TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN
----------o0o---------

Thạc Bình Cường

Bài giảng điện tử môn học

QUẢN LÝ DỰ ÁN PHẦN MỀM

Lời giới thiệu

Cuốn sách này trình bày cách tiếp cận tới những thế hệ thực hành về quản lý phần mềm. Rất nhiều tổ chức vẫn bám vào mô hình thác nước, thậm chí nó không được hoàn thiện lắm nhưng nó đưa ra được một hướng dẫn quản lý khá tỉ mỉ, cách để tiến hành để xử lý các tình trạng phần mềm đưa ra.

Trên thực tế khó đưa ra được một cách tiếp cận quản lý đầy đủ thích hợp với những vấn đề như là các vấn đề về tích hợp các thành phần thương mại, tái sử dụng phần mềm, quản lý rủi ro và các tiến trình phần mềm tăng trưởng xoáy chôn ốc. Cuốn sách này cung cấp một khung kiểm tra bằng các kinh nghiệm và tập các hướng dẫn để xử lý nó như thế nào?

Ông Walker Royce đã phát triển và kiểm tra cách tiếp cận quản lý phần mềm trong quá trình tham gia từ khảo sát sơ bộ đến phân phối sản phẩm phần mềm cho không lực của Mỹ.

Nền công nghiệp phần mềm đã hướng tới một phương pháp mới để quản lý độ phức tạp không ngừng tăng nhanh của các dự án phần mềm. Trước đây chúng ta đã từng thấy cuộc cách mạng, cuộc biến đổi và những vấn đề đang diễn ra kể cả thành công và thất bại. Trong khi những công nghệ các tiến trình và các phương pháp phần mềm đang phát triển một cách nhanh chóng thì kỹ nghệ phần mềm vẫn còn là một quá trình đòi hỏi sự nghiên cứu sâu sắc của con người.

Tài liệu này sẽ đề cập đến các nhận thức tổng quan về quản lý phần mềm và nhấn mạnh một cách nhìn cân đối những yếu tố sau:

- Lý thuyết và thực tiễn.
- Kỹ thuật của con người.
- Yêu cầu giá trị của khách hàng và lợi ích của nhà cung cấp.
- Chiến lược và sách lược.

Mặc dù vậy bạn cũng nên quan tâm đến một vấn đề quản lý quan trọng đó là sự điều chỉnh cân đối. Điều đặc biệt quan trọng là đạt tới các mục tiêu của các cổ đông khác nhau, người mà có giao tiếp với những người khác bằng những ngôn ngữ và các ký hiệu khác nhau. Đây là sự thúc đẩy những nhà sáng lập, một sự mô tả trừu tượng của hòn đá Rosetta. Ba ngôn ngữ biểu diễn cơ bản vốn có trong công nghệ phần mềm là với các yêu cầu (ngôn ngữ của không gian vấn đề), với thiết kế (ngôn ngữ chuyển dịch của kỹ sư phần mềm) và với cài đặt (ngôn ngữ thực hiện không gian vấn đề trên máy tính). Chỉ có những mốc như là những hòn đá Rosetta mới có thể chuyển dịch được các ký tự Hy Lạp, các kỹ thuật phần mềm có thể chuyển dịch những vấn đề thành các giải pháp mà nó thoả mãn tất cả các cổ đông.

Không có một cuốn sách chế biến nào cho quản lý phần mềm. Không có một công thức làm món ăn nào cho một thực tiễn rõ ràng. Tôi sẽ cố gắng tiếp cận đến các vấn đề một cách

khoa học hiện thực và kinh nghiệm nhất, nhưng việc quản lý là một vấn đề rất rộng trong việc đánh giá theo một nghĩa chung và quyết định phụ thuộc vào tình huống. Đó là điều tại sao mà các nhà quản lý được động viên.

Một vài chương bao gồm những phần thực dụng và thường được xử lý chặt chẽ trong các chủ đề cụ thể. Để phân biệt thế giới thực với các mô hình xử lý chung: các kỹ thuật và nguyên lý, thì phần đầu của mỗi một chương có từ thực dụng (pragmatic). Bởi nghĩa thực dụng có nghĩa là không có sự ảo tưởng và về mặt thực tiễn, nó là chính xác về ý nghĩa của những phần này. Chúng sẽ bao gồm những ý kiến mạnh và những vị trí khiêu khích và nó sẽ làm cho thần kinh của độc giả, những người bảo thủ trong một số thực hành, công cụ hoặc kỹ thuật lỗi thời hay quá hạn.

Tôi sẽ cố gắng để phân biệt trong các kỹ thuật đưa ra, những cách tiếp cận mới và những kỹ thuật lỗi thời bằng cách sử dụng những cách chứng minh phù hợp. Trong hầu hết các trường hợp tôi ủng hộ những quan điểm với những lí lẽ kinh tế đơn giản và nghĩa chung cùng với những kinh nghiệm vặt từ những ứng dụng. Rất nhiều những tư liệu giả thuyết đã rút ra từ cách quản lý những dự án thành công trên 10 năm qua (những vấn đề của thực tiễn). Mặt khác một số những tư liệu trình bày những vấn đề đã được chứng minh (những vấn đề của nghệ thuật), những cách tiếp cận về giả thuyết mà không có việc chứng minh rõ ràng trong thực tiễn. Chúng ta phải đấu tranh với một quan điểm của cuốn sách này được coi là giáo dục về quản lý hay là đào tạo về quản lý. Việc phân biệt này dường như là sự bới lông tìm vết nhưng nó rất quan trọng, một ví dụ là chúng ta hãy nghe việc minh hoạ sự khác nhau 15 năm trước đây. Giả sử rằng một bé gái của bạn từ trường về nhà vào một ngày nào đó và hỏi: "Thưa cha mẹ! Con có thể tham dự một khoá học về giáo dục giới tính ở trường được không?". Phản ứng của bạn hẳn là sẽ khác nếu như bé gái hỏi: " Con có thể tham dự một khoá huấn luyện về giới tính ở trường được không? " (điều này có nghĩa phần nào giúp tôi hiểu rằng con gái đã trưởng thành).

Quá trình đào tạo huấn luyện có một khía cạnh về tri thức ứng dụng mà làm cho tri thức hữu ích hoặc kém hữu ích hơn ngay lập tức. Mặt khác giáo dục là tập trung về việc giảng dạy các nguyên lý dựa vào kinh nghiệm và tinh thần của các mục tiêu với việc ứng dụng của những tri thức này dành cho người học. Tôi cố gắng để tập trung vào cuốn sách này như là một sự chuyển tải giáo dục quản lý. (Tôi không chắc chắn một điều rằng việc đào tạo quản lý khác với kinh nghiệm vừa học vừa làm). Tôi sẽ không ngụy biện rằng lời khuyên của tôi là có thể áp dụng được trực tiếp trên mọi dự án. Mặc dù tôi đã cố gắng chứng minh các quan điểm nếu có thể được, một số quan điểm sẽ không được chứng minh vì nó chỉ thuần tuý là giả thuyết. Tôi hy vọng rằng sự phỏng đoán và lời khuyên của tôi sẽ khuyến khích các cuộc tranh luận và sự tiến triển sau này.

Các bạn đọc của tôi đang thực hiện một bản gam (gamut) những thực hành chuyên môn về phần mềm. Các đọc giả chính là những người ra quyết định: những người có trách nhiệm đầu tư và chi phí về ngân sách phần mềm. Nhóm này bao gồm các nhà quản lý về tổ chức, những nhà quản lý về dự án, những nhân viên yêu cầu phần mềm và cán bộ của họ. Đối

với đọc giả này tôi sẽ cố gắng đưa ra các hướng dẫn có thể ứng dụng được trực tiếp đối với việc sử dụng các quyết định thực tế ngày nay và đầu tư chiến lược trong tương lai. Một loại đọc giả quan trọng khác là những người thực hành phần mềm mà họ thoả thuận và thực hiện kế hoạch, triển khai phần mềm trên những mục tiêu dự án và tổ chức.

Nội dung cách viết cuốn sách

Bởi vì tôi viết cho lượng lớn độc giả nên tôi đã không đi sâu vào chi tiết kỹ thuật hoặc những nguyên lý kỹ thuật, những vấn đề này được trình bày tốt hơn trong những cuốn sách khác. Thay vào đó tôi đưa ra một cách bàn luận khá sâu sắc về kinh tế, về mẫu quản lý, về những chiến lược phân chia công việc, về chiến lược tổ chức, những độ đo; đó là những điều cần thiết để xây dựng kế hoạch và thực hiện thành công một dự án phần mềm.

Có nhiều minh hoạ sẽ làm cho những chủ đề phức tạp trở nên dễ hiểu hơn. Sự chính xác và đúng đắn của các hình vẽ và các bảng là sự minh hoạ tốt nhất. Trong khi hầu hết các dữ liệu số mô tả chính xác một số khái niệm, xu hướng, kỳ vọng hoặc các quan hệ, thì các cách thức trình bày mang tính không chính xác vì mục đích. Trong khung cảnh quản lý phần mềm sự khác biệt giữa tính chính xác và tính đúng đắn là không đáng kể vì có thể từ hai lý do:

1. Quản lý phần mềm là những vùng đầy màu xám, nó phụ thuộc vào tình trạng và những trả giá nhập nhằng. Đó là sự khó khăn nếu không muốn nói là không thể chứng minh tính đúng đắn của nhiều khái niệm và giữ lại sự chính xác của cách trình bày trong một lĩnh vực rộng lớn.

2. Hiểu được sự khác nhau giữa chính xác và đúng đắn là kỹ năng cơ bản của những nhà quản lý phần mềm giỏi, người phải dự đoán một cách đúng đắn những ước lượng rủi ro và những ảnh hưởng của sự thay đổi. Độ chính xác không hiệu chỉnh trong các yêu cầu hoặc kế hoạch đã được chứng minh dù chưa rõ ràng, nhưng nó thường gây trở ngại tới thành công của dự án.

Trong rất nhiều cách biểu diễn số, các giá trị tuyệt đối thường là không quan trọng và hoàn toàn thay đổi trong các lĩnh vực và các tình huống dự án khác nhau. Các giá trị quan hệ của nó tạo nên hầu hết các hình vẽ và bảng biểu.

Nhân tiện tôi đưa ra những chứng cứ và kinh nghiệm thực tế để các nhà quản lý hướng tới những ngữ cảnh cụ thể, và liên hệ với những tiêu chuẩn đúng đắn và chính xác trong các điều kiện cụ thể. Một số phần phụ lục sẽ làm sáng tỏ các kỹ thuật được trình bày ở đây có thể đã được ứng dụng trên thực tế như thế nào. Một thí về hệ thống tàu đô đốc sẽ được nghiên cứu xuyên suốt trong tài liệu, đây là một dự án lớn và thành công, đã đưa ra một ví dụ cụ thể là làm thế nào có thể quản lý tốt được công việc. Nó cũng cung cấp một một khuôn khổ để hợp lý hoá một số tiến trình cải tiến và kỹ thuật.

Tổ chức

Cuốn sách được chia thành 5 phần , mỗi phần gồm một số chương:

Phần I, thời kỳ phục hưng của quản lý phần mềm. Phần này mô tả hiện trạng của nền kinh tế phần mềm và thực tiễn quản lý phần mềm và đưa ra sự chuyển dịch cần thiết đối với phần mềm được cải thiện về đầu tư.

Phần II, những khuôn khổ của quản lý phần mềm. Mô tả các nguyên lý về xử lý và khuôn khổ cho việc quản lý phần mềm tiên tiến bao gồm : các pha về vòng đời, pha về chế tạo thử, pha về dòng công việc, và các điểm kiểm tra.

Phần III, nguyên lý quản lý phần mềm. Phần này tóm tắt một vài kỹ thuật áp dụng cho lập kế hoạch, điều khiển và tự động hoá một quá trình phần mềm tiên tiến.

Phần IV, xu hướng phát triển. Các giả thuyết về các hiệu năng của dự án tiên tiến và nền kinh tế phần mềm trong thế hệ tới và bàn luận về sự dịch chuyển văn hoá cần thiết cho sự thành công.

