời gian thực hiện cải tiến

Ghi nhận thời gian thực hiện cải tiến chương trình. So sánh với thời gian mà họ đã thực hiện trong mức 0.0 với thời gian thực hiện cải tiến. Việc ghi

nhận và so sánh thời gian thực hiện giữa các giai đoạn, giúp các lập trình viên đánh giá hiệu quả của quá trình thực hiện, từ đó đưa ra các điều chỉnh kịp thời để đảm bảo tiến độ thực hiện.

Cuối cùng, họ xem xét lại toàn bộ quá trình thực hiện, xem những gì là tốt, những gì chưa tốt trong quá trình phát triển, ghi nhận lại những điều đó trong khi xem xét. Từ đó, đưa ra quyết định những gì nên làm và những gì không nên làm trong giai đoạn tiếp theo, để có được chương trình chất lượng cao.

3.2.2. Mức 1: Quản lý chất lượng cộng tác

Sau khi tạo ra chương trình ở mức đơn giản, mức 0, để giao cho người dùng, cũng như đánh giá được thời gian và kích thước của chương trình, xác định được các lỗi xảy ra trong chương trình, và các nguy cơ xảy lỗi trong quá trình thực hiện. Đồng thời xác định được những gì nên làm và những gì không nên làm cho hệ thống. Các lập trình viên tiến hành phát triển chương trình thực sự đáp ứng được yêu cầu của hệ thống.

Trong CSP, giai đoạn này là mức 1: quản lý chất lượng cộng tác. Trong giai đoạn này các cặp tập trung vào việc cải tiến chất lượng chương trình, và mục tiêu là loại bỏ tất cả các lỗi trước lần dịch đầu tiên. Quá trình thực hiện được tiến hành theo 2 mức nhỏ, mức 1.0: cải tiến chất lượng chương trình và mức 1.1: kiểm tra và loại bỏ lỗi.

3.2.2.1. Mức 1.0: Cải tiến chất lượng chương trình

Mức này các nhà phát triển tập trung vào giai đoạn đầu tiên của quá trình, phân tích và thiết kế. Giai đoạn phân tích đề cập đến vấn đề hiểu bài toán, xác định các mục tiêu và các ràng buộc của chương trình. Quá trình phân tích được thực hiện thông qua việc phát triển các ca sử dụng. Các ca sử dụng được thể hiện bằng mô hình ca sử dụng của UML.

Với kết quả đã được thực hiện ở mức 0, quá trình phân tích và thiết kế sẽ được thực hiện rất nhanh, bởi vì các lập trình viên chỉ cần cải tiến các nhiệm vụ mà họ đã thực hiện.

Trong quá trình phân tích và thiết kế, các lập trình viên vẫn thường xuyên trao đổi với người dùng, để nắm bắt sự thay đổi yêu cầu hệ thống, hoặc các yêu cầu mới để sửa đổi và bổ sung các yêu cầu vào các nhiệm vụ của hệ thống.

Việc phân tích và thiết kế hệ thống được thực hiện như sau:

a. Phân tích hệ thống

Dựa vào các yêu cầu đã được nhận định ở mức 0, và việc trao đổi thường xuyên với người dùng để nắm bắt các yêu cầu được thay đổi và các yêu cầu mới. Các lập trình viên mô tả các yêu cầu hệ thống bằng mô hình ca sử dụng của UML.

Các bước phân tích gồm:

- Nhận biết các tác nhân hệ thống
- Nhận định các ca sử dụng
- Đặc tả các ca sử dụng
- Thiết lập biểu đồ ca sử dụng
- Phát hiện các lớp/đối tượng tham gia các ca sử dụng
- Biểu diễn vai trò của các lớp/đối tượng

❖ **Nhận biết các tác nhân hệ thống**

Xác định các yếu tố, con người hoặc các hệ thống ngoài tác động hoặc hoạt động cùng với hệ thống, qua đó phát hiện ra các ca sử dụng. Giai đoạn này các lập trình viên làm việc kết hợp, chưa phân nhiệm vụ riêng cho các cặp. Sau khi nhận biết được các đối tác, các lập trình viên thiết lập biểu đồ mức khung cảnh.

❖ **Nhận định các ca sử dụng**

Ca sử dụng là biểu diễn của một chuỗi hành động, mà hệ thống thực hiện nhằm cung cấp một kết quả cụ thể cho một tác nhân. Thực chất một ca sử dụng là một chức năng chính của hệ thống. Việc nhận định các ca sử dụng gồm 2 bước:

- Tìm hiểu các chức năng của hệ thống
- Nhận định các ca sử dụng

Sau khi nhận định được các ca sử dụng, các lập trình viên chuyển sang việc đặc tả các ca sử dụng.

❖ **Đặc tả các ca sử dụng**

Việc đặc tả các ca sử dụng được thực hiện bởi các cặp lập trình, tuỳ theo số cặp lập trình hiện có, và số lượng các ca sử dụng, mỗi cặp lập trình có thể nhận một hoặc nhiều hơn các ca sử dụng. Việc đặc tả các ca sử dụng gồm:

- Mô tả tóm tắt ca sử dụng: tên, mục đích, tóm lược, tác nhân, …
- Mô tả các kịch bản: chỉ rõ các điều kiện đầu vào, đầu ra, các kịch bản thông lệ, khả dĩ, ngoại lệ…
- Các yêu cầu về giao diện: có thể thêm các ràng buộc về giao diện người-máy, hiển thị gì, người dùng có thể khởi phát những thao tác nào.
- Các ràng buộc phi chức năng: có thể thêm các thông tin sau: tần suất, khối lượng, khả năng sẵn dùng, mức tin cậy, tính toàn vẹn, tính bảo mật, hiệu năng… Các thông tin này có ích cho việc nắm bắt các nhu cầu kỹ thuật sau này.

❖ **Thiết lập biểu đồ ca sử dụng**

Để thiết lập biểu đồ ca sử dụng cần làm các việc sau:

- Xác định mối liên quan giữa các tác nhân
- Xác định mối liên quan giữa các đối tác và ca sử dụng

- Xác định mối liên quan giữa các ca sử dụng
- Thiết lập biểu đồ ca sử dụng

❖ **Phát hiện các đối tượng/lớp tham gia ca sử dụng**

Mục đích của bước này là:

- Phát hiện các đối tượng tham gia ca sử dụng
- Xác định vai trò của mỗi đối tượng trong ca sử dụng
- Nghiên cứu các liện kết giữa các lớp tham gia ca sử dụng
- Nghiên cứu sự tương tác giữa các lớp tham gia ca sử dụng

Đầu vào của bước này là các ca sử dụng và các kịch bản của chúng. Các ca sử dụng được nghiên cứu song song bởi các cặp lập trình (các cặp làm việc độc lập), để phát hiện các đối tượng/lớp tham gia ca sử dụng đó. Các lập trình viên cần thực hiện các công việc sau:

**** Phát hiện các lớp/đối tượng tham gia ca sử dụng***

Các lớp tham gia ca sử dụng được gọi là các lớp phân tích, gồm 3 loại:

- Các lớp biên: Là các lớp nhằm chuyển đổi thông tin giao tiếp giữa tác nhân và hệ thống.

- Các lớp điều khiển: Là các lớp điều hành sự diễn biến trong một ca sử dụng.

- Các lớp lĩnh vực: Là các lớp nghiệp vụ, đây là các lớp mà dữ liệu và các mối liên quan của chúng còn được lưu lại sau khi ca sử dụng đã kết thúc.

**** Diễn tả vai trò của các lớp bằng biểu đồ cấu trúc đa hợp***

Khi phân loại các lớp thành các lớp biên, lớp điều khiển và lớp lĩnh vực, ta chỉ ra vai trò cụ thể của mỗi lớp, với từ vựng của ứng dụng, bằng cách thiết lập một biểu đồ cấu trúc đa hợp.

**** Diễn tả cấu trúc tĩnh của hợp tác bằng một biểu đồ lớp***

Mục đích của việc này là lập một biểu đồ lớp cho mỗi ca sử dụng, phản ánh cấu trúc tĩnh của hợp tác. Biểu đồ chính là cái nền trên đó diễn ra hoạt động tương tác giữa các lớp.

Đến đây phân tích đã hoàn thành đối với các cặp lập trình, họ kết hợp các kết quả phân tích, trao đổi với nhau để điều chỉnh sự phù hợp giữa các kết quả phân tích của các cặp và có được "bức tranh toàn cảnh" về hệ thống.

Kết quả của việc phân tích là các lớp, thông tin về mỗi lớp được ghi vào một thẻ gọi là thẻ CRC (hình 2). Việc ghi nội dung trên thẻ CRC được làm như sau: Chọn một kịch bản từ một ca sử dụng, các lập trình viên biểu diễn yêu cầu mà mã lệnh cần thực hiện phù hợp với kịch bản.