Phần V, các ví dụ cụ thể và tài liệu tham khảo. Gồm 5 phụ lục, đưa ra những cái cơ bản cho việc chứng minh một vài nhận xét, chỉ dẫn và ý kiến được trình bày ở một vài nơi.

Chương 1

Quản lý phần mềm cổ truyền

Thời kì phục hưng của quản lý phần mềm

Nền công nghiệp phần mềm đã có một kinh nghiệm trong thời kì phục hưng. Rất nhiều những nguyên lý công nghệ phần mềm đã hằn sâu đang bị bó hẹp và lỗi thời bởi những kỹ thuật mới hoặc thay thế bằng những kỹ thuật tốt hơn hoặc mức độ tự động hoá cao hơn.

Cho dù nguyên lý nào đi chăng nữa thì điều quan trọng là người làm thực tế phải hiểu được trạng thái hiện tại trước khi biến đổi, chuyển dịch sang cái mới. Trước khi cân nhắc một khuôn khổ quản lý phần mềm cho tương lai thì cần thiết phải hiểu nền công nghiệp hiện nay đang ở đâu và làm sao có thể chiếm lĩnh được nó.

Trong 10 năm đã qua tôi đã tham gia và đóng góp để cải tiến các quá trình phần mềm của trên 500 công ty. Mục tiêu cụ thể của các đóng góp này là đạt được 2X, 3X, hoặc 10X tăng lên về năng suất, chất lượng, thời gian đối với thị trường hoặc tổ hợp của cả 3 điều trên. ở đây X là tương ứng với độ tốt lên của công ty đó giờ đây như thế nào. Một điều hài hước rằng rất nhiều các tổ chức này không có ý tưởng X là cái gì theo nghĩa mục tiêu.

Những chương trong phần I giới thiệu trạng thái thực tế trong nền công nghiệp phần mềm và xác định X trong các tiến trình quản lý phần mềm thông thường.

Điểm chính :

- Những thực tiễn quản lý phần mềm cổ truyền dường như chỉ là lý thuyết nhưng thực tiễn vẫn còn gắn chặt với công nghệ và kỹ thuật cổ xưa.
- Nền kinh tế phần mềm cổ truyền đưa ra những tiêu chuẩn về hiệu suất của các nguyên lý quản lý phần mềm cổ truyền.

Một điều tốt nhất về phần mềm đó là tính linh hoạt mềm dẻo: Nó có thể được lập trình để thực hiện hầu hết mọi việc. Điều tồi nhất về phần mềm cũng là tính linh hoạt mềm dẻo: các đặc tính "hầu như mọi thứ" rất khó trong lập kế hoạch, tiến độ và điều khiển sự phát triển phần mềm. Việc không dự đoán này là điều cơ bản của cuộc "khủng hoảng phần mềm" trên 30 năm nay.

Vào giữa những năm 1990 ít nhất có 3 phân tích quan trọng về nền công nghiệp kỹ nghệ phần mềm được thực hiện kết quả được công bố trong các ấn phẩm

1. Patterns of Software Systems Failure and Success (Jones, 1996).
2. Chaos (Standish Group , 1995).
3. Report of the Defense Science Board Task Force on Acquiring Defense Software Commercially (Defense Science Board, 1994).

Phụ lục A làm nổi bật một và kết quả có liên quan.

Tất cả 3 phân tích đó cùng đạt tới một kết luận chung: Mức độ thành công đối với dự án phần mềm là rất thấp. Mặc dù các phân tích này có một vài nhận thức khác nhau nhưng thông báo chủ yếu của họ được bổ sung cho nhau và rất kiên định. Chúng ta có thể tóm tắt như sau:

1. Việc phát triển phần mềm vẫn là cái không dự đoán được rất cao chỉ có khoảng 10% các dự án phần mềm được coi là thành công, với những ước lượng về ngân sách và tiến độ ban đầu.
2. Các nguyên lý về quản lý nặng về phán đoán thành công hay thất bại hơn là các tiến bộ về kỹ thuật.
3. Mức độ manh mún của phần mềm cũng như sự không kế thừa đã chỉ ra một tiến trình còn non nớt

Ba phân tích này đã giới thiệu cách quản lý các phần mềm và những tiêu chuẩn hiện tại đối với quá trình quản lý phần mềm cổ truyền. Có rất nhiều mảnh đất để phát triển.

Hãy nhớ những tóm tắt của các chương về khung tiến trình quản lý phần mềm mà hầu hết những phần mềm truyền thống đã được sử dụng. Trong khi những khuôn khổ mà chúng ta đã biết là mô hình thác nước có rất nhiều sự biến động đó là tiến trình vạch danh giới đối với hầu hết những kinh nghiệm của dự án phần mềm đã được tích luỹ cho tới ngày nay. Và trong

khi sự lo ngại đang phát sinh thì điều quan trọng được đặt ra là môi trường tốt cho các kỹ thuật cải tiến tiến trình sẽ được thảo luận trong suốt cuốn sách này.

1.1.Mô hình thác nước

Hầu hết nội dung công nghệ phần mềm trình bày theo mô hình thác nước coi như là nguồn gốc của tiến trình phần mềm truyền thống. Chú ý rằng nó sẽ là tiêu chuẩn hơn quá trình đó. Phần này sẽ xem xét và đánh giá mô hình thác nước, sau đó xem nền công nghiệp đã được thực hành tiến trình phần mềm truyền thống như thế nào? Trên thực tế mặc dù nền công nghiệp này đã bỏ qua rất nhiều phần lý thuyết, nó vẫn còn được quản lý để mở ra nhiều thực hành tốt (và một vài thực tiễn không tốt lắm) đặc biệt khi nó sử dụng các kỹ thuật tiên tiến.

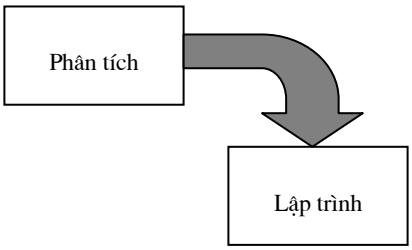
1.1.1. Lý thuyết

Vào năm 1970 cha tôi ông Winston Royce đã đưa ra một bài báo với tiêu đề " Quản lý việc phát triển hệ thống phần mềm lớn" trên tạp chí IEEE WESCON (Royce, Winston, 1970) bài báo này dựa và các bài giảng về quản lý các dự án phần mềm lớn mà nó còn giữ lại gốc của mô hình thác nước. Nó đã đưa ra một tóm tắt ngắn gọn và sáng sủa về tính triết học của quản lý phần mềm truyền thống trong khoảng những năm 1970 và hầu như những lời khuyên trong 30 năm qua đã được thời gian kiểm nghiệm trước tốc độ thay đổi của công nghệ.

Bài báo này đã đưa ra 3 luận điểm quan trọng (Phần để trong dấu nháy và các đoạn được in nghiêng).

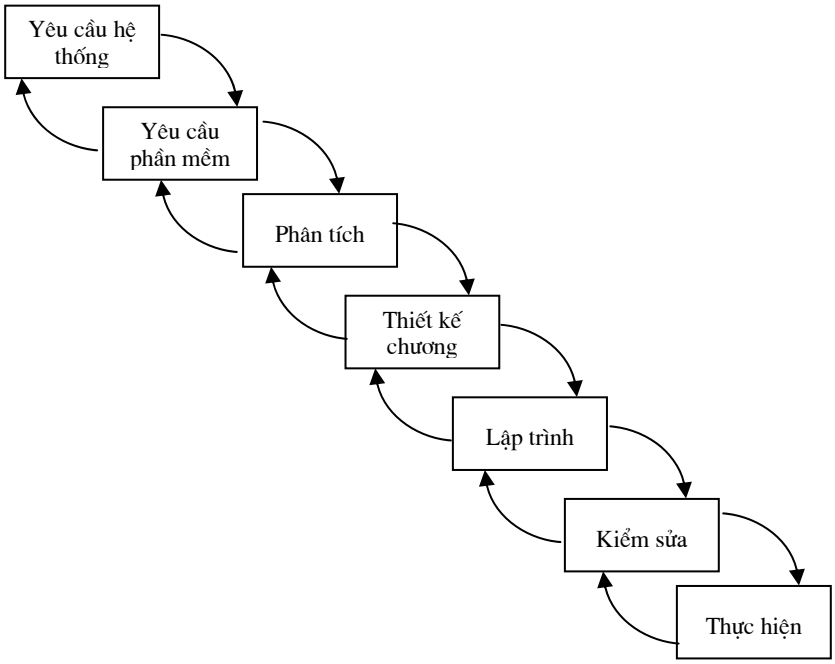
1. *Có hai bước cần thiết để phát triển một chương trình máy tính: phân tích và lập trình.*
2. *Để quản lý và điều khiển tất cả những sự tự do sáng tạo với phát triển phần mềm người ta sẽ giới thiệu một vài bước "ở phía trước (overhead)" , gồm xác định các yêu cầu của hệ thống, xác định yêu cầu phần mềm, thiết kế chương trình và kiểm sửa. Những bước này bổ sung cho các bước phân tích và lập trình. Hình 1.1 sẽ minh hoạ sơ thảo dự án đưa ra và những bước cơ bản trong việc phát triển một chương trình quy mô lớn.*
3. *Khuôn khổ cơ bản đã mô tả trong mô hình thác nước sẽ có những rủi ro và những sai sót. Giai đoạn kiểm thử xuất hiện tại cuối của vòng phát triển mà đầu tiên là thời gian, bộ nhớ, truyền vào ra... là những kinh nghiệm khi phân biệt từ bước phân tích. Sự thay đổi của các thiết kế đưa ra hầu như nó sẽ phá vỡ tất cả các yêu cầu phần mềm khi mà việc thiết kế dựa vào các yêu cầu bị phá huỷ. Hoặc là các yêu cầu này phải thay đổi hoặc phần thay đổi thiết kế trọng yếu phải được bảo hành.*

Phần 1 của mô hình thác nước: Hai bước cơ bản để xây dựng một chương trình.



Phân tích và lập trình sẽ bao gồm các công việc sáng tạo mà nó đóng góp trực tiếp tới tính hữu dụng của sản phẩm.

Phần 2 của mô hình thác nước: Cách tiếp cận của hệ thống lớn



Phần 3 của mô hình thác nước: Năm sự cải tiến cần thiết để tiếp cận công việc.

1. Hoàn thiện thiết kế chương trình trước khi phân tích và viết chương trình.
2. Bảo trì hiện hành và hoàn thiện tài liệu.
3. Thực hiện công việc hai lần nếu có thể.
4. Lập kế hoạch, điều khiển và điều hành kiểm sửa.
5. Trao đổi và thu hút khách hàng.

Hình 1-1. *Mô hình thác nước*

Mục 1, dường như quan trọng, sau này nó sẽ được mở rộng thành một trong những chủ đề quản lý toàn bộ : Sự phân chia giai đoạn công nghệ từ giai đoạn sản phẩm.

Bẩy trong chín trang của bài báo để dành cho mô tả 5 bước phát triển tiến trình thác nước cơ bản mà nó sẽ loại bỏ đi hầu hết những rủi ro được nói đến trong mục 3. Năm sự cải

tiến được trình bày tiếp sau. (Phần để trong dấu nháy và những đoạn được in nghiêng, kèm theo đó là những nhận xét của tôi về những công nghệ và thuật ngữ ngày nay).

1. ***Đầu tiên là giai đoạn thiết kế chương trình.*** *Việc đầu tiên để giải quyết vấn đề là bổ sung một thiết kế chương trình sơ bộ vào giữa giai đoạn xác định yêu cầu phần mềm và giai đoạn phân tích. Bằng kỹ thuật này, các nhà thiết kế chương trình qủa quyết rằng phần mềm sẽ không bị sai vì bộ nhớ, thời gian, và sự thay đổi dữ liệu. Khi phân tích được tiến hành trong giai đoạn tiếp theo thì người thiết kế chương trình phải tác động với các nhà phân tích các hạn chế về bộ nhớ, thời gian và tác nghiệp theo cách mà anh ta cảm nhận thấy. Nếu như tất cả các nguồn tài nguyên sẽ dùng đến không đủ đáp ứng hoặc những thiết kế về tác nghiệp bị sai sót thì nó sẽ được phát hiện tại trạng thái sớm hơn và việc lặp lại các yêu cầu và thiết kế sơ bộ có thể được lặp lại trước khi thiết kế, lập trình và kiểm sửa. Làm thế nào để thủ tục thiết kế chương trình được thực hiện. Nó đòi hỏi các bước sau đây:*

Bắt đầu quá trình thiết kế với các nhà thiết kế không phải các nhà phân tích hoặc các nhà lập trình.