Khi cần tạo ra một lớp để thực hiện các yêu cầu của kịch bản, một thẻ trắng được lựa chọn. Lập trình viên ghi tên và các nhiệm vụ của lớp trên thẻ, nếu lớp cộng tác với một lớp khác thì tên lớp cộng tác được viết trên thẻ ngang hàng với nhiệm vụ của lớp này.

Khi các lập trình viên ghi xong thông tin của các lớp trên các thẻ, thì bước phân tích hoàn thành, họ chuyển sang bước thiết kế hệ thống.

b. Thiết kế hệ thống

Trong phân tích, ta tập trung nghiên cứu cấu trúc logic của thông tin, cần thiết cho việc xây dựng một giải pháp nghiệp vụ. Còn mục đích của thiết kế là nghiên cứu các phương pháp tốt nhất để cài đặt các cấu logic nói trên, nhằm tối ưu hoá hiệu năng của ứng dụng. Việc phân tích được triển khai theo nhãn quan ứng dụng, còn thiết kế tiếp nhận đầu vào là các mô hình từ các bước phân tích hệ thống trước đây, và được triển khai theo nhãn quan kỹ thuật.

Dựa vào các kết quả phân tích, các lập trình viên thực hiện cải tiến các thiết kế đơn giản, được thực hiện ở mức 0.

Các bước thiết kế hệ thống:

- Thiết kế các lớp
- Thiết kế các liên kết
- Thiết kế các thuộc tính
- Thiết kế các thao tác
- Đánh giá nghiệm thu

Khi hoàn thành các bước nêu trên, kết quả thu được là các lớp thiết kế chi tiết. Tuy nhiên, việc thiết kế không phải chỉ thực hiện theo một chiều, mà các bước thiết kế từ bước 1 đến bước 4, được thực hiện lặp đi lặp lại cho đến khi đạt được thiết kế hoàn chỉnh. Dưới đây ta sẽ nghiên cứu các bước thiết kế chi tiết.

❖ **Thiết kế các lớp**

Bước này các lập trình viên làm việc theo cặp, mỗi cặp thực hiện thiết kế nhận một (hoặc một số) thẻ CRC ghi nội dung các lớp, dựa vào thông tin về lớp ghi trên thẻ CRC, các lập trình viên thực hiện thiết kế các lớp bao gồm:

- Chuyển các lớp phân tích thành các lớp thiết kế cho phù hợp với các yêu cầu kỹ thuật.
- Phân bổ lại hay giải phóng bớt trách nhiệm cho các lớp phân tích.
- Thêm các lớp mới để cài đặt cấu trúc dữ liệu
- Thêm các lớp mới để cài đặt các khái niệm phân tích
- Thêm các lớp mới vì mục đích tối ưu hoá

❖ **Thiết kế các liên kết**

Thiết lập mối liên kết giữa các lớp.

❖ **Thiết kế các thuộc tính.**

Xác định các thuộc tính của lớp, kiểu của các thuộc tính, và tầm nhìn của các thuộc tính và cách truy cập vào chúng.

❖ **Thiết kế các thao tác**

Là bước cuối cùng trong vòng lặp thiết kế. Đây là công việc tốn nhiều thời gian nhất, nội dung thiết kế các thao tác đưa ra một hình ảnh khá chi tiết cho các phương thức của các lớp.

❖ **Đánh giá và nghiệm thu**

Đến đây có thể nói rằng việc phân tích và thiết kế tạm hoàn thiện, các cặp quay lại xem xét kết quả đã làm được. So sánh với thiết kế ban đầu (mức 0), và với các yêu cầu đặt ra của hệ thống.

Trên đây trình bày các bước phân tích và thiết kế một hệ thống được xây dựng theo CSP, kết hợp với phương pháp XP. Tuy nhiên, ở đây chỉ đưa ra phương pháp chứ không đi sâu vào việc phân tích và thiết kế hệ thống.

Kết thúc giai đoạn thiết kế, các lập trình viên chuyển sang viết mã lệnh.

c. Viết mã lệnh

Việc viết mã lệnh được thực hiện bởi các cặp lập trình. Dựa vào các kết quả ở bước thiết kế, các lập trình viên cải tiến mã lệnh trong cơ sở mã lệnh chung mà họ đã viết ở mức 0, để có được mã lệnh chất lượng cao hơn, đáp ứng được yêu cầu của hệ thống.

Khi hoàn thành việc cải tiến mã lệnh, các lập trình viên chuyển sang mức tiếp theo của qúa trình, mức 1.1 kiểm tra thiết kế và mã lệnh.

3.2.2.2. Mức 1.1: Kiểm tra

CSP mức 1.1 tập trung vào việc kiểm tra thiết kế và mã lệnh. Các mục tiêu kiểm tra bao gồm:

- Kiểm tra tổng quát:

+ Kiểm tra lại khi mỗi bước thực hiện khi nó được hoàn thiện

+ Hoàn thiện danh sách kiểm tra cho một đơn vị chương trình khi chuyển sang kiểm tra xem xét đơn vị chương trình tiếp theo.

- Kiểm tra tính hoàn thiện:

Để đảm bảo rằng các yêu cầu và các đặc tả được mô tả một cách đúng đắn và đầy đủ bởi thiết kế.

+ Tất cả các đầu ra xác định đều được đưa ra.

+ Tất cả các đầu vào cần thiết đều được cung cấp.

+ Tất cả các chức năng yêu cầu đã được bắt đầu.

- Kiểm tra thiết kế lớp:

+ Tất cả các thành phần dữ liệu riêng tư và công cộng đều được lấy ra/đặt vào đúng vị trí của nó theo yêu cầu.

+ Sự liên kết dữ liệu: bạn có thể đi qua một mạng lưới cộng tác giữa các lớp để lấy những thông tin cần thiết để phân phối các dịch vụ dựa trên một tập biểu diễn các kịch bản không?

+ Sự trừu tượng hoá: tên của mỗi lớp có chuyển tải được các trừu tượng của nó không?

+ Sắp xếp trách nhiệm: tên, các dữ liệu và hàm có trách nhiệm chính trong mỗi lớp?

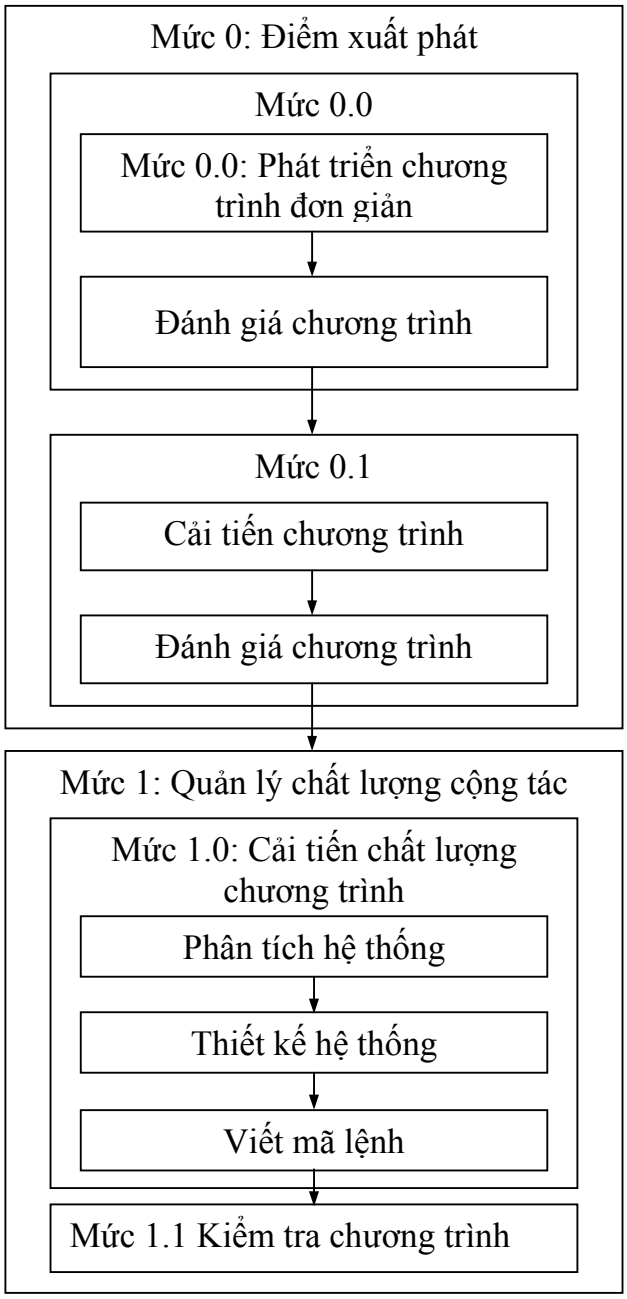
- Kiểm tra tất cả các trường hợp đặc biệt:

+ Đảm bảo thao tác thích hợp với các giá trị rỗng, đầy, nhỏ nhất, lớn nhất, âm, bằng không đối với tất cả các biến.

+ Giữ cho các điều kiện không bị vượt giới hạn.

+ Đảm bảo các điệu kiện không thể xảy ra là tuyệt đối không thể xảy ra.

+ Xử lý tất cả những điều kiện đầu vào không đúng.



Hình 3.1. Mô tả các bước trong quy trình ứng dụng XP trong CSP

3.3. ĐÁNH GIÁ SO SÁNH

3.3.1. So sánh với quy trình cộng tác phần mềm

Bảng 3.1

TT	Tiêu chí	Quy trình mới	CSP
1	Thời gian thực hiện chương trình	Nhanh	Chậm
2	Chất lượng chương trình	Tốt	Tốt
3	Tính mềm dẻo	Mềm dẻo	Mềm dẻo
4	Đáp ứng yêu cầu	Tốt	Tốt

3.3.2. So sánh với phương pháp lập trình linh hoạt

Bảng 3.2

TT	Tiêu chí	Quy trình mới	XP
1	Thời gian thực hiện chương trình	Nhanh	Nhanh
2	Chất lượng chong trình	Tốt	Đạt yêu cầu
3	Độ tin cậy người dụng	Tốt	Tốt

3.4. KẾT LUẬN

Việc áp dụng XP vào các mức của CSP cho ta một quy trình phát triển phần mềm mới, phát huy được các ưu điểm của cả XP và CSP. Quy trình này được ứng dụng để phát triển các phần mềm có kích thước lớn, với chất lượng cao trong một thời gian vừa phải, tạo được sự tin tưởng của khách hàng. Tuy nhiên, mỗi ứng dụng phần mềm đều có những đặc trưng riêng, nên trong quá trình phát triển, việc áp dụng các giá trị, các quy tắc, các thông lệ của XP đòi hỏi các lập trình viên phải hết sức linh hoạt.

Để chứng tỏ tính hiệu quả của quy trình này, trong chương 4 em trình bày các thử nghiệm áp dụng quy trình này vào việc phát triển một phần mềm ứng dụng.

Chương 4. THỬ NGHIỆM QUY TRÌNH TRONG ĐÀO TẠO VÀ TRONG PHÁT TRIỂN PHẦN MỀM

Với hai lập trình viên cùng thực hiện một nhiệm vụ, trên một máy tính và lợi điểm của phương pháp XP là các lập trình viên có thể giải quyết một bài toán với thời gian ngắn hơn và giải pháp cho bài toán là tối ưu hơn. Do vậy, chương trình được tạo ra có chất lượng cao hơn.

Để chứng tỏ hiệu quả của XP, dưới đây NVLV xin trình bày một số thử nghiệm ứng dụng quy trình vào một số lĩnh vực công nghệ thông tin.

4.1. THỬ NGHIỆM LẬP TRÌNH LINH HOẠT TRONG GIẢNG DẠY MÔN HỌC "LẬP TRÌNH TRÊN WINDOWS"

4.1.1. Giới thiệu nội dung và mục đích môn học

Khoa Công nghệ thông tin, Trường Đại học Công nghiệp Hà Nội, có giảng dạy môn học "Lập trình trên windows" cho sinh viên hệ cao đẳng, với ngôn ngữ lập trình là Visual Basic, và chủ yếu hướng vào việc tạo cho sinh viên các kỹ năng lập trình, và ứng dụng cho các bài toán quản lý, với hệ quản trị cơ sở dữ liệu Access.

Đề cương môn học (xem phụ lục)

4.1.1.1. Mục đích của môn học

- Giới thiệu các kiến thức về kỹ thuật lập trình trên windows.
- Trang bị các kiến thức về lập trình hướng đối tượng: thiết kế, cài đặt các đối tượng.
- Ứng dụng các kỹ thuật lập trình windows trong phát triển phần mềm.

4.1.1.2. Nội dung môn học

Môn học gồm có hai phần:

- Phần 1: Lý thuyết cơ bản
    + Giới thiệu về ngôn ngữ lập trình Visual Basic
    + Kỹ thuật lập trình hướng đối tượng bằng Visual Basic

+ Lập trình cơ sở dữ liệu

- Phần 2: Đồ án môn học, sinh viên thực hiện bài tập lớn

4.1.2. Phương pháp giảng dạy truyền thống

Lớp học gồm 36 sinh viên học trong một phòng máy tính, mỗi sinh viên sử dụng một máy tính. Quá trình giảng dạy được thực hiện như sau:

4.1.2.1. Phần 1: Lý thuyết cơ bản

- Giáo viên giới thiệu các kiến thức về lý thuyết bài học.
- Giao bài tập, giới thiệu mục đích và dự kiến thời gian hoàn thành.
- Hướng dẫn sinh viên phân tích, thiết kế các yêu cầu bài toán và cài đặt trên máy tính.
- Sinh viên giải quyết bài tập theo hướng dẫn của giáo viên.

Khi kết thúc phần lý thuyết cơ bản, giáo viên thực hiện một bài kiểm tra kiến thức của các sinh viên, để đánh giá kết quả học tập.

- Hình thức kiểm tra: thực hành, mỗi sinh viên thực hiện trên một máy tính.
- Thời gian: 90 phút

4.1.2.2. Phần 2: Đồ án môn học

Trong phần đồ án môn học, các sinh viên được giao thực hiện bài tập lớn, cài đặt mô phỏng một ứng dụng quản lý, trong thời gian 2 tuần.

Bài tập lớn được thực hiện dưới sự hướng dẫn của giáo viên như sau:

- Giáo viên giới thiệu mục đích của việc làm đồ án môn học.
- Chia lớp thành các nhóm từ 3-5 sinh viên.
- Mỗi nhóm nhận một đề tài (các đề tài cho mỗi nhóm có thể giống hoặc khác nhau và độ phức tạp tuỳ thuộc vào số lượng sinh viên).
- Giáo viên hướng dẫn cách thực hiện đề tài.
- Các nhóm sinh viên thảo luận để nắm bắt mục đích của đề tài và đưa ra kế hoạch thực hiện đề tài.

- Phân tích để xác định các yêu cầu của bài toán, chuyển các yêu cầu thành các nhiệm vụ cần phải thực hiện.

- Phân chia các nhiệm vụ cho mỗi thành viên trong nhóm.

- Mỗi thành viên nhận nhiệm vụ, thiết kế và cài đặt nhiệm vụ của mình.

- Nhóm sinh viên thảo luận để kết hợp các nhiệm vụ thành ứng dụng hoàn thiện.

- Báo cáo kết quả chương trình với giáo viên.

4.1.2.3. Nhận xét

a. Về phần lý thuyết cơ bản, thể hiện trong quá trình làm bài tập

Mỗi sinh viên mất nhiều thời gian để thực hiện một bài tập.

Thời gian hoàn thành bài tập không đều nhau giữa các sinh viên.

Hầu hết các câu hỏi đều tập trung vào giáo viên, nên giáo viên mất rất nhiều thời gian để giải thích.

Nhiều sinh viên đưa ra giải pháp không tốt, chương trình cài đặt còn mắc nhiều lỗi.

Kiến thức và các kỹ năng đạt được không đồng đều giữa các sinh viên.

Không phát huy nhiều tính tích cực học tập của sinh viên.

b. Về phần thực hiện bài tập lớn:

Có rất ít thảo luận giữa các sinh viên, ngoại trừ giai đoạn phân tích, và giai đoạn kết hợp các nhiệm vụ.

Các câu hỏi vẫn tập trung nhiều vào giáo viên.

Mỗi sinh viên chỉ hiểu được thiết kế và cách cài đặt nhiệm vụ của mình, nên sự trợ giúp giữa các sinh viên gặp khó khăn.

Thiếu sự thống nhất trong thiết kế và cài đặt, nên việc kết hợp nhiệm vụ rất khó khăn, và cần nhiều thời gian để điều chỉnh.

Nhiều giải pháp để giải quyết nhiệm vụ chưa tối ưu, thiết kế và đặt chưa tốt, dẫn đến chương trình chạy chậm, hoặc không giải quyết được triệt để yêu cầu của bài toán và còn mắc khá nhiều lỗi.

Tỷ lệ sinh viên không thể hoàn thành nhiệm vụ đúng thời hạn khá cao.

Cá biệt có những sinh viên không thể hoàn thành nhiệm vụ, làm ảnh hưởng đến kết quả của cả nhóm.

Kiến thức và các kỹ năng đạt được sau khi thực hiện xong bài tập lớn không đồng đều giữa các sinh viên.