Thiết kế , định nghĩa và xác định chế độ xử lý dữ liệu thậm chí cả các rủi ro. Chỉ định các chức năng xử lý, thiết kế cơ sở dữ liệu xác định thời gian thực hiện, xác định giao diện và chế độ xử lý với hệ điều hành, mô tả quá trình xử lý vào ra và xác định các thủ tục thao tác sơ bộ.

Viết một tài liệu tổng quan rễ đọc, dễ hiểu , đầy đủ thông tin và mang tính thời sự để cho mọi người tham gia dự án có thể nắm bắt được những nét cơ bản về hệ thống.

- Bản chất của quá trình xử lý mà tôi trình bày trong những chương sau là sự phát triển đầu tiên về kiến trúc. Mặc dù một vài thuật ngữ có thể thay đổi (chẳng hạn như từ kiến trúc có thể được thay thế bằng thiết kế chương trình), nhưng bản chất của các quá trình tiên tiến luôn phù hợp với việc giải thích đưa ra ở đây. Như sự mô tả sau này thì kiến trúc sẽ được làm trước, và nó sẽ được thiết kế và phát triển song song với việc lập kế hoạch và xác định yêu cầu như là một bộ phận của một giai đoạn công nghệ trong một dự án.

2. ***Lập tài liệu thiết kế.*** *Toàn bộ số tài liệu yêu cầu về những chương trình phần mềm là rất lớn, chắc chắn nó phải nhiều hơn những tài liệu do những người lập trình, những người phân tích hoặc những người thiết kế chương trình đưa ra. Tại sao chúng ta phải làm nhiều tài liệu như vậy? (1).Mỗi một người thiết kế phải trao đổi với những nhà thiết kế khác, những nhà quản lý và thậm trí với cả những khách hàng.(2). Ngay trong những giai đoạn ban đầu thì tài liệu cũng là một thiết kế. (3). Giá trị bằng tiền thực tế của các tài liệu cũng hỗ trợ việc sửa đổi sau này do một nhóm kiểm sửa độc lập, do một nhóm bảo trì độc lập và do những cá nhân không có kiến thức về phần mềm thực hiện.*

- Nếu như chúng ta lờ bỏ đi sự thiếu hụt không tương thích về kỹ thuật trong một khung thời gian mà tài liệu được viết thì thực chất thông điệp của "lập tài liệu cho thiết kế " vẫn còn giá trị. Việc trình bày một cách dễ hiểu các khuôn mẫu mà các cổ đông và các nhóm có thể truy xuất được là điều cốt yếu. Tuy nhiên ưu điểm chính trong các ký hiệu, ngôn ngữ, cách duyệt, công cụ và phương pháp đã đáp lại những yêu cầu đối với những sự lạc hậu về tài liệu. Trong chương sau , tôi chỉ rõ ràng rằng nếu tập trung quá nhiều về tài liệu thì sẽ không tốt và phản tác dụng. Bởi vì các công nghệ hiện nay dã hỗ trợ cho cách biểu diễn những ký hiệu của tài liệu rất chính xác để xác định yêu cầu, thiết kế và thể hiện.

3. Làm hai lần. *Nếu như một chương trình máy tính được phát triển lần đầu tiên thì việc chỉnh lý làm ra phiên bản cuối cùng cấp phát cho khách hàng để triển khai thực hiện thực sự là phiên bản thứ hai mà đã được đánh giá và thực hiện. Chú ý rằng đây là một sự đơn giản của toàn bộ quá trình được thực hiện thu nhỏ lại, về mặt thời gian điều này là rất nhỏ theo khía cạnh của toàn bộ sự nỗ lực. Trong phiên bản đầu tiên, toàn đội phải có một nỗ lực đặt biệt mà họ có thể nhanh chóng cảm nhận được các điểm trục trặc trong thiết kế, trong mô hình, sự lựa chọn mô hình, quên đi những khía cạnh trực diện của thiết kế mà không có giá trị nghiên cứu tại điểm khởi đầu và cuối cùng thu được một chương trình không còn lỗi nữa.*

- Đây là một cách mô tả súc tích và ngắn gọn sự phát triển kiến trúc đầu tiên, mà trong đó nhóm kiến trúc phải chịu trách nhiệm về những công nghệ ban đầu. Bằng cách tạo ra một thực tiễn, mà sau này tôi sẽ làm, đưa ra một cách tiếp cận "làm N lần", đó là nguyên tắc cơ bản của sự phát triển lặp tiên tiến ngày nay.

Người quản lý dự án phải có óc phán đoán nếu không có giai đoạn đầu tiên này. Với một bước mô phỏng đầu tiên, ở mức kiểm sửa kinh nghiệm về các giả thiết và các phạm vi những cái mà do con người phán đoán trong các lĩnh vực thiết kế chương trình máy tính (như là việc ước lượng về trọng số nhại lại, chi phí hoàn thành hoặc những gấp bội hàng ngày) là những cái thường xảy ra và cái tối ưu trầm trọng.

- Đây là sự mô tả rất quan trọng trên tinh thần của sự phát triển tuần hoàn và những thuận lợi cố hữu cho quản lý rủi ro.

4. Lập kế hoạch, điều khiển và kiểm tra chất lượng. *Không có đòi hỏi, người dùng lớn nhất của nguồn nhân lực của dự án, thời gian (xử lý) máy tính và /hoặc đánh giá quản lý là pha kiểm tra. Đây là pha rủi ro lớn nhất trong kì giá trị và lập lịch, khi cách lưu trữ lại là giá trị tối thiểu sẵn có, nếu trong mọi trường hợp. Ba điều giới thiệu trước đây tất cả tập trung vào việc khám phá và giải quyết các vấn đề trước khi đi vào pha kiểm tra. Tuy nhiên, thậm chí sau khi được thực hiện những điều đó, vẫn còn pha kiểm tra và vẫn có nhiều điều quan trọng cần được làm, bao gồm: (1) việc làm của đội ngũ kiểm tra những người mà không chịu trách nhiệm về thiết kế ban đầu; (2) công việc*

kiểm định trực quan để đánh dấu những lỗi rõ ràng như là rơi xuống dấu âm, thiếu hai nhân tố, nhảy tới các địa chỉ sai sót (không sử dụng máy tính để dò tìm lỗi này, nó quá đắt); (3) kiểm tra các đường dẫn logic; (4) công việc kiểm tra cuối cùng trên các máy đích.

- ở đây chúng ta có vài lời khuyên tốt và một vài lời khuyên lỗi thời, các mục 1 và 4 vẫn là những lời khuyên tốt, nó được thảo luận kỹ lưỡng trong các chương sau. Mục 2 vẫn chắc chắn là một cách thích thú kì cục phổ biến (sử dụng các phần mềm kiểm tra), nhưng mục đích của nó như đã trình bày ở đây hầu như đã lỗi thời. Mặc dù có thể nó đã là một sản phẩm có giá trị hiệu quả thực hiện trong kỹ thuật của những năm 70, nhưng nó không phù hợp với ngày nay. Các máy tính, các bộ diễn dịch , bộ phân tích và những công cụ khác đã là những máy móc có hiệu suất cao hơn để bắt kịp các lỗi rõ ràng. Như ở mục 3, việc kiểm tra các đường dẫn logic rất khó đầy đủ trong những năm 70, nếu không có việc thêm vào các phần tử phân phối phức tạp, các phần tử dùng lại được và một vài nhân tố phức tạp khác. Nó chắc chắn không khả thi với hầu hết các hệ thống ngày nay. Đây là điều đặc biệt đúng với các phân phối việc tính toán, trong đó, với thời gian như một hướng thêm vào, đó là một số vô tận những đường dẫn logic. Trong một xử lý tiên tiến, việc kiểm tra là một vòng đời hoạt động khi mà việc thực hiện đúng đắn các yêu cầu ít hơn tổng số tài nguyên và những khám phá phát hiện ra còn dễ dàng hơn trong vòng đời , khi lưu trữ lại vẫn có thể được sử dụng.

5. Thu hút khách hàng. *Có một vài lý do, một thiết kế phần mềm nào đó sẽ được làm là một chủ đề được diễn giải rộng rãi, thậm chí sau cả hợp đồng trước đó. Đó là điều quan trọng để thu hút khách hàng trong một cách thức hình thức vì vậy khách hàng đã chuển giao lại cho chính họ những điểm dễ hơn trước khi giao hàng cuối cùng. Có ba điểm sau đây, các yêu cầu được định nghĩa là sự hiểu thấu bên trong sự vật (insight), sự phán đoán và sự tận tình (commitment) của khách hàng có thể ủng hộ sự nỗ lực phát triển. Nó bao hàm việc "xem xét lại phần mềm sơ thảo" sau bước thiết kế chương trình sơ thảo, một tuần tự "xem xét lại phần mềm thiết kế tới hạn" trong suốt chương trình thiết kế và một "xem xét lại phần mềm chấp nhận cuối cùng" sau đó kiểm thử.*

- Sự hiểu thấu bên trong sự vật này đã được theo đuổi trong nhiều năm và những nơi được thực hiện đã sản xuất cho những kết quả đáng tin cậy. Lôi kéo khách hàng với những luận chứng dễ dàng và kế hoạch giải phóng anpha / beta là đã được chứng minh, một kỹ thuật có giá trị.

Tôi đã luôn nhấn mạnh (lấn át bằng) sự thấu hiểu bản chất đã được trình bày trên trang giấy này. Trong khi hầu hết công nghệ đã sử dụng nguồn năng lượng đập vào được coi như gần với mô hình thác nước, tôi chỉ thấy những lỗi nhỏ trong lý thuyết thậm chí khi nó đã được áp dụng trong hoàn cảnh của công nghệ hiện nay. Sự phê phán sẽ là mục tiêu trong thực hành cách tiếp cận, nơi kết hợp các giá trị không tốt khác nhau với những yếu tố không thể thực hiện

được. Tôi nghi ngờ rằng hầu hết những người phê phán chưa bao giờ thực sự hiểu được lý thuyết này; họ mới chỉ hiểu phần thực hành định trước.

Trong suốt cuốn sách này, tôi tham khảo vấn đề thực hành trong quá khứ và hiện tại gần với mô hình thác nước, sẽ tiếp tục được thảo luận, như "quy ước (conventional)" tiếp cận hay xử lý phần mềm quản lý. Tôi chứng tỏ rằng nó không dài hơn một khung làm việc tốt cho kỹ nghệ phần mềm hiện đại về mặt thực hành và và kỹ thuật, và tôi sử dụng nó như là một tiêu chuẩn thực sự để hợp lý hoá một cải tiến xử lý mà loại bỏ đi một vài sai sót cơ bản của nó.

1.1.2. Trong thực hành

Mặc dù lời khuyên của nhiều chuyên gia phần mềm và lý thuyết sau mô hình thác nước, nhưng một vài dự án phần mềm vẫn thực hiện gần giống với quản lý phần mềm truyền thống. Tuy nhiên, bởi vì sử dụng của nó đang tàn tạ và là nhiều điều phổ biến hơn trong quá khứ, tôi chứng minh nó là một thời quá khứ đã qua.

Điều hữu ích để tóm tắt đặc điểm của xử lý truyền thống như là đặc tính trưng đã được áp dụng, điều mà không cần thiết như nó đã là một ý định. Các dự án thường xuyên có các phiền hà thể hiện ra ở các triệu chứng sau đây:

- Kéo dài sự tích hợp và điểm gãy thiết kế muộn .
- Sự phân tích rủi ro muộn.
- Các yêu cầu điều khiển phân rã (phân huỷ) chức năng.
- Các quan hệ đối thủ đặt cược (người giữ tiền đặt cược).
- Chủ điểm trong tài liệu và xem lại gặp gỡ.