4.1.3. Áp dụng phương pháp XP vào việc giảng dạy môn học "Lập trình trên windows"

Thí nghiệm được tiến hành với lớp học gồm 40 sinh viên, học trong một phòng máy tính. Phương pháp giảng dạy được tiến hành như sau:

4.1.3.1. Phần 1: Lý thuyết cơ bản

- Giáo viên chia lớp thành các nhóm, mỗi nhóm mỗi nhóm 2 sinh viên (cặp), ngồi chung một máy tính (các thành viên trong nhóm được luân chuyển trong mỗi buổi học).

- Giáo viên giới thiệu các kiến thức về lý thuyết trong bài học.

- Hướng dẫn sinh viên cách thức làm việc theo cặp (chỉ phải làm trong buổi học đầu tiên).

- Giao bài tập cho sinh viên, nêu mục đích của bài tập và dự kiến thời gian hoàn thành.

- Hướng dẫn sinh viên phân tích, thiết kế các yêu cầu bài toán và cài đặt trên máy tính.

- Sinh viên giải quyết bài tập theo hướng dẫn của giáo viên.

+ Hai sinh viên cùng phân tích và đưa ra giải pháp để giải quyết bài toán.

+ Cùng thảo luận để thiết kế thuật toán.

+ Một sinh viên thực hiện cài đặt, sinh viên còn lại ngồi quan sát kết quả do người kia tạo ra. Nếu có vướng mắc trong cài đặt, họ thảo luận để giải quyết, nếu không giải quyết được thì nhờ sự trợ giúp của giáo viên.

+ Sau khi cài đặt xong, cả hai cùng xem xét để cải tiến, tìm và sửa lỗi.

Khi kết thúc phần lý thuyết cơ bản, giáo viên thực hiện một bài kiểm tra kiến thức của các sinh viên, để đánh giá kết quả học tập.

- Hình thức kiểm tra: thực hành, mỗi sinh viên thực hiện trên một máy tính.

- Thời gian: 90 phút

4.1.3.2. Phần 2: Đồ án môn học

Kết thúc phần lý thuyết, sinh viên được giao thực hiện bài tập lớn, cài đặt mô phỏng một ứng dụng quản lý, trong thời gian 2 tuần.

Bài tập lớn được thực hiện dưới sự hướng dẫn của giáo viên như sau:

- Giáo viên giới thiệu mục đích của việc làm đồ án môn học.

- Chia lớp thành các nhóm, mỗi nhóm 4 hoặc 6 sinh viên.

- Mỗi nhóm nhận một đề tài (các đề tài cho mỗi nhóm có thể giống hoặc khác nhau và độ phức tạp tuỳ thuộc vào số lượng sinh viên mỗi nhóm).

- Giáo viên hướng dẫn cách thực hiện đề tài.

- Các nhóm sinh viên thảo luận để xác định mục đích của đề tài và đưa ra kế hoạch thực hiện đề tài.

- Phân tích để xác định các yêu cầu của bài toán, chuyển các yêu cầu thành các nhiệm vụ và dự kiến thời gian cần thiết để hoàn thành mỗi nhiệm vụ.

- Phân chia nhóm thành các cặp (2 hoặc 3 cặp).

- Giao các nhiệm vụ cho mỗi cặp (dựa vào thời gian dự kiến).

- Mỗi cặp nhận các nhiệm vụ, thảo luận và thiết kế nhiệm vụ của mình.

- Các cặp kết hợp kết quả thiết kế và thảo luận để có được thiết kế tổng thể của ứng dụng. Dự kiến thời gian cài đặt cho mỗi nhiệm vụ.

- Phân chia các nhiệm vụ cài đặt cho các cặp (thường là các nhiệm vụ do các cặp đã thiết kế).

- Các cặp thực hiện cài đặt: Hai người cùng làm trên một máy tính, một người cài đặt, một người quan sát. Hoặc có thể mỗi người thực hiện cài đặt một nhiệm vụ trên một máy, nếu là các nhiệm vụ đơn giản, để tiết kiệm thời gian.

- Các cặp xem xét lại kết quả cài đặt, tìm và sửa lỗi trong mã lệnh, hoặc cải tiến giao diện, mã lệnh để có chương trình chất lượng cao hơn.

- Nhóm sinh viên thảo luận để kết hợp các nhiệm vụ thành ứng dụng hoàn thiện.

- Báo cáo kết quả chương trình với giáo viên.

4.1.3.3. Nhận xét

Việc áp dụng phương pháp XP vào giảng dạy môn học "Lập trình trên windows" có các ưu điểm sau, so với phương pháp truyền thống:

a. Phần 1: Lý thuyết cơ bản

- Thời gian để hoàn thành bài tập ít hơn, và khá đồng đều giữa các nhóm. Nên số lượng các bài tập được giải quyết nhiều hơn.

- Nhiều vướng mắc được các sinh viên thảo luận và tự giải quyết, nên cần ít hơn sự trợ giúp của giáo viên.

- Giải pháp do các nhóm sinh viên đưa ra tốt hơn, chương trình ít lỗi hơn.

- Do có sự thảo luận và bổ sung kiến thức giữa các sinh viên, nên lượng kiến thức và kỹ năng mà các sinh viên đạt được nhiều hơn và đồng đều hơn giữa các sinh viên (một phần do sự luân chuyển sinh viên giữa các nhóm trong các buổi học).

- Số sinh viên không hoàn thành bài tập theo thời gian dự kiến ít hơn.

b. Phần 2: Đồ án môn học

- Các sinh viên trong nhóm thảo luận với nhau và kết hợp kết quả thường xuyên, nên tất cả đều hiểu được toàn bộ chương trình do họ tạo ra.
- Các nhóm cần ít hơn sự trợ giúp của giáo viên.
- Có sự thảo luận và thống nhất trong thiết kế và cài đặt, nên việc kết hợp các kết quả đơn giản và không cần nhiều thời gian để điều chỉnh.
- Các giải pháp để giải quyết nhiệm vụ tối ưu hơn, thiết kế và đặt tốt hơn, chương trình ít lỗi hơn.
- Tỷ lệ sinh viên không thể hoàn thành nhiệm vụ đúng thời hạn thấp, do phát huy được tính tích cực của các sinh viên.

c. So sánh một số kết quả định lượng giữa 2 phương pháp

* Phương pháp truyền thống, giảng dạy cho lớp CĐTin1-K5

- Số sinh viên: 36
- Thời gian: học kỳ 2, năm học 2004-2005
- Lớp được chia thành các nhóm làm đồ án môn học như sau:
  + Nhóm 5 sinh viên: 1 nhóm
  + Nhóm 4 sinh viên: 4 nhóm
  + Nhóm 3 sinh viên: 5 nhóm

* Phương pháp áp dụng XP, giảng dạy cho CĐTin1-K6

- Số sinh viên: 40
- Thời gian: học kỳ 2, năm học 2005-2006
- Lớp được chia thành các nhóm làm đồ án môn học như sau:
  + Nhóm 4 sinh viên: 4 nhóm
  + Nhóm 6 sinh viên: 4 nhóm

Bảng 4.1: So sánh tỷ lệ sinh viên hoàn thành bài tập trong thời gian quy định trong một buổi học (tính trung bình trong các buổi học)

Tỷ lệ sinh viên	P.P. truyền thống	Áp dụng XP
Hoàn thành bài tập trong thời gian quy định	64%	80%
Không hoàn thành bài tập trong thời gian quy định	36%	20%

Bảng 4.2: So sánh kết quả học tập phần lý thuyết cơ bản

Tỷ lệ sinh viên	P.P. Truyền thống	Áp dụng XP
Xuất sắc	7,5%	7,5%
Giỏi	12,5%	20%
Khá	35%	35%
Trung bình khá	25%	22,5%
Trung bình	15%	12,5%
Yếu	5%	2,5%

Bảng 4.3: So sánh kết quả đánh giá thực hiện bài tập lớn

Tỷ lệ sinh viên	Truyền thống	Áp dụng XP
Xuất sắc	15%	20%
Giỏi	30%	40%
Khá	35%	20%
Trung bình khá	12,5%	10%
Trung bình	7,5%	10%
Yếu	0%	0%

Qua các kết quả đánh giá trên, ta thấy được một số các ưu điểm khi áp dụng XP vào việc giảng dạy sau:

1. Số bài tập được thực hiện trong một buổi học nhiều hơn, kiến thức và các kỹ năng đạt được nhiều hơn. Bởi vì:

+ Hai người làm nhanh hơn một người

+ Việc thảo luận giúp các sinh viên bổ sung kiến thức cho nhau.