Kéo dài sự tích hợp và điểm gãy thiết kế muộn.

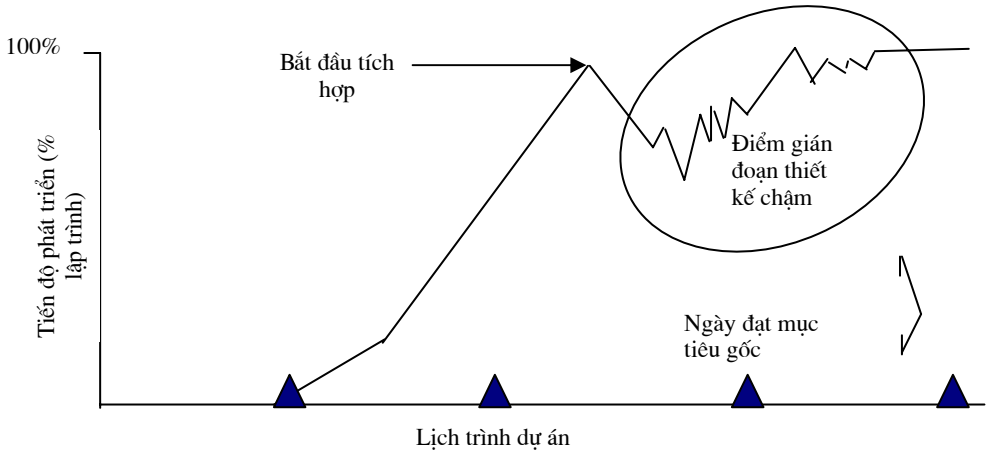
Cho một điển hình phát triển dự án là sử dụng một mô hình thác nước để quản lý tiến trình, Hình 1-2 minh hoạ sự phát triển tiến triển gắn với (đấu với) thời gian. Sự tiến triển đã được định nghĩa bằng phần trăm chương trình, đó là, có thể giải thích được trên mẫu biểu đích của nó. (Phần mềm có thể dịch được và có thể thực hiện (chạy) được; nó không nhất thiết cần đầy đủ, tương đối dễ dãi, cũng không cần chỉ định rõ ràng). Tuần tự sau đây là (sự tiến triển) chung:

- Sớm thành công qua những thiết kế trên giấy và những chỉ dẫn rất đầy đủ, tường tận (thường quá tường tận).
- Sự tận tình để mã hoá muộn (bổ sung) trong vòng đời.
- Sự phân tích các rủi ro (nguy cơ) phải trả giá đến các thực hiện bất ngờ phát sinh và những sự nhập nhằng giữa các mặt chung.
- Bao nặng và sức ép lập lịch sẽ cho hệ thống làm việc.

- Sắp xếp lại các thiết kế muộn không tối ưu, nếu không có thời gian để thiết kế lại.
- Một sản phẩm yếu ớt, không thể giữ được đã được phát ra muộn.

Dạng	Văn bản không theo thể thức	Sơ đồ luồng	Chương trình nguồn	Vạch ranh giới cấu hình
Hoạt động	Yêu cầu phân tích	Thiết kế chương trình	Lập trình và kiểm thử	Tích hợp theo tỉ lệ mở rộng và kiểm sửa
Sản phẩm	Tài liệu	Tài liệu	Chương trình	Vạch ranh giới yếu ớt

Các hoạt động kế tiếp: yêu cầu - thiết kế - lập trình – tích hợp - kiểm sửa



Hình 1-2. *Tiến trình sơ thảo của một dự án phần mềm cổ truyền.*

Bảng 1-1. Phí tổn cho các hoạt động của một dự án phần mềm.

Hoạt động	Chi phí
Quản lý	5%
Yêu cầu	5%
Thiết kế	10%
Lập trình và kiểm tra	30%
Tích hợp và kiểm sửa	40%
Triển khai	5%
Môi trường	5%
Tổng	**100%**

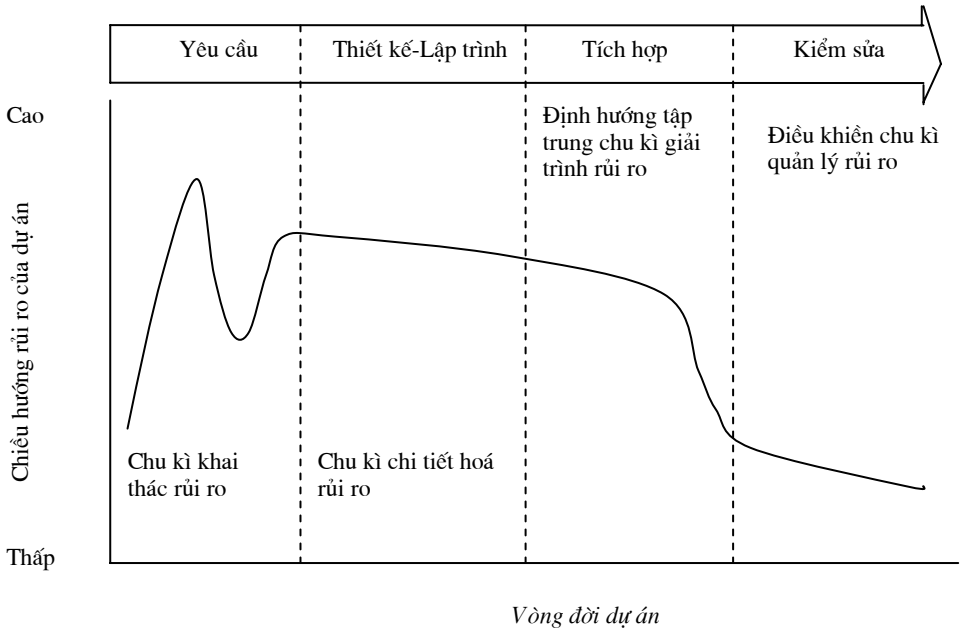
Dựa trên ngôn ngữ và kĩ thuật chưa chín muồi được sử dụng trong cách tiếp cận truyền thống, đã có tầm quan trọng đáng kể trong sự hoàn thành "phần mềm thiết kế " trước khi chuyển nó sang ngôn ngữ lập trình mục đích, ở đó nó sẽ rất khó hiểu và thay đổi. Thực hành này đã cho kết quả trong sử dụng nhiều khuôn mẫu (các yêu cầu bằng tiếng Anh , thiết kế sơ bộ trong các sơ đồ luồng, các thiết kế chi tiết trong ngôn ngữ thiết kế chương trình và việc thực hiện đầy đủ trong ngôn ngữ mục đích chẳng hạn như FORTRAN, COBOL, hoặc C) và những sự dịch chuyển giữa lỗi dễ xảy ra, lao động chuyên sâu và các định dạng.

Các kỹ thuật truyền thống được áp dụng vào cho mô hình thác nước trong tiến trình thiết kế thì sẽ không tránh khỏi kết quả trong tích hợp và sự cổ vũ thực hiện. Trong mô hình truyền thống, toàn bộ hệ thống đã được thiết kế trên giấy, sau đó được thực hiện (thử nghiệm) một lần, sau đó được tích hợp. Chỉ tại giai đoạn cuối của tiến trình này thì nó mới được thử nghiệm trên hệ thống kiểm tra để thẩm tra lại rằng kiến trúc thiết yếu (giao diện và cấu trúc) đã đúng đắn. Một chủ chủ đề tuần hoàn (lặp lại) của các dự án tiếp tục sau tiến trình truyền thống là kiểm tra các hoạt động, nó chiếm tới 40% hoặc trên 40% vòng đời của phương pháp. Bảng 1-1 cung cấp một kiểu sơ thảo về các chi phí phải trả qua toàn bộ phạm vi của các hoạt động phần mềm.

Phân tích rủi ro chậm.

Một sự phát sinh nghiêm trọng được kết hợp với vòng đời (chu trình) thác nước đã thiếu mất sự phân tích rủi ro sớm. Đây không phải là kết quả của chu trình thác nước mà là tiêu điểm trong các tạo tác sớm trên giấy, một trong các giai đoạn thiết kế thật, thực hiện và tích hợp các rủi ro vẫn tương đối khó nắm bắt. Hình 1-3 minh hoạ một kiểu phác thảo rủi ro trong

các dự án mô hình thác nước truyền thống. Nó gồm 4 chu kỳ bản chất rủi ro khác biệt, những nơi mà rủi ro được định nghĩa là có khả năng mất giá trị, lịch trình, đặc trưng hoặc mục tiêu chất lượng. Tính dễ dãi trong vòng đời như là các yêu cầu phải rõ ràng, các (trình bày) rủi ro hoạt động rất khó đoán trước được. Sau một thiết kế chấp nhận được đã có sẵn có phải cân bằng các hiểu biết về các yêu cầu, thậm chí nó mới chỉ trên giấy, thì trình bày về rủi ro cũng đã phải vững vàng. Tuy nhiên thường nó chỉ vững vàng ở một mức tương đối bởi vì cũng có vài sự việc rõ ràng với một nhà



Hình 1-3. *Rủi ro ban đầu của dự án phần mềm qua vòng đời của nó.*

quản lý phần mềm có được một đánh giá khách quan. Như hệ thống đã được mã hoá (lập trình), một vài thành phần rủi ro riêng lẻ đã được giải quyết (phân tích). Sau đó sự tích hợp được bát đầu, và phẩm chất thực của mức hệ thống và các rủi ro đã bắt đầu trở nên rõ ràng. Thường thì trong suốt chu kì này nhiều thiết kế phát sinh được giải quyết và các trả giá kỹ thuật đã được tạo ra. Tuy nhiên, quyết định các phát sinh muộn đó trong vòng đời, khi một kìm hãm cố định lớn thay đổi các tạo tác chính, là rất đắt giá. Kết quả là các dự án có chiều hướng kéo dài pha tích hợp (như minh hoạ trong hình 1-2) như là khả năng tái thiết kế cơ bản đã được thực hiện. Tiến trình này theo chiều hướng giải quyết các rủi ro quan trọng, mà nó không mất đi chất lượng của sản phẩm cuối cùng, đặc biệt là tính năng bảo trì của nó. Tôi sử dụng kỳ tái thiết kế khá lỏng lẻo. Hỗu hết những kết quả đạt được này sẽ được miêu tả tốt hơn như sự gán ghép muộn và sự chắp vá thành những hiệu lực sẵn có. Vì vậy toàn bộ kết quả đạt được vẫn là nhỏ nhất. Những sự thay đổi sắp xếp đó đã không bảo toàn được tất cả các thiết kế và sự tương hợp tính bảo trì.

Phân tích chức năng yêu cầu-điều khiển.

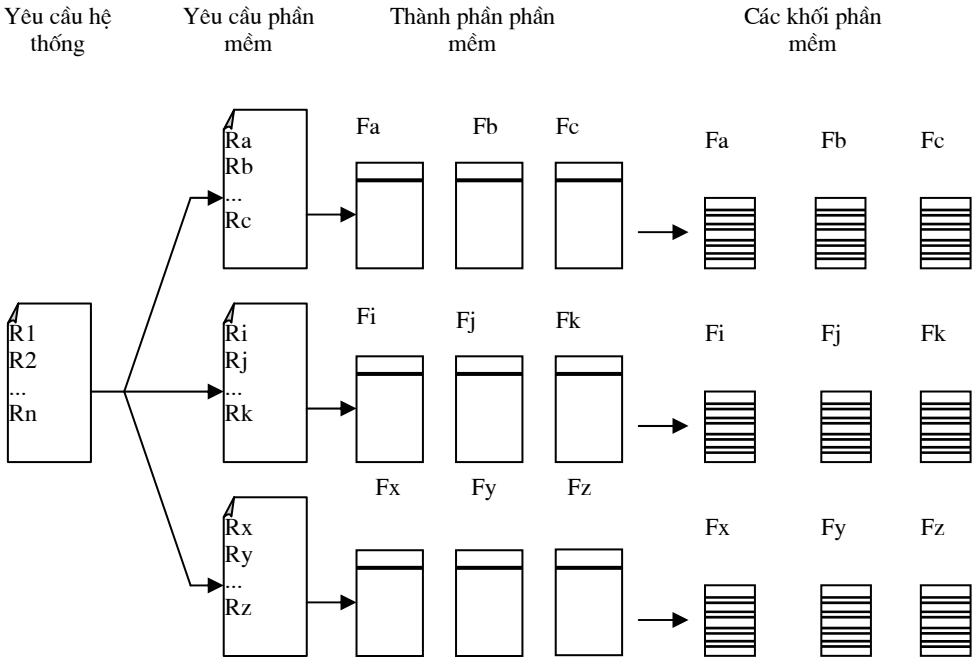
Theo truyền thống, tiến trình phát triển phần mềm là yêu cầu - điều khiển: Một sự nỗ lực là đảm bảo đưa ra một lời định nghĩa yêu cầu chính xác và sau đó để thực hiện chính xác các yêu cầu đó. Cách tiếp cận này phụ thuộc vào việc định rõ các yêu cầu một cách đầy đủ và rõ ràng trước khi các hoạt động phát triển khác bắt đầu. Sự thiếu chuyên môn sẽ xem xét toàn bộ các yêu cầu quan trọng như nhau và phụ thuộc vào các yêu cầu cố định còn lại đó trong vòng đời phát triển phần mềm. Những điều kiện hiếm hoi này lại xuất hiện trong thế giới thực. Đặc điểm kỹ thuật của các yêu cầu là một phần khó và quan trọng của tiến trình phát triển phần mềm. Như tranh luận trong phụ lục A, hầu như mọi chương trình phần mềm chính đều phải trải qua những thử thách khắc nghiệt trong các đặc điểm yêu cầu kỹ thuật. Hơn nữa tất cả các yêu cầu xử ký bình đẳng đều thoát ra khỏi số giờ kỹ thuật tất yếu, từ điều chỉnh các yêu cầu và lãng phí những sự cố gắng đó trên công việc văn phòng kết hợp với tính theo dõi, tính kiểm thử được, sự hỗ trợ hậu cần, và vì vậy công việc văn phòng chắc chắn sẽ bị loại bỏ muộn hơn. do việc điều chỉnh các yêu cầu và tính kế thừa các thiết kế tiên tiến đã biết.

Một ví dụ, xem xét một dự án lớn như dự án CCPDSR , được trình bày như một thí dụ nghiên cứu tiêu biểu (case study) trong phụ lục D, nơi mà các yêu cầu phần mềm bao gồm 2000 "shall" (Một "shall" là một yêu cầu riêng rẽ nghĩa là "hệ thống phải chịu đựng tất cả những hư hỏng phần cứng đơn lẻ mà không mất đi các khả năng tới hạn). Sự quan hệ tương xứng với định hướng thiết kế trong hệ thống như vậy (đặc biệt chỉ có từ 20 đến 50 "shall") là rất khó khăn khi sự chuẩn hoá giao ước yêu cầu toàn bộ 2000 shall được xác định trước và được xử lý tại mọi mốc chính. Mức nỗ lực công nghệ mà nó có thể được tiêu phí trong vấn đề thiết kế quan trọng là bị mờ nhạt bởi việc tiến hành vượt quá 1950 shalls và việc xử lý tính theo dõi được, tính kiểm thử được, tài liệu hoá được...

Một tính chất khác của cách tiếp cận truyền thống là các yêu cầu được xác định cụ thể theo nghĩa chức năng. Việc xây dựng quá trình thác nước cổ điển là giả định cơ bản mà phần mềm này tự phân rã thành các chức năng, các yêu cầu này đã được thiết lập thành những thành phần. Việc phân rã này thường rất khác với việc phân rã dựa trên thiết kế hướng đối tượng và sử dụng những thành phần hiện có. Việc phân rã chức năng cũng trở thành ràng buộc trong các hợp đồng, phụ lục hợp đồng và cấu trúc phân rã công việc, thông thường nó loại bỏ cách tiếp cận kiến trúc. Hình 4-1 minh hoạ kết quả của tiếp cận hướng yêu cầu, đó là một cấu trúc phần mềm mà được tổ chức xung quanh các cấu trúc xác định yêu cầu.

Những mối quan hệ cổ đông đối phương.

Tiến trình truyền thống dẫn tới kết quả trong quan hệ cổ đông đối phương, trong mối quan hệ lớn, bởi vì sự khó khăn trong việc xác định các yêu cầu và sự trao đổi thông tin chỉ qua những tài liệu giấy mà nó có thể bao gồm những thông tin công nghệ không theo thể thức. Sự thiếu những ký hiệu chặt chẽ xảy ra trong hầu hết các xem xét vấn đề và sự bảo thủ khi thay đổi thông tin.



Hình 1-4. Tổ chức các thành phần của phần mềm tối ưu hoá cục bộ từ cách tiếp cận hướng yêu cầu.

Dãy các sự kiện sau đây có thể được áp dụng đối với hầu hết những nỗ lực phần mềm có hợp đồng.

1. Người làm hợp đồng chuẩn bị một tài liệu hợp đồng nháp mà nó đề cập tới những mẫu tác trực tiếp và phân phát cho khách hàng để đánh giá.
2. Khách hàng sẽ được đề nghị để cung cấp những nhận xét (đặc biệt trong vòng từ 15 đến 30 ngày) .
3. Người chủ hợp đồng sẽ tổng hợp những nhận xét này và đệ trình một phiên bản cuối cùng để đánh giá (đặc biệt trong vòng từ 15 đến 30 ngày).

Quá trình xem xét ngắn gọn này thể hiện sự nhạy cảm cao độ giữa khách hàng và chủ hợp đồng. Quá trình xem xét sự thay đổi chỉ chiếu trên một trang giấy như vậy là không thể được. Cách tiếp cận này cũng đưa ra một mối quan hệ giữa khách hàng và chủ hợp đồng sẽ làm suy đổi sự không tin cậy lẫn nhau, làm khó khăn để đạt tới sự thoả thuận về các yêu cầu về kế hoạch và chi phí.

Trọng tâm các tài liệu và thảo luận trong các cuộc gặp gỡ.

Tiến trình truyền thống chú trọng vào việc đưa ra những tài liệu khác nhau để mô tả những sản phẩm phần mềm, mà thiếu chú trọng vào sự tăng trưởng thực tế của chính các sản phẩm đó. Những mốc chính thường được thực hiện như những nghi thức gặp gỡ được viết (định

Bảng 1-2. Kết quả của việc xem xét lại thiết kế của dự án phần mềm truyền thống.

Những kết quả rõ ràng	Những kết quả thực tế
Chỉ dẫn rộng rãi cho nhiều loại độc giả	Chỉ một tỉ lệ phần trăm nhỏ độc giả hiểu phần mềm. Những chỉ dẫn và tài liệu trình bày một vài giá trị quan trọng và những rủi ro của hệ thống phần mềm phức tạp.
Một thiết kế xuất hiện khá dễ dãi.	Có sự không rõ ràng xác thực của sự dễ dãI Sự dễ dãi với các yêu cầu mập mờ là một lượng nhỏ giá trị.
Các yêu cầu gộp (hàng trăm loại)	Một vài (khoảng 10) là các điều khiển thiết kế. Giải quyếta các yêu cầu làm mờ nhạt trọng tâm các điều khiển tới hạn.
Một thiết kế được coi là "vô tội cho đến khi được chứng minh là có tội"	Thiết kế luôn luôn có lỗi Các dòng thiết kế đã được bày ra muộn hơn trong vòng đời.

nghĩa) một cách đơn điệu trong các thuật ngữ của các tài liệu cụ thể. Những nhà đầu tư thường đưa ra hàng tấn những giấy tờ để đạt được những mốc và những quá trình mô phỏng cho cổ đông hơn là họ sử dụng các năng lực của mình đối với công việc mà nó sẽ làm giảm rủi ro và đưa ra những phần mềm có chất lượng. Đặc biệt những người trình bày và những người nghe sẽ xem xét lại những điều đơn giản mà họ đã hiểu hơn là những vấn đề quan trọng và phức tạp. Bởi vậy hầu như việc xem xét lại thiết kế sẽ đưa ra những giá trị công nghệ rất thấp mà chi phí cao theo sự nỗ lực và lập lịch trong việc chuẩn bị và điều khiển của họ. Họ trình bày hầu như là bề ngoài của quá trình. Bảng 1-2 tóm tắt những kết quả việc xem xét lại thiết kế.

Việc chuẩn đoán năm triệu chứng của dự án gây ra những trục trặc có thể là rất khó khăn (như chúng ta vừa thảo luận) đặc biệt trong những pha ban đầu của vòng đời khi vấn đề với cách tiếp cận thông thường hầu như rất dễ điều trị. Bởi vậy các dự án phần mềm hiện đại phải sử dụng một cơ chế để đánh giá độ mạnh của dự án trong những pha đầu của vòng đời và nó sẽ được tiếp tục với việc kiểm tra theo chu kì và khách quan (có mục tiêu).

1.2. Quản lý phần mềm thông thường

Một trang của Barry Boehm "danh sách dẫn đầu 10 độ đo phần mềm công nghiệp" [Boehm, 1987] là một đặc trưng khách quan tốt của tình hình phát triển phần mềm. (Có rất ít

các chứng cớ của những sự thay đổi quan trọng.) Mặc dù rất nhiều các độ đo, chúng đã mô tả một vài mối quan hệ kinh tế đơn giản mà nó là kết quả từ những quá trình phần mềm thông thường thực hành trên 30 năm qua.

Trong các đoạn sau, được trích dẫn từ danh sách 10 độ đo dẫn đầu của Boehm sẽ được trình bày bằng chữ in nghiêng, sau đó là phần giải thích:

1. *Phát hiện và chỉnh sửa các sai sót phần mềm sau khi phân phối có chi phí gấp hơn 100 lần so với việc phát hiện và chỉnh sửa các sai sót.*

➢ Cách này chiếm ưu thế chính trong hầu hết mọi hướng cải tiến tiến trình được nói đến ở đây hay trong bất kỳ cuốn sách nào khác. Nó không phải là cách độc nhất để phát triển phần mềm. Khi một trong những nhà máy ô tô lớn thi hành việc gọi lại một sản phẩm khiếm khuyết đã phát tán (ra thị trường), thì giá sắp xếp sửa chữa có thể lớn hơn nhiều so với chi phí sửa chữa khiếm khuyết đó trong giai đoạn kỹ thuật hoặc giai đoạn sản xuất.

2. *Bạn có thể nén lịch trình phát triển phần mềm tới 25% nhưng không được vượt quá.*

➢ Một lý do cho N% này giảm xuống trong lịch trình sẽ kéo theo M% kia tăng lên trong nguồn nhân lực (giả sử rằng các thông số còn lại khác giữ nguyên). Bất kỳ sự tăng lên về số lượng nhân lực nào cũng yêu cầu tổng phí quản lý lớn hơn. Nói chung, sự linh hoạt của giới hạn trong tổng phí này, theo lịch trình phù hợp với các hoạt động, bảo vệ dãy các hoạt động và các nguồn bắt buộc khác, vào khoảng 25%. Một cách tối ưu, sự nỗ lực của 100 nhân viên trong một tháng cũng sẽ bằng sự nỗ lực của 10 người trong 10 tháng. Công việc này có thể thực hiện được trong một tháng với 100 người không? Hay có thể thực hiện trong hai tháng với 50 người được không? Hoặc trong khoảng 5 tháng với 20 người không? Điều rõ ràng là, sự lựa chọn này là phi thực tế. Độ đo nén 25% nói rằng giới hạn trong sự lựa chọn này là 7 tháng rưỡi (và có thể đòi hỏi thêm có lẽ là khoảng 20 nhân viên làm việc theo chế độ bình thường). Bất kỳ lịch trình nén nào xa hơn cũng sụp đổ thành đống tro tàn. Trên một phương diện khác, một lịch trình tối ưu có thể bị kéo dài một cách tuỳ tiện và phụ thuộc vào con người, có thể được thực hiện trong thời gian dài hơn với nguồn nhân lực nhiều hơn đôi chút. Thí dụ, nếu bạn có một lịch trình xa xỉ trong 25 tháng, thì bạn có thể chỉ cần 75 nhân viên và 3 người làm việc theo chế độ bình thường.