2. Giáo viên cần ít thời gian hướng dẫn hơn, do các sinh viên tự thảo luận để có thể giải quyết các vướng mắc.

3. Phát huy được tính tích cực của các sinh viên, vì khi làm theo cặp, các sinh viên buộc phải thực hiện để bắt kịp với cộng sự của mình.

4. Các kiến thức và các kỹ năng đạt được là đồng đều hơn giữa các sinh viên, do có sự bổ sung kiến thức giữa các sinh viên trong quá trình thảo luận để giải quyết bài toán.

5. Khi làm việc theo cặp, các sinh viên tin tưởng hơn vào công việc của mình, và có hứng thú hơn với việc học tập.

Kết luận: Áp dụng XP vào việc giảng dạy môn học "Lập trình trên windows" đã chứng tỏ được tính hiệu quả của XP trong lĩnh vực đào tạo kỹ năng lập trình.

4.2. THỬ NGHIỆM QUY TRÌNH ĐỂ PHÁT TRIỂN ỨNG DỤNG "QUẢN LÝ NHÂN SỰ" CHO CÔNG TY HỒNG HÀ

4.2.1. Giới thiệu hệ thống

Công ty Hồng Hà là doanh nghiệp nhà nước. Trụ sở chính của công ty tại số 15, Phường Quan Hoa - Cầu Giấy – Hà Nội.

Công ty Hồng Hà là một công ty sản xuất và phân phối thiết bị văn phòng và đồ dùng học tập.

Ngành kinh doanh của công ty gồm:

- Sản xuất kinh doanh vật liệu, trang thiết bị nghiên cứu, văn phòng.

- Kinh doanh sách và thiết bị học tập v.v...

Mặc dù trong quy trình nghiệp vụ, công ty đã sử dụng các trang thiết bị hiện đại. Xong việc quản lý nhân sự và lương nhân viên vẫn được thực hiện theo phương pháp thủ công và sự trợ giúp của các phần mềm đã có sẵn là Word và Excel. Điều này làm mất khá nhiều thời gian của nhân viên phòng tổ chức. Nay công ty muốn xây dựng một phần mềm thực sự để thực hiện việc quản lý nhân sự và lương nhân viên. Việc xây dựng phần mềm được giao cho một nhóm làm phần mềm gồm có 8 người. Thông thường, việc xây dựng một hệ thống tương tự nhóm lập trình mất khoảng thời gian là 3 tháng, nên nhóm dự kiến xây dựng phần mềm này với khoảng thời gian là 3 tháng, từ 25 tháng 5 đến 25 tháng 8 năm 2006. Đây là một nhóm lập trình chuyên nghiệp. Trước đây, khi nhận được yêu cầu xây dựng phần mềm, họ thường áp dụng phương pháp lập trình truyền thống, nghĩa là sau khi chuyển các yêu cầu của hệ thống thành các nhiệm vụ, thì mỗi (hoặc một số) nhiệm vụ được giao cho một người thực hiện. Sau khi mỗi người thực hiện xong nhiệm vụ của mình, họ kết hợp các kết quả lại với nhau, để có một chương trình thống nhất.

4.2.2. Phương pháp phát triển hệ thống

Nhóm phát triển phần mềm, khi nhận được yêu cầu xây dựng hệ thống phần mềm này, họ đã chọn phương án "ứng dụng XP trong CSP". Việc phân tích và thiết kế được thực hiện theo phương pháp hướng đối tượng (sử dụng các mô hình của UML), ngôn ngữ lập trình là Java, cơ sở dữ liệu Access.

Trong mỗi bước thực hiện, các lập trình viên đều đánh giá ban đầu, chẳng hạn, thời gian dự kiến, kích thước chương trình…, và ghi lại thời gian mà họ đã thực hiện mà họ đã sử dụng để hoàn thiện, ghi lại kích thước của chương trình thực sự, và các lỗi mà họ tìm thấy và loại bỏ. Trên cơ sở đó họ đánh giá hiệu quả của quá trình xây dựng hệ thống.

Với cách làm như vậy, họ đã đạt được những hiệu quả rất thiết thực.

Họ áp dụng quy trình và thực hiện xây dựng hệ thống như sau:

4.2.3. Xây dựng hệ thống

4.2.3.1. Xây dựng phần mềm mức đơn giản.

a. Bước 1: Lập kế hoạch từng bước

**** Tìm hiểu bài toán***

Bước này được thực hiện phối hợp bởi các thành viên trong nhóm và người dùng, nhằm tìm hiểu cơ cấu tổ chức của công ty và các yêu cầu mà hệ thống phải thực hiện.

Sau khi nhận được yêu cầu về việc xây dựng hệ thống quản lý, nhóm lập trình tiến hành trao đổi với người dùng để nhận định các yêu cầu của hệ thống. Và họ xác định được cơ cấu tổ chức của công ty, và các yêu cầu mà hệ thống cần phải thực hiện như sau:

Cơ cấu tổ chức của công ty gồm:

- Ban giám đốc.
- Các phòng chức năng.
- Các xí nghiệp sản xuất, các trung tâm.

Các yêu cầu hệ thống:

- Quản lý hồ sơ đầu vào
  + Các nguồn đăng ký nhân sự vào công ty.
  + Tiếp nhận hồ sơ nhân sự
- Quản lý thông tin nhân sự và phân công công tác.
  + Điều động công tác.
  + Chuyển công tác
  + Xét chế độ nghỉ việc
- Quản lý lương
  + Chấm công
  + Tính lương

**** Chuyển giao yêu cầu***

Tìm hiểu quy trình nghiệp vụ của hệ thống

Quy trình nghiệp vụ quản lý nhân sự và lương nhân viên như sau:

- Khi một người được nhận vào làm việc ở công ty, thông tin về nhân viên mới được lưu vào máy tính.

- Nhân viên mới nhận được quyết định phân công công việc (làm ở phòng ban nào).

- Nhân viên phải qua giai đoạn thử việc, thể hiện bởi một hợp đồng ngắn hạn giữa công ty với nhân viên.

- Sau giai đoạn thử việc, nếu được chấp nhận, công ty sẽ ký một hợp đồng dài hạn với nhân viên.

- Phòng quản lý nhân sự cần phải làm các công việc sau:

+ Quản lý thông tin nhân sự

+ Điều động công tác

+ Báo cáo tình hình biến đổi nhân sự hàng tháng lên ban giám đốc

+ Xét khen thưởng, kỷ luật, nâng, hạ lương đối với các nhân viên

+ Xét chế độ nghỉ việc cho nhân viên (nghỉ hưu, chuyển công tác, hoặc hết hợp đồng)

- Phòng tài vụ quản lý lương của nhân viên làm các công việc sau

+ Chấm công làm việc hàng ngày của nhân viên, theo từng tháng

+ Tính lương cho nhân viên hàng tháng

+ Báo cáo tình hình chi trả lương hàng tháng, hàng quý, hàng năm.

Hệ thống mới phải đáp ứng được các nhu cầu sau:

- Hệ thống chạy trên một mạng nội bộ của công ty.

- Ban giám đốc có thể truy nhập hệ thống để xem xét tình hình quản lý nhân sự và lương nhân viên.

- Cán bộ quản lý nhân sự duy trì thông tin nhân sự, điều động công tác và lập các báo cáo.

- Phòng tài vụ truy nhập hệ thống để chấm công, tính lương cho nhân viên, và lập các báo cáo.

- Các nhân viên được phép truy nhập hệ thống để xem lịch công tác và các thông tin về mình (thông tin cá nhân, thông tin về lương trong tháng…).

Chuyển các yêu cầu thành các chức năng

- Chức năng quản lý hồ sơ nhân sự và điều động công tác

+ Duy trì thông tin nhân sự (cán bộ quản lý nhân sự)

+ Điều động công tác

+ Lập báo về tình hình biến đổi nhân sự

+ Xét khen thưởng, kỷ luật, nâng, hạ lương đối với các nhân viên

+ Xét chế độ nghỉ việc cho nhân viên (nghỉ hưu, chuyển công tác, hoặc hết hợp đồng)

- Chức năng quản lý lương (cán bộ quản lý lý lương)

+ Lập bảng chấm công

+ Tính lương cho nhân viên hàng tháng

+ Lập báo cáo tình hình chi trả lương hàng tháng, hàng quý, hàng năm.

- Xem thông tin hệ thống (Ban giám đốc)

- Xem lịch công tác, thông tin cá nhân (nhân viên)

b. Bước2: Lặp lại kế hoạch

Bước này chuẩn bị các hoạt động và các nhiệm vụ cho các lập trình viên gồm 3 giai đoạn:

**** Giai đoạn tìm hiểu***

Chuyển các chức năng thành các nhiệm vụ, ghi các nhiệm vụ vào phiếu.

**** Giai đoạn chuyển giao***

Hình thành các cặp lập trình, phân công nhiệm vụ cho các cặp, đồng thời dự kiến thời gian để hoàn thành nhiệm vụ.