3. *Sử dụng một đôla cho việc phát triển, nhưng bạn sẽ phải sử dụng 2 đôla cho việc bảo trì.*

➢ Boehm gọi điều này là "luật thép phát triển phần mềm." Dù bạn xây dựng một sản phẩm có tuổi thọ cao mà phiên bản ngoài thị trường nâng cấp 2 lần một năm hoặc xây dựng một hệ thống phần mềm đã đặt trước, số tiền được dùng để bảo trì vòng

đời của sản phẩm có thể sẽ nhiều gấp đôi số tiền đã được dùng để phát triển vòng đời sản phẩm. Sẽ rất khó khăn để nói rằng mối quan hệ đầu tiên này là tốt hay xấu. Trong miền sản phẩm lưu hành, quan hệ điều khiển chính này là sự thành công của sản phẩm trên thị trường. Các sản phẩm phần mềm thành công (như Oracle, các chương trình ứng dụng của Microsoft, Rational Rose, và hệ điều hành Unix) đã sống trên thị trường rất lâu và có thể kết quả là tỉ lệ giữa chi phí bảo trì và chi phí phát triển còn cao hơn nhiều. Trên một phương diện khác những nhà quản lý một trong các loại dự án phần mềm, hiếm thấy có một kế hoạch để chi tiêu nhiều như bảo trì phần mềm. Mặt khác, bất kỳ người nào đã làm việc trong nền công nghiệp phần mềm từ trên 10 đến 20 năm đều biết rằng phần mềm đang trong hoạt động được coi như là rất khó bảo trì.

4. *Các chi phí phát triển và bảo trì phần mềm là một chức năng chính của số lượng các dòng chương trình nguồn.*

- Độ đo này là kết quả cơ bản của ưu thế của phát triển phần mềm đặt trước, thiếu tính hợp thành thương mại và thiếu tính kế thừa những cái sẵn có trong kỷ nguyên tiến trình cổ điển.

5. *Sự biến đổi trong tính toán của con người là sự khác biệt lớn nhất trong các sản phẩm phần mềm.*

- Đây là chìa khoá của sự thông thái truyền thống: Thuê nhân công tốt. Độ đo này là một chủ đề của cả sự quá cường điệu và dưới mức cường điệu. Khi bạn không biết một cách khách quan tại sao lại bạn thành công hay thất bại, sự bung xung (scapegoat) rõ ràng là khả năng chuyên môn của con người. Sự đánh giá này là khách quan và khó thách thức.

6. *Toàn bộ tỉ lệ các chi phí về phần cứng đến phần mềm đều vẫn đang tăng trưởng. Năm 1955 tỉ lệ này là 15:85; năm 1985 là 85:15.*

- Thực tế phần mềm chiếm 85% giá trị của hầu hết các hệ thống là không quá nhiều đối với hiệu suất phần mềm (điều mà có thể cho rằng nó không tốt như chúng ta cần) như là khoảng mức chức năng đã được chỉ định đối với phần mềm trong hệ thống phân tán. Điều cần thiết cho phần mềm, tính rộng rãi của các chương trình ứng dụng, và tính phức tạp tiếp diễn để tăng trưởng không có giới hạn.

7. *Chỉ 15% nỗ lực phát triển phần mềm là được chuyển biến thành chương trình.*

- Đây là một chỉ báo quan trọng của sự cần thiết cho sự cân bằng. Nhiều hoạt động ngoài việc lập trình là cần thiết cho sự thành công của các dự án phần mềm. Quản lý yêu cầu, thiết kế , kiểm sửa, điều khiển dự án , quản lý thay đổi và các công cụ được xem xét tầm quan trọng như nhau mà nó chi phí khoảng 85% nguồn tài nguyên.

8. *Các hệ thống phần mềm và các sản phẩm tiêu biểu có giá trị nhiều hơn 3 lần trên một SLOC so với các chương trình phần mềm riêng rẽ. Các sản phẩm phần mềm hệ thống (ví dụ, hệ thống của các hệ thống) có giá trị nhiều gấp 9 lần.*

- Mối quan hệ luỹ thừa này là cần thiết cho những gì mà ta gọi là sự không kinh tế của các thang chia. Không giống như những hàng hoá khác càng có nhiều phần mềm được xây dựng thì chi phí càng cao đối với mỗi một dòng lệnh nguồn

9. *Vượt qua được trên 60% các lỗi.*

- Điều này là thực tế. Tuy nhiên, độ đo 1 đã cho sự vượt quá này là không nắm bắt được cái sai sót mà những phần chính và những sự chắc chắn không nắm bắt được đủ sớm trong vòng đời. Toàn bộ các phát hiện ra là không bằng nhau. Nhìn chung sự vượt qua và những dạng khác của các thanh tra con người đủ để nắm bắt được toàn bộ vấn đề và kiểu loại vấn đề. Nếu chúng ta sử dụng những kí hiệu thiết kế không theo chuẩn thì sự xem xét của con người có thể là một cơ chế đảm bảo chất lượng cơ bản nhưng nó không được tốt trong việc khắc phục những vấn đề bậc 2, bậc 3,... và bậc thứ n. Hơn nữa, một vài người rất giỏi trong việc xem xét lại những vấn đề ngữ nghĩa trong đoạn mã nguồn.

10. *80% sự đóng góp đến từ 20% những nhà đóng góp.*

- Đây là một phát biểu có trách nhiệm mà nó thực sự đúng trong hầu hết các nguyên lý công nghệ (hoặc là các nguyên lý chuyên nghiệp đối với vấn đề đó). Tôi sẽ mở rộng độ đo này trong việc giải thích cụ thể hơn đối với phần mềm. Sự thừa nhận cơ bản sau đây với một tỉ lệ cho các khuôn mẫu xử lý, quản lý phần mềm hiện đại:

80% công nghệ là chi phí bởi 20% của các yêu cầu

80% giá trị phần mềm là chi phí bởi 20% thành phần

80% sai sót là nguyên nhân bởi 20% thành phần

80% sự phế thải và làm lại phần mềm là bởi 20% của sai sót

80% các nguồn tài nguyên là tiêu phí bởi 20% thành phần

80% các công nghệ là hoàn thành bởi 20% công cụ

80% sự tiến bộ là được làm bởi 20% con người.

Những mối này đã đưa ra một và tiêu chuẩn tốt để đánh giá sự cải tiến của quá trình và công nghệ. Chúng biểu diễn các quy tắc thô mà nó đặc trưng cho tính chủ quan sự thực thi của quá trình quản lý phần mềm và kỹ thuật truyền thống. Trong những chương sau chúng ta sẽ quay trở lại những độ đo này để hợp lý hoá một cách tiếp cận mới bảo vệ cách tiếp cận cũ và cải thiện công nghệ và tiến trình.

Chương 2

Sự tiến hoá nền kinh tế phần mềm

Những điểm chính:

- *Những kết quả kinh tế của dự án phần mềm truyền thống phản ánh một nền công nghiệp bị chi phối bởi các tập quán phát triển, quy trình đặc biệt và tỉ lệ phi kinh tế.*
- *Giá trị của các mô hình hiện nay là cơ sở chính trong các cơ sở dữ liệu dự án thực nghiệm với một vài câu chuyện hợp thời lặp lại về thành công phát triển.*
- *Giá trị của một phần mềm tốt được ước lượng là rất khó được hoàn thiện, người đưa ra quyết định phải tập trung xử lý những ước lượng không chính xác.*
- *Một khuôn khổ tiến trình hiện đại tấn công vào tài nguyên cơ bản của tỉ lệ phi kinh tế cố hữu trong tiến trình phát triển phần mềm.*

Công nghệ phần mềm đã bị chi phối bởi các hoạt động trí tuệ mà trọng tâm là việc giải quyết các vấn đề phức tạp rộng lớn với nhiều điều chưa biết trong các phối cảnh tham dự. Những cách tiếp cận phần mềm sớm trong những thập niên 60 và 70 có thể rất tốt được mô tả như một sự điêu luyện với mỗi dự án sử dụng một tiến trình và những công cụ định trước. Trong những thập niên 80 và 90, nền công nghệ phần mềm đã hoàn thiện và chuyển biến sang một kỹ thuật có kỷ luật hơn. Tuy nhiên hầu hết các dự án phần mềm trong kỷ nguyên này vẫn chủ yếu là đi sâu nghiên cứu, bị chi phối bởi những ngươi tạo ra và tỉ lệ phi kinh tế. Các tiến trình phần mềm thế hệ tiếp theo, đặc biệt là các kỹ thuật trình bày trong cuốn sách này là nhằm hướng tới một cách tiếp cận chuyên sâu được chi phối bởi kỹ thuật tự động hoá và tỉ lệ phi kinh tế.

2.1.Nền kinh tế phần mềm

Hầu hết các mô hình giá trị phần mềm có thể được trừu tượng hoá thành một hàm có 5 tham số cơ bản là: kích thước, tiến trình, đội ngũ cán bộ, môi trường và yêu cầu chất lượng.

1. Kích thước của sản phẩm phần mềm cuối cùng (các thành phần của nó do con người tạo ra) cái mà tiêu chuẩn điển hình là số lượng câu lệnh nguồn hoặc số lượng các điểm chức năng đòi hỏi để phát triển các chức năng yêu cầu.
2. Tiến trình được sử dụng để sản xuất ra sản phẩm cuối cùng, đặc biệt khả năng của tiến trình để tránh các hoạt động thêm vào vô giá trị (sửa chữa làm lại, nhân viên nghỉ, tổng phí truyền thông.)
3. Năng lực của đội ngũ kỹ thuật viên phần mềm và đặc biệt là kinh nghiệm của họ về khoa học máy tính và các miền ứng dụng của dự án.

4. Môi trường, nơi mà đã có sẵn các công cụ và kỹ thuật đã được tạo ra để tăng cường hiệu năng phát triển phần mềm và để tự động hoá tiến trình.
5. Yêu cầu chất lượng của sản phẩm bao gồm: các đặc tính, hiệu năng, sự tin cậy và tính phù hợp.

Các quan hệ giữa các tham biến này và việc ước lượng giá trị có thể được viết như sau:

Công sức = (Đội ngũ nhân viên)(Môi trường)(Chất lượng)(Kích thước$^{\text{Tiến trình}}$).

Một vài mô hình tham số đã được phát triển để ước lượng giá trị phần mềm; hầu hết chúng có thể còn khá trừu tượng đối với mẫu biểu này. Một khía cạnh quan trọng của nền kinh tế phần mềm (như đã trình bày trong các mô hình chi phí phần mềm hiện nay) là mối quan hệ giữa công sức và kích thước trưng bày một tỉ lệ phi kinh tế. Tỉ lệ phi kinh tế của phát triển phần mềm là kết quả của sự ủng hộ tiến trình,tỉ lệ này lớn hơn 1.0. Điều trái ngược đối với hầu hết các tiến trình sản xuất là càng nhiều phần mềm được xây dựng thì nó lại càng đắt hơn trên một đơn vị khoản mục.

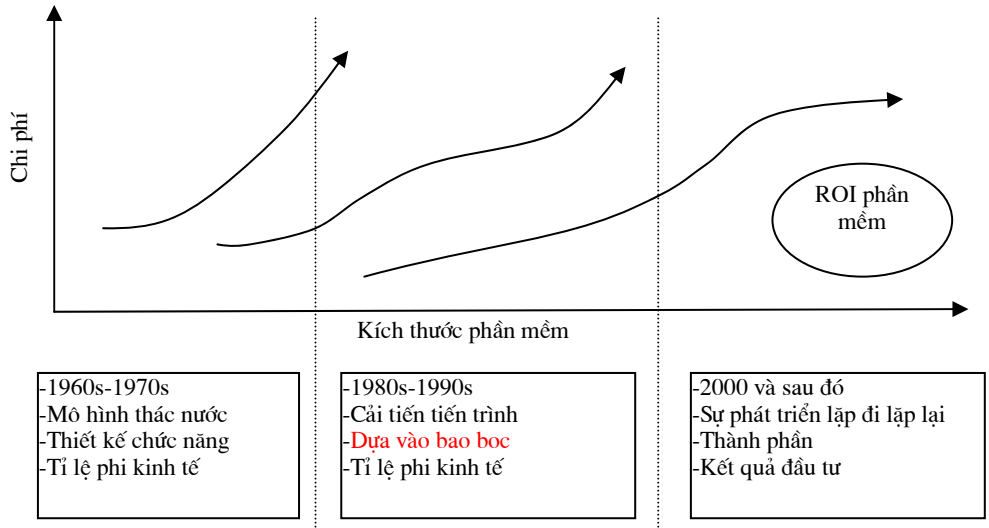
Ví dụ, cho một ứng dụng ,10000 dòng giải pháp phần mềm sẽ có giá trị nhỏ hơn trên một dòng so với 100000 dòng giải pháp phần mềm. Sự nhỏ hơn này là bao nhiêu? Giả sử rằng 100000 dòng hệ thống đòi hỏi 900 nhân viên-tháng để phát triển, hay khoảng 111 dòng trên một nhân viên-tháng, hay 1,37 giờ trên một dòng. Nếu cùng một hệ thống này, nhưng chỉ có 10000 dòng và tất cả các thông số khác giữ nguyên không đổi, thì hệ thống này sẽ được ước lượng là vào 62 nhân viên-tháng hay khoảng 175 dòng trên một nhân viên-tháng, hay 0,87 giờ trên một dòng. (Hình B-1 trong Phụ lục B sẽ đưa ra một mô tả chi tiết hơn về ví dụ này sử dụng mô hình ước lượng giá trị COCOMO.) Giá trị trên dòng đối với một ứng dụng nhỏ nhỏ hơn nhiều so với một ứng dụng lớn hơn. Nguyên nhân chính là sự phức tạp trong quản lý sự hoà hợp các cá nhân truyền thông như là số lượng các thành viên của đội (và sự phù hợp các mục tiêu, chiến thắng các điều kiện, các sai lệch kỹ thuật) tăng lên. Tỉ lệ phi kinh tế này là tính chất của bất kỳ dự án nghiên cứu nào nơi mà sản phẩm là một loại ví dụ của tài sản trí tuệ.

Hình 2-1 trình diễn 3 thế hệ kỹ thuật nâng cao cơ bản là các công cụ, các thành phần và các tiến trình. Các mức độ yêu cầu về chất lượng và đội ngũ nhân viên được giả thiết là không đổi. Tung độ của hình vẽ quy vào giá trị đơn vị phần mềm (đặt theo sở thích của bạn: trên một SLOC, trên một điểm chức năng, trên một phần tử) được thực hiện (bán ra) bởi một tổ chức. Hoành độ biểu diễn vòng đời của thương mại phần mềm được thuê bởi một tổ chức. Ba thời kỳ phát triển phần mềm được định nghĩa như sau:

1. Thời kỳ cổ truyền: thủ công, vào những thập niên 60 và 70. Việc tổ chức sử dụng các công cụ, các tiến trình tập quán cổ truyền và thực sự toàn bộ các phần tử được xây dựng lên từ các ngôn ngữ nguyên thuỷ. Hiệu năng của dự án được dự đoán cao trong các giá trị, lịch trình và chất lượng mục tiêu đó nhưng hầu hết chúng luôn không đạt được như dự đoán.

2. Thời kỳ quá độ: kỹ thuật phần mềm, vào những thập niên 80 và 90. Việc tổ chức sử dụng nhiều hơn các tiến trình lặp, các công cụ không lỗi thời và hầu hết (>70%) các thành phần tập quán được xây dựng trên các ngôn ngữ bậc cao hơn. Một vài thành phần (<30%) sẵn có từ trước như những sản phẩm thương mại, bao gồm hệ điều hành, hệ quản trị cơ sở dữ liệu, mạng máy tính, và giao diện đồ hoạ người dùng. Trong suốt những năm 80, một vài tổ chức đã bắt đầu giành được hiệu quả kinh tế, nhưng với sự tăng trưởng trong các ứng dụng phức tạp (các hệ thống chính đang chuyển sang hệ phân tán), những ngôn ngữ, những kỹ thuật và công nghệ đã có không đủ để làm nhụt (làm bẩn) được mong muốn về hiệu năng thương mại.
3. Thời kỳ hiên đại: sự sản xuất phần mềm, những năm 2000 và thời kỳ sau đó. Triết lý của cuốn sách này là nguồn gốc trong việc quản lý và phân phối các tiến trình, tích hợp các môi trường tự động, và hầu hết (70%) các thành phần không lỗi thời. Có lẽ khoảng 30% thành phần cần được làm theo sự đặt trước. Với những tiến bộ trong công nghệ phần mềm và sự tích hợp trong các môi trường sản xuất, các hệ thống thành phần cơ bản đó có thể được sản xuất rất nhanh chóng.

Mục tiêu đích: Cải tiến ROI (kết quả đầu tư) - Result of investmend



Sự phù hợp môi trường, kích thước, và các công nghệ tiến trình

Môi trường/Công cụ Tập quán	**Môi trường/Công cụ** Không lỗi thời, riêng rẽ	**Môi trường/Công cụ** Không lỗi thời, tích hợp
Kích thước: 100% tập quán	**Kích thước:** 30% Dựa vào thành phần-cơ bản 70% tập quán	**Kích thước:** 70% Dựa vào thành phần-cơ bản 30% tập quán
Tiến trình: Theo trường hợp đặc biệt	**Tiến trình:** Có thể lặp lại	**Tiến trình:** Quản lý/ Phân phối

Hiệu năng dự án điển hình

Khả năng dự đoán tồi	**Không có khả năng dự đoán**	**Có khả năng dự đoán**
Luôn luôn:	Không thường xuyên:	Thường xuyên:
Vượt quá ngân sách	Đúng ngân sách	Đúng ngân sách
Vượt quá lịch trình	Đúng lịch trình	Đúng lịch trình

Hình 2-1. Ba giai đoạn của nền kinh tế phần mềm dẫn tới mục tiêu đích.

Các công nghệ đối với môi trường tự động, sự giảm kích thước, và cải tiến tiến trình không độc lập với những điều khác. Trong mỗi kỷ ngyuên mới, chìa khoá là sự tăng trưởng hoàn thiện toàn bộ các công nghệ. Ví dụ một tiến trình tiên tiến sẽ không thể được sử dụng một cách thành công nếu thiếu các thành phần công nghệ mới và sự tăng trưởng của công cụ tự động hoá.

ROI đạt được qua đường thương mại.

Đầu tư vào kiến trúc, tiến trình, và môi trường chung cho toàn bộ các hệ thống đường thương mại	Hệ thống đầu tiên	Hệ thống thứ hai	Hệ thống thứ N

Vòng đời kinh doanh: hệ thống kế tiếp

ROI đạt được qua một dự án với nhiều sự lặp lại.

Đầu tư vào kiến trúc thô, tiến trình lặp chín chắn, và tiến trình tự động hóa.	Sự lặp lại đầu tiên	Sự lặp lại thứ hai	Sự lặp lại thứ N

Vòng đời dự án: Cách lập liên tục

ROI đạt được qua một vòng đời của sản phẩm bán ra.

Đầu tư vào kiến trúc, sản phẩm trường chung cho toàn bộ các hệ thống đường thương mạI quá trình bán và tự động hoá tiến trình.	Lần bán đầu tiên	Lần bán thứ 3	Lần bán N

Vòng đời sản phẩm:

Hình 2-2. Kết quả đầu tư trong những miền khác nhau.

Sự quá độ lên mô hình hiện đại và sự hứa hẹn cải tiến nền kinh tế phần mềm không có nghĩa là nó đã được bảo đảm. Chúng ta phải thực tế trong sự so sánh tính khả thi của sự hứa hẹn, tiến trình thế hệ tiếp theo sử dụng các công nghệ hiện đại chống lại những thực tế không tốt còn lại của lịch sử. Một điều đánh cuộc chắc chắn rằng nhiều tổ chức đang cố gắng mang về các dự án hiện đại với những kỹ thuật và công nghệ hiện đại nhưng cũng kết thúc với sự hỗn độn tương tự trước đó.

Nhiều tổ chức đang đạt được tỉ lệ kinh tế tốt hơn trong kỷ nguyên công nghệ kế tiếp với những dự án rất lớn (những hệ thống của hệ thống), các sản phẩm tuổi thọ cao và các đường thương mại chứa nhiều dự án tương tự. Hình 2-2 đưa ra một điều tra tổng quát về kết quả đầu tư (ROI) ban đầu có thể đạt được trong sự nỗ lực kế tiếp qua các vòng đời của các miền khác nhau.

2.2.Sự ước lượng chi phí phần mềm thực dụng

Một vấn đề giới hạn trong việc ước lượng chi phí phần mềm là thiếu một tài liệu chuẩn làm căn cứ để nghiên cứu các dự án mà sử dụng một cách tiếp cận phát triển lặp lại. Mặc dù mô hình chi phí mà những người bán khẳng định rằng các công cụ của họ là phù hợp đối với ước lượng sự phát triển các dự án, một vài cơ sở trong các cơ sở dữ liệu dự án thực nghiệm với những câu chuyện hiện đại về sự thành công phát triển lặp lại. Hơn nữa, bởi vì nền công nghiệp phần mềm có sự không đồng nhất trong việc định nghĩa các độ đo hoặc các đơn vị đo đạc nguyên tố, dữ liệu từ các dự án hoạt động có dấu hiệu đáng ngờ cao trong giới hạn kiên định và tính so sánh được. Sẽ khó có thể thu lượm đủ các bộ dữ liệu dự án đồng nhất trong một tổ chức, sẽ cực kỳ khó để đồng nhất các dữ liệu qua các tổ chức khác nhau với những tiến trình, những ngôn ngữ, và các miền khác nhau..v.v... Ví dụ, kích thước đơn vị cơ bản (một đường mã nguồn hay điểm chức năng) có thể, và được đo khác nhau qua nền công nghiệp. Đó là điều ngẫu nhiên mà ngôn ngữ chuẩn hiện đại (như Ada 95 và Java) cũng không tạo ra một định nghĩa đơn giản về một đường thông báo nguồn bằng bộ dịch của nó. Định nghĩa chính xác về điểm chức năng hoặc một SLOC là không thật quan trọng, nó cũng chỉ như độ dài chính xác của một foot (0,8085 mét) hay một mét là bằng giá trị tuỳ ý. Điều quan trọng đơn giản là mọi người cùng sử dụng một cách định nghĩa.

Có nhiều thế đứng dài được thảo luận giữa những người phát triển và những người bán các mô hình và công cụ ước lượng chi phí phần mềm. Ba chủ đề của các thảo luận đặc biệt thú vị sau đây:

1. Mô hình ước lượng chi phí được sử dụng.
2. Có hay không một cách đo kích thước phần mềm trong các đường mã nguồn hay điểm chức năng.
3. Những gì cấu thành nên một ước lượng tốt.

Khoảng 50 nhà cung cấp (bán) các công cụ ước lượng chi phí phần mềm ,dữ liệu và các dịch vụ hoàn thiện ở trong nền công nghiệp phần mềm. Các mô hình ước lượng chi phí (như COCOMO, CHECKPOINT, ESTIMACS, KnowledgePlan, Price-S, ProQMS, SEER, SLIM, SOFTCOST, và SPQR/20) tốt như đông đảo các mô hình tổ chức-rõ ràng. Bởi vì kinh ngiệm đầu tay của tôi với các mô hình này là trọng tâm của COCOMO và những người thành công với nó, Ada COCOMO và COCOMO II, đó là nền tảng trong nhiều lí lẽ và triển vọng về nền kinh tế phần mềm của tôi. COCOMO cũng là một trong những mô hình mở nhất và sự ước lượng chi phí được tài liệu hoá tốt. Sự phát triển của COCOMO sang phiên bản hiện tại của nó, COCOMO II, đã được tổng kết trong phụ lục B. Trong khi các phần của phụ lục không thể trực tiếp áp dụng các kỹ thuật và công nghệ hiện nay, thì nó đưa ra một triển vọng lịch sử thú vị trong sự phát triển những phát sinh và những ưu tiên của nền kinh tế phần mềm qua 20 năm qua.

Việc đo đạc kích thước phần mềm là chủ đề của nhiều bài hùng biện. Điều cơ bản về 2 điểm mục đích của quan điểm: các đường mã nguồn và các điểm chức năng. Cả hai triển vọng đều đã được chứng minh là có giá trị hơn cái thứ ba, nó là mục tiêu hay điểm đặc biệt của quan điểm thực hành của nhiều tổ chức non nớt mà sử dụng không theo hệ thống đo đạc kích thước.

Nhiều nhà chuyên môn phần mềm đã chỉ rõ rằng SLOC là một cách đo đạc kích thước tồi tệ. Tuy nhiên khi đoạn mã là mô tả như 1000 dòng lệnh chương trình nguồn, thì hầu hết mọi người đều cảm thấy thoải mái với tổng thể "khối" của nó. Nếu sự mô tả là 20 điểm chức năng, 6 lớp, 5 trường hợp sử dụng, 4 điểm đối tượng, 6 tệp, 2 hệ thống con, 1 phần tử hoặc 6000 byte, hầu hết mọi người, bao gồm cả chuyên gia phần mềm, cũng sẽ hỏi những câu hỏi xa xôi để đạt được một sự hiểu biết về đối tượng mã chương trình. (Nhiều người trong số họ sẽ hỏi có bao nhiêu SLOC). Vì vậy SLOC vẫn là một cách đo đạc còn một vài giá trị.

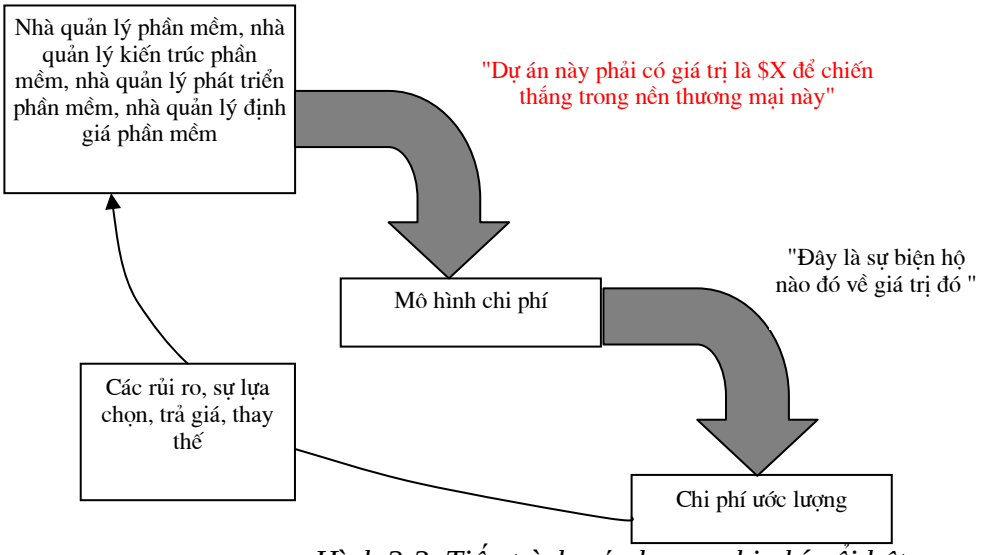
Tôi là một SLOC cuồng tín một thập kỷ trước đây bởi vì SLOC đã làm việc rất tốt trong các ứng dụng mà hầu hết được đặt trước và bởi vì SLOC là một phương pháp đo đạc dễ tự động hoá và dễ cung cấp. Ngày nay, ngôn ngữ bậc cao và sử dụng các phần tử, sự sinh ra tự động hoá mã nguồn và sự hướng đối tượng đã làm cho SLOC có nhiều đo đạc nhập nhằng hơn. Như một ví dụ sắc bén, trường hợp nghiên cứa trong phụ lục D mô tả rất cẩn thận các cách tiếp cận thủ công đối với đo đạc SLOC để điều tiết việc sử dụng lại, tập quán phát triển, và các công cụ sinh mã trong một dự án phần mềm lớn.

Việc sử dụng những điểm chức năng có một sự noi theo lớn, bao gồm Capers Jones, người trích dẫn (cites) sự kết hợp đầy may rủi với việc sử dụng độ đo SLOC cho các chương trình hướng đối tượng [Jones,1994]. Tổ chức "Nhóm của người sử dụng điểm chức năng quốc tế" (The International Function Point User's Group), được thành lập năm 1884, là tổ chức đo đạc phần mềm tiêu biểu trong công nghiệp. Ưu điểm chính trong việc sử dụng các điểm chức năng là phương pháp này độc lập với công nghệ và vì thế có nhiều điều tốt hơn đơn vị nguyên thuỷ để so sánh giữa các dự án và các tổ chức. Nhược điểm chính của nó là các định nghĩa

nguyên thuỷ khá trừu tượng và cách thức đo đạc không dễ bắt nguồn trực tiếp từ việc mở ra các tạo tác.

Mặc dù cả hai cách đo đạc kích thước đều có trở ngại của chúng, nhưng tôi nghĩ một tổ chức có thể tạo ra cả hai trong một công việc. Việc sử dụng một vài cách đo là tốt hơn không dùng cách nào. Bất kỳ người nào đang làm các phép so sánh qua các dự án hoặc tổ chức thì đều nên sử dụng các điểm chức năng như là cách đo đạc kích thước. Các điểm chức năng cũng có thể ước lượng chính xác hơn trong những pha sớm của một vòng đời dự án. Tuy nhiên trong những pha muộn hơn, SLOC sẽ trở nên hữu ích và rõ ràng hơn phép đo đạc cơ bản của những độ đo triển vọng khác nhau. Chương 16 trình bày giả thuyết của tôi về một mô hình chi phí thế hệ tiếp theo mô hình mà có thể nhỏ nhất hay thậm chí lỗi thời điều cần thiết đối với phương pháp đo SLOC.

Sự đúng đắn nói chung của các mô hình chi phí truyền thống (như COCOMO) đã được mô tả như "ở trong 20% các hoạt động, 70% thời gian." Mức không thể dự đoán này trong tiến trình phát triển phần mềm truyền thống sẽ thực sự khủng khiếp đối với mọi nhà đầu tư, đặc biệt trong trong ánh sáng của thực tế một vài dự án bị mất ước lượng của nó bởi những việc làm tốt hơn sự mong đợi. Đây là một hiện tượng thú vị được coi trọng khi lập lịch trình công sức lao động chuyên sâu. Trừ khi sự khuyến khích cụ thể được cung cấp để đánh vào toàn bộ lịch trình, các dự án hiếm khi thực hiện tốt hơn kế hoạch đã đặt ra. Tại sao vậy? Đồng đội và các cá nhân thực hiện kế hoạch con (một phần kế hoạch) để gặp nhau ở các mục tiêu chung của họ. Nếu mục tiêu thời gian là không khắt khe, họ sử dụng hết nguồn năng lượng ở một nơi nào đó (trong đào tạo từ xa, những sự giúp đỡ khác, hay sử dụng vào mục đích ngu ngốc) hoặc họ tiếp tục tăng thêm chất lượng vượt xa những gì cần thiết. Hầu hết họ chưa bao giờ đề nghị thực thi nhanh lịch trình. Nếu họ làm , đề nghị của họ hầu như sẽ gặp sự chống đối của các cổ đông khác, người mà đang mong đợi sự đồng bộ hoá. Vì vậy các kế hoạch cần phải mập mờ như nó có thể đạt được.



Hình 2-3. Tiến trình ước lượng chi phí nổi bật.

Hầu hết thực tế thế giới sử dụng các mô hình chi phí từ dưới lên (bottom-up) (chứng minh một chi phí đích) nhiều hơn là sử dụng mô hình từ trên xuống (top-down) (ước lượng chi phí "cần"). Hình 2-3 minh hoạ thực hành nổi bật: Nhà quản lý dự án phần mềm định nghĩa chi phí đích của phần mềm, sau đó vận động theo các tham biến và kích thước cho đến khi chi phí đích có thể được bào chữa. Nguyên nhân cơ bản để chi phí đích có thể vượt qua một dự kiến, để thu hút khách hàng tài trợ, để dành được tài trợ của các đoàn thể bên trong, hay để đạt được một vài mục đích khác.

Tiến trình được miêu tả ở Hình 2-3 là không hoàn toàn không tốt. Trong thực tế nó là một sự lỗi thời cần thiết để phân tích chi phí rủi ro và để hiểu được sự nhạy cảm và để trả giá một cách khách quan. Nó bắt buộc nhà quản lý phần mềm phải thẩm tra các rủi ro kết hợp với việc đạt được các chi phí đích và để bàn thảo các thông tin này với những cổ đông khác. Kết quả thường là các trạng thái lo lắng khác nhau trong các kế hoạch, các thiết kế, tiến trình, hay các phạm vi đã được đề xuất. Tiến trình này đưa ra một phương tiện truyền bá tốt cho nền tảng của ước lượng và toàn bộ chi phí phân tích.

Một bài thực hành được học tập từ trong lĩnh vực là độc lập với các ước lượng chi phí (những điều đó đã được làm bởi những người độc lập trong đội ngũ phát triển) là thường không đúng đắn. Con đường duy nhất để sản sinh ra một ước lượng đúng đắn là để cho một đội quản lý dự án phần mềm và nhà kiến trúc phần mềm, nhà phát triển, và những nhà quản lý kiểm thử có đủ khả năng (giỏi) để ước lượng lặp đi lặp lại vài lần và nhạy cảm trong phân tích. Đội ngũ này phải nắm lấy quyền sở hữu ước lượng chi phí đó cho các dự án thành công.

Điều gì cấu thành nên một ước lượng chi phí phần mềm tốt? Câu hỏi dai dẳng này đã được bàn thảo chi tiết ở trong chương 10. Tóm lại, một ước lượng tốt có các thuộc tính sau đây:

- Nó đã được hiểu và ủng hộ của nhà quản lý dự án, đội ngũ kiến trúc, đội ngũ phát triển và đội ngũ kiểm tra có trách nhiệm thực thi công việc.
- Nó đã được chấp nhận bởi tất cả các cổ đông như sự mập mờ nhưng có thể nhận thức được.
- Nó được cơ sở trong một mô hình chi phí phần mềm được định nghĩa tốt với một nền tảng đúng đắn.
- Nó được cơ sở trong một cơ sở dữ liệu của dự án có kinh nghiệm phù hợp nó bao gồm những tiến trình tương tự, công nghệ tương tự , môi trường tương tự, yêu cầu chất lượng tương tự, và những con người tương tự.
- Nó được định nghĩa đầy đủ chi tiết đến nỗi mà các vùng rủi ro chính của nó đã được nhận biết và khả năng thành công là đánh giá khách quan.

Sự ngoại suy từ một ước lượng tốt, một quan niệm ước lượng sẽ nhận được từ một mô hình chi phí trưởng thành với một nền tảng kinh nghiệm đã được phản chiếu từ nhiều dự án tương tự đã được thực hiện bởi cùng một đội ngũ với những tiến trình và công cụ trưởng thành tương tự. Mặc dù hoàn cảnh này hiếm khi tồn tại khi một đội ngũ làm dự án bắt tay vào một dự án mới, những ước lượng tốt có thể đạt được trong một cách thức thẳng thắn trong pha vòng đời muộn hơn của một dự án chín muồi sử dụng một tiến trình chín muồi.

Chương 3

Cải tiến kinh tế phần mềm

Những điểm chính:

- *Công nghệ phần mềm hiện đại cho phép các hệ thống được xây dựng với các dòng tài nguyên sinh ra từ nhiều người hơn.*
- *Các tiến trình hiện đại lặp đi lặp lại*
- *Sự phát triển phần mềm và môi trường bảo quản hiện đại là sự phân chia máy móc cho tiến trình tự động.*

Sự cải tiến trong phát triển kinh tế phần