Nhóm lập trình chia thành 4 cặp và được giao các nhiệm vụ:

- Cặp thứ nhất: 'Duy trì thông tin nhân sự'
- Cặp thứ hai: 'Điều động công tác'
- Cặp thứ ba: 'Lập bảng chấm công' và 'Tính lương'
- Cặp thứ tư: 'Xem thông tin hệ thống' và 'Xem lịch công tác và thông tin nhân viên'.

**** Giai đoạn điều chỉnh***

- Các cặp thực hiện các nhiệm vụ được giao: thiết kế, viết mã lệnh, cải tiến mã lệnh.
- Biên dịch chương trình.
- Giao chương trình cho người dùng và nhận các thông tin phản hồi.

4.2.3.2. Đánh giá chương trình

- Đánh giá hiệu quả của chương trình: khả năng đáp ứng các yêu cầu của hệ thống.
- Đánh giá hiệu quả thực hiện: thời gian đã thực hiện so với dự kiến

Nhóm lập trình đánh giá hiệu quả của chương trình ban đầu, đánh giá thời gian mà họ đã sử dụng để hoàn thành nhiệm vụ của mình. Các lập trình viên thấy rằng họ đã thực hiện nhiệm vụ nhanh hơn so với thời gian dự kiến là 1 ngày, nghĩa là họ hoàn thành nhiệm vụ của mình trong 9 ngày. Nhóm lập trình cũng đánh giá những việc đã làm được và chưa làm được. Cụ thể, chương trình mới chỉ chạy trên máy đơn, chưa phải là một ứng dụng mạng,

nhóm lập trình cũng chưa tính đến cấu hình của hệ thống máy tính của công ty. Trong chương trình này, việc quản lý nhân viên và điều động công tác đã thực hiện được. Tuy nhiên, việc tính lương nhân viên còn chưa được hoàn chỉnh. Chương trình chưa thực hiện được việc tính bảo hiểm cho nhân viên.

4.2.3.3. Cải tiến chương trình

Sau khi có các đánh giá ban đầu về chương trình, các cặp lập trình bắt đầu thực hiện cải tiến các nhiệm vụ của họ, đề phù hợp với yêu cầu của hệ thống. Sau đó, chuyển giao chương trình cho công ty, trao đổi với người quản lý hệ thống để kiểm tra và đánh giá chương trình.

4.2.3.4. Xây dựng chương trình thực sự đáp ứng được các yêu cầu hệ thống

a. Cải tiến chương trình

Bước 1: Phân tích hệ thống (Sử dụng mô hình ca sử dụng của UML)

** Phát hiện các tác nhân của hệ thống*

Giai đoạn này nhóm lập trình làm việc kết hợp

Dựa vào các yêu cầu xác định ở phần 2.2.1, nhóm lập trình xác nhận các tác nhân của hệ thống.

- Cán bộ quản lý nhân sự, sử dụng hệ thống để quản thông tin nhân viên, quản lý quá trình công tác của nhân viên và làm các báo cáo về tình hình biến đổi nhân sự, quá trình công tác của nhân viên.

- Cán bộ tài vụ sử dụng hệ thống để chấm công và tính lương cho nhân viên, căn cứ vào thông tin nhân viên, tính bảo hiểm cho nhân viên, và làm các báo cáo.

- Ban giám đốc truy nhập hệ để xem các thông tin về hệ thống: kế hoạch thay đổi nhân sự, thông tin nhân viên, việc tính lương nhân viên, các khen thưởng, kỷ luật đối với nhân viên.

- Các nhân viên truy nhập hệ thống để xem lịch công tác, xem thông tin và bảng tính lương cá nhân.

** Nhận định các ca sử dụng*

Dựa vào các tác nhân và các yêu cầu của hệ thống, nhóm lập trình nhận định các ca sử dụng gồm.

- Duy trì thông tin nhân sự (phòng nhân sự)
- Điều động công tác (phòng nhân sự)
- Chấm công (phòng tài vụ)
- Tính lương (phòng tài vụ)
- Xem thông tin hệ thống (ban giám đốc)
- Xem thông tin nhân viên (nhân viên)

** Đặc tả các ca sử dụng*

Bước này nhóm lập trình viên chia thành các cặp, mỗi cặp nhận hai ca sử dụng để đặc tả.

- Cặp thứ nhất: 1 ca sử dụng, 'Nhập thông tin nhân sự'
- Cặp thứ hai: 1 ca sử dụng, 'Điều động công tác'
- Cặp thứ ba: 2 ca sử dụng, 'Chấm công' và 'Tính lương'
- Cặp thứ tư: 2 ca sử dụng, 'Xem thông tin hệ thống' và 'Xem thông tin nhân viên'

Các cặp tiến hành mô tả tóm tắt các ca sử dụng, sau đó mô tả các kịch bản của ca sử dụng và lập biểu đồ ca sử dụng.

** Phát hiện các lớp/đối tượng tham gia các ca sử dụng*

Cả 2 lập trình viên trong cặp xem xét các ca sử dụng và các kịch bản của ca sử dụng để xác định các lớp/đối tượng. Tên và nhiệm vụ của mỗi lớp được ghi vào một thẻ, gọi là thẻ CRC.

Khi hoàn thành việc phân tích, các cặp lập trình kết hợp các kết quả lại với nhau, xem xét lại toàn bộ quá trình phân tích để điều chỉnh lại cho phù hợp, rồi chuyển sang bước thiết kế.

Đánh giá hiệu quả của bước phân tích:

+ Thời gian dành cho việc phân tích nhiều hơn so với cách thức trước đây.

+ Hiệu quả đạt được tốt hơn, cụ thể, họ nắm bắt được hầu hết các yêu cầu của hệ thống, nhờ vậy họ phát hiện triệt để các ca sử dụng, và mô tả các ca sử dụng và các kịch bản của chúng thể hiện được yêu cầu của hệ thống. Nhờ sự hiệu quả của việc phân tích, nên khi kết hợp các kết quả phân tích giữa các cặp, nhóm lập trình chỉ cần điều chỉnh lại một chút cho phù hợp.

+ Phát hiện được đầy đủ các lớp/đối tượng cần thiết và nhiệm vụ của các lớp.

Bước 2: Thiết kế hệ thống

Cuối bước phân tích, nhóm lập trình phát hiện các lớp/đối tượng, tên các lớp và nhiệm vụ của chúng được ghi trên các thẻ.

Chuyển sang bước thiết kế:

+ Thiết kế các lớp, mỗi cặp nhận các thẻ có ghi tên lớp và nhiệm vụ của nó, và thực hiện thiết kế.

+ Thiết kế các liên kết, nhóm làm việc kết hợp, để xác định mối quan hệ giữa các lớp.

+ Thiết kế các thuộc tính và các thao tác của lớp. Nhóm lập trình làm việc theo cặp. Mỗi cặp thực hiện việc thiết kế cho một số lớp.

Sau khi việc thiết kế hoàn thành, các cặp lập trình kiểm tra lại các thiết kế, sửa các lỗi trong thiết kế của họ, hoặc cải tiến thiết kế nếu nó chưa phù hợp với yêu cầu, hoặc nó còn phức tạp, gây khó khăn cho việc cài đặt sau này.

Tiếp đó các cặp lập trình lại kết hợp các sản phẩm thiết kế của họ với nhau, để có được một bản thiết kế hoàn chỉnh của hệ thống.

Bước 3: Cài đặt

Mã lệnh chương trình được viết bằng ngôn ngữ lập trình Java, là một ngôn ngữ lập trình hướng đối tượng, phù hợp với phương pháp phân tích và thiết kế mà nhóm đã thực hiện.

Việc viết mã lệnh cũng được thực hiện theo cặp, mỗi cặp thực hiện việc cài đặt cho một số lớp. Các cặp lập trình không cần phải viết mã lệnh mới, mà chỉ cần sửa đổi và bổ sung mã lệnh mới trong cơ sở mã lệnh đã có từ trước. Đó là chương trình đơn giản mà họ đã viết để giao cho người sử dụng trong giai đoạn đầu tiên.

Bước 4: Cải tiến mã lệnh

Các cặp lập trình tiến hành xem xét lại toàn bộ thiết kế và mã lệnh do họ tạo ra. Sử dụng các kỹ thuật refatoring để sửa các lỗi nếu nó được phát hiện, hoặc cải tiến các thiết kế và mã lệnh chưa tốt, để có được thiết kế và mã lệnh có chất lượng tốt hơn.

b. Kiểm tra chương trình

- Kiểm tra tính hoàn thiện của mỗi bước
- Kiểm tra dữ liệu đầu vào, đầu ra và các chức năng cần thiết của hệ thống
- Kiểm tra các thành phần dữ liệu của các lớp, sự liên kết dữ liệu
- Kiểm tra các ràng buộc của hệ thống

4.2.3.5. Các kết quả đánh giá về thời gian thực hiện

a. Khi xây dựng chương trình đơn giản

- Tìm hiểu bài toán và lập kế hoạch
    + Dự kiến: 6 ngày
    + Thực hiện: 6 ngày
- Thiết kế

+ Dự kiến: 5 ngày

+ Thực hiện: 5 ngày (chậm 1 ngày)

- Viết mã lệnh và kiểm tra

+ Dự kiến: 5 ngày

+ Thực hiện: 4 ngày (nhanh 1 ngày)

b. Khi xây dựng ứng dụng thực sư

- Phân tích hệ thống

+ Dự kiến: 12 ngày

+ Thực hiện: 10 ngày (nhanh 2 ngày)

- Thiết kế hệ thống

+ Dự kiến: 19 ngày

+ Thực hiện: 16 ngày: (nhanh 1 ngày)

- Viết mã lệnh

+ Dự kiến: 14 ngày

+ Thực hiện: 12 ngày (nhanh 2 ngày)

- Kiểm thử

+ Dự kiến: 5 ngày

+ Thực hiện: 4 ngày (nhanh 1 ngày)

Qua các đánh giá trên, nhóm phần mềm thấy rằng, tổng thời gian thực hiện nhanh hơn so với thời gian dự kiến là 12%, nhưng thời gian này không nhiều. Nguyên nhân là nhóm lập trình mới áp dụng phương pháp này lần đầu tiên, nên trong quá trình làm việc theo cặp họ cần thời gian để điều chỉnh cho phù hợp. Ngoài ra làm việc theo cặp, hai người thực hiện một nhiệm vụ, cũng làm cho các lập trình viên mất nhiều thời gian hơn.

Thời gian viết mã lệnh và kiểm tra ít hơn so với dự kiến, bởi thiết kế được làm tốt hơn, và khi lập trình sử dụng kỹ năng của 2 người, nên mã lệnh ít lỗi (vì một người viết còn một người quan sát) và có chất lượng tốt hơn.

4.2.4. Đánh giá hiệu quả việc ứng dụng "Lập trình linh hoạt" trong "Quy trình cộng tác phần mềm"

Phần này so sánh thời gian thực hiện và chất lượng chương trình giữa hai ứng dụng: "Quản lý nhân sự" và "Quản lý kho hàng" được phát triển bởi cùng nhóm phần mềm.

Ứng dụng "Quản lý kho hàng" được xây dựng Công ty phân phối thiết bị máy tính và máy văn phòng Mạnh Trung, giúp công ty quản lý việc nhập và phân phối thiết bị cho khách hàng.

Trụ sở chính của Công ty: Số 20, Văn Cao - Hà Nội

Các yêu cầu hệ thống gồm:

Quản lý việc nhập thiết bị từ nơi cung ứng.

Quản lý việc phân phối thiết bị cho khách hàng

Thống kê lượng thiết bị đã phân phối và lượng thiết bị còn tồn kho theo định kỳ.

Thống kê và lập báo cáo tài chính theo định kỳ.

Thời gian xây dựng ứng dụng: từ 07/01/2006 đến 15/04/2006

Phương pháp: Áp dụng phương pháp truyền thống

Bảng 4.4: Tóm tắt phương pháp thực hiện và kết quả của các ứng dụng

TT		Quản lý nhân sự	Quản lý kho hàng
1	Thời gian thực hiện	25/05 đến 25/08/2006	07/01 đến 15/04/2006
2	Phương pháp	Ứng dụng XP trong CSP	Lập trình truyền thống
3	Phân tích thiết kế	Hướng đối tượng	Hướng chức năng
4	Cơ sở dữ liệu	Access	Access
5	Số bảng	17	14
6	Số form	20	21

Bảng 4.5: So sánh thời gian thực hiện

TT	Các bước thực hiện	Quản lý nhân sự	Quản lý kho
1	Viết chương trình mẫu	15 ngày	Không có
2	Phân tích	10 ngày	17 ngày
3	Thiết kế	16 ngày	25 ngày
4	Mã hoá	12 ngày	18 ngày
5	Kiểm thử	5 ngày	6 ngày
6	Tổng cộng	58 ngày	66 ngày

Bảng 4.6: So sánh chất lượng chương trình

TT	Tiêu chí	Quản lý nhân sự	Quản lý kho
1	Đáp ứng yêu cầu hệ thống	Tốt	Đạt yêu cầu
2	Bảo trì	Dễ	Khó
3	Thiết kế và mã lệnh	Chất lượng tốt	Đạt tiêu chuẩn
4	Lỗi chương trình	Ít hơn	Nhiều hơn

4.3. KẾT LUẬN

Qua việc so sánh kết quả thực nghiệm giữa hai ứng dụng, có thể đưa ra các nhận định sau về hiệu quả của việc ứng dụng XP trong CSP:

- ✓ Về thời gian thực hiện: Nhanh hơn so với phương pháp truyền thống.
- ✓ Về chất lượng chương trình
  - Chương trình được xây dựng có cấu trúc hợp lý và dễ sửa đổi.
  - Thiết kế và mã lệnh của chương trình có chất lượng tốt hơn.

- Phần mềm đáp ứng tốt các yêu cầu đặt ra.
- Chương trình ít lỗi hơn

✓ Đối với các lập trình viên:

- Họ tin tưởng vào công việc và có hứng thú hơn trong công việc của mình.
- Các kỹ năng của các lập trình viên tăng đáng kể.
- Việc trao đổi với người dùng được thực hiện thường xuyên. Vì vậy, các lập trình viên nắm bắt được hầu hết các yêu cầu của hệ thống, kể cả các yêu cầu nảy sinh trong quá trình thực hiện.
- Người dùng có thể hướng theo quá trình xây dựng hệ thống, nên họ tin tưởng vào tính khả thi của hệ thống được xây dựng.

TỔNG KẾT

Nội dung luận văn gồm 4 chương, nghiên cứu việc ứng dụng XP trong CSP để phát triển phần mềm. Đây là một vấn đề còn khá mới trong công nghệ phần mềm.

Tóm tắt luận văn:

- Tính cấp thiết của đề tài: Giải thích tại sao cần ứng dụng XP trong CSP để phát triển phần mềm.
- Chương 1: Nghiên cứu và trình bày các khái niệm, các quy tắc và các hoạt động trong XP. Tiếp đó trình bày tổng quan về CSP, các vấn đề liên quan đến CSP và mô hình mức tăng trưởng của CSP áp dụng trong phát triển phần mềm.
- Chương 2: Trình bày các "thông lệ" trong XP, đây là các nguyên tắc, các bước được thực hiện khi phát triển phần mềm theo XP. Xác định khả năng kết hợp XP và CSP.
- Chương 3. Đề xuất quy trình phát triển phần mềm theo CSP với việc ứng dụng các thông lệ của XP trong mô hình mức tăng trưởng của CSP. Quy trình này giúp phát triển nhanh một dự án phần mềm có quy mô lớn, với chất lượng cao và các yêu cầu thay đổi thường xuyên. Luận văn trình bày các bước trong quá trình phát triển phần mềm, theo các mức của CSP và việc áp dụng các thông lệ của XP trong mỗi mức nhằm giảm bớt thời gian thực hiện mà vẫn đạt chất lượng cao.
- Chương 4: Trình bày việc áp dụng XP và quy trình ứng dụng XP trong CSP, đồng thời chứng tỏ hiệu quả của chúng bằng hai thử nghiệm:

- ✓ Áp dụng XP trong giảng dạy môn học "Lập trình windows". Trình bày cách áp dụng XP trong giảng dạy, đánh giá các kết quả đạt được, so sánh với phương pháp truyền thống để chứng tỏ hiệu quả của XP.
- ✓ Phát triển phần mềm "Quản lý nhân sự": Mô tả hệ thống, trình bày các bước phát triển hệ thống theo mô hình ứng dụng XP trong CSP. Ghi nhận, đánh giá các kết quả được. So sánh với các kết quả đạt được của ứng dụng "Quản lý kho", phát triển theo phương pháp truyền thống. Từ đó đánh giá hiệu quả của quy trình đã đề xuất.

Đóng góp khoa học của luận văn:

- ➢ Luận văn nghiên cứu quy trình cộng tác phần mềm và phương pháp lập trình linh hoạt. Xác định khả năng ứng dụng lập trình linh hoạt trong đào tạo và ứng dụng lập trình linh hoạt trong quy trình cộng tác phần mềm.
- ➢ Đề xuất quy trình: Ứng dụng "Lập trình linh hoạt" trong "Quy trình cộng tác phần mềm". Định hướng quá trình phát triển phần mềm theo mô hình mức tăng trưởng của "Quy trình cộng tác phần mềm". Cách kết hợp các thông lệ của "Lập trình linh hoạt" trong mỗi mức.
- ➢ Ứng dụng thử nghiệm "Lập trình linh hoạt" trong đào tạo: Luận văn đã đưa ra phương pháp giảng dạy áp dụng "Lập trình linh hoạt" cho môn học "Lập trình trên windows". Đánh giá kết quả thực nghiệm, thấy được hiệu quả của phương pháp. Từ đó áp dụng phương pháp với các môn học khác, nhằm nâng cao chất lượng đào tạo trong lĩnh vực công nghệ thông tin nói chung.

- Thử nghiệm phát triển các dự án phần mềm: Mô tả ứng dụng "Lập trình linh hoạt" trong "Quy trình cộng tác phần mềm" để phát triển ứng dụng "Quản lý nhân sự". Luận văn cho thấy cách áp dụng và hiệu quả của việc áp dụng quy trình này trong các dự án phần mềm. Nhờ vậy, các tổ chức có thể áp dụng quy trình này để phát triển các ứng dụng của mình.

Hướng phát triển tiếp theo của đề tài:

- Hoàn thiện hơn nữa việc ứng dụng XP trong đào tạo và quy trình ứng dụng XP trong CSP.
- Thử nghiệm quy trình trên nhiều dự án phần mềm khác nhau để khẳng định hiệu quả. Từ đó ứng dụng quy trình vào thực tiến.
- Xây dựng mô hình và các phương pháp kiểm tra chất lượng phần mềm được phát triển theo quy trình này.

PHỤ LỤC

CHƯƠNG TRÌNH MÔN HỌC

LẬP TRÌNH TRÊN WINDOWS

I. NỘI DUNG TỔNG QUÁT VÀ PHÂN PHỐI THỜI GIAN

TT	*Tên chương*	*Thời gian*		
		Tổng số (tiết)	*Lý thuyết (tiết)*	*Thực hành (tiết)*
1	***Chương I: Giới thiệu chung về Visual Basic***	***1.5***	***3***	***3***
2	***Chương II: Đối tượng trong Visual Basic***	***8***	***5***	***6***
3	***Chương III: Các kiểu dữ liệu - hằng, biến***	***10.5***	***6***	***9***
4	***Chương IV: Các cấu trúc điều khiển - dữ liệu kiểu mảng***	***9***	***3***	***12***
5	***Chương V: Menu - thủ tục và hàm***	***7.5***	***3***	***9***
6	***Chương VI: Dùng form dạng MDI – Các CommonDialog***	***6***	***3***	***6***
7	***Chương VII: Lập trình với cơ sở dữ liệu***	***14.5***	***7***	***15***
	Tổng số	*60*	*30*	*60*

II. NỘI DUNG CHI TIẾT

CHƯƠNG I: GIỚI THIỆU CHUNG VỀ VISUAL BASIC

I. Giới thiệu Visual Basic

II. Các thành phần cơ bản của Visual Basic

III. Các bước xây dựng đề án bằng Visual Basic

CHƯƠNG II: ĐỐI TƯỢNG TRONG VISUAL BASIC

I. Giới thiệu chung về đối tượng
   1. Đặc điểm của các đối tượng trong chương trình
   2. Các đối tượng chính trong ToolBox
   3. Cách truy xuất đến mọi đối tượng
   4. Cách đặt tên cho các đối tượng

II. Thuộc tính của đối tượng - Sử dụng Properties Windows
   1. Khái niệm thuộc tính (Property)
   2. Dùng Properties Windows để thay đổi thuộc tính của đối tượng
   3. Một số thuộc tính cơ bản

III. Phương thức của đối tượng
   1. Khái niệm phương thức (Method)
   2. Một số phương thức cơ bản

IV. Sự kiện trên các đối tượng
   1. Khái niệm sự kiện và thủ tục đáp ứng sự kiện
   2. Một số sự kiện trên các đối tượng

V. Tìm hiểu một số đối tượng
   1. Form
   2. Label
   3. TextBox
   4. Command Button
   5. Frame
   6. CheckBox
   7. OptionButton
   8. ScrollBar
   9. Shape
   10. Line

TÀI LIỆU THAM KHẢO

[1] ExtremeProgramming.org home

[2] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*. Englewood Cliffs, NJ: Prentice Hall, 1991.

[3] W. W. Gibbs, "Software's Chronic Crisis," *Scientific American*, pp. 86-95, Sept.1994.

[4] S. L. Pfleeger, *Software Engineering: Theory and "thông lệ"*. Upper Saddle River, J:Prentice Hall, 1998.

[5] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, Massachusetts: Addison-Wesley, 2000.

[6] Wiki, "Programming In Pairs," in *Portland Pattern Repository*, http://c2.com/cgi/wiki?ProgrammingInPairs., 1999.

[7] J. T. Nosek, "The Case for Collaborative Programming," in *Communications of the ACM*, pp. 105-108, March 1998.

[8] W. S. Humphrey, *A Discipline for Software Engineering*: Addison Wesley Longman, Inc, 1995.

[9] P. Ferguson, W. S. Humphrey, S. Khajenoori, S. Macke, and A. Matvya, "Results of Applying the Personal Software Process," in *Computer*, pp. 24-31, May

1997.

[10] A. Anderson, Beattie, Ralph, Beck, Kent et al., "Chrysler Goes to "Extremes"," in *Distributed Computing*, pp. 24-28, Oct. 1998.

[11] Wiki, "Extreme Programming Roadmap," in *Portland Pattern Repository*, http://c2.com/cgi/wiki?ExtremeProgramming, 1999.

[12] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Reading, Massachusetts: Addison-Wesley, 1999.

[13] G. Salomon, *Distributed Cognitions: Psychological and educational onsiderations*. Cambridge: Cambridge University Press, 1993.

[14] N. V. Flor and E. L. Hutchins, "Analyzing Distributed Cognition in Software Teams: A Case Study of Team Programming During Perfective Software Maintenance," presented at Empirical Studies of Programmers: Fourth Workshop,

1991.

[15] M. C. Paulk, B. Curtis, and M. B. Chrisis, "Capability Maturity Model for Software Version 1.1," Software Engineering Institute CMU/SEI-93-TR, February 24,1993.

[16] W. Hayes and J. W. Over, "The Personal Software Process: An Empirical Study of the Impact of PSP on Individual Engineers," Software Engineering Institute, Pittsburgh, PA CMU/SEI-97-TR-001, December 1997.

[17] B. Meyer, *Object-Oriented Software Construction*, Second Edition ed. Upper Saddle River, New Jersey: Prentice Hall, 1997.

[18] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, *Object-Oriented Software Engineering: A Use Case Driven Approach*. Wokingham, England: Addison-Wesley, 1992.

[19] D. Rosenberg and K. Scott, *Use Case Driven Object Modeling with UML: A Practical Approach*. Reading, Massachusetts: Addison-Wesley, 1999.

[20] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*. Upper Saddle River, NJ: Prentice Hall, 2000.

[21] T. Quatrani, *Visual Modeling with Rational Rose and UML*. Reading, assachusetts: Addison Wesley, 1998.

[22] D. Bellin and S. S. Simone, *The CRC Card Book*. Reading, Massachusetts: Addison-Wesley, 1997.

[23] A. Cockburn, "Using CRC Cards," Humans and Technology TR.99.01, Salt Lake City, UT , March 11, 1999.

[24] Beck K., Extreme Programming Explained: Embrace Change. Addison-Wesley, 1999.

[25] Binder R., Testing Object-Oriented Systems: Models, Patterns, and Tools. Addison-Wesley, 1999.

[26] Martin Fowler's home page: patterns, refactoringg, extreme programming, unit testing, and UML material and links. http://www.martinfowler.com, 2000.

[27] Fowler M. et al., Cải tiến mã lệnh: Improving the Design of Existing Code. Addison-Wesley, 1999.

[28] Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[29] Java 2 SDK, Standard Edition Documentation. http://java.sun.com/j2se/1.3/docs/,2000.

[30] Junit-Testing Resources for Extreme Programming. http://www.junit.org, 2000.

[31] Opdyke W., Refactoring Object-Oriented Frameworks. Ph.D. diss., University of Illinois at Urbana-Champaign, http://st.cs.uiuc.edu/pub/papers/refactoring/opdykethesis. ps.Z, 1992.

[32] Refactoring Home Page. http://www.refactoring.com, 2000.

[33] V h ho M., Refactoring II. To be represented in seminar on Programming Paradigms, University of Helsinki, Department of Computer Science, 2000.

[34] XProgramming.com (extreme programming home page), http://www.XProgramming.com, 2000.

[35] L.A. Williams, The Collaborative Software Process, 2000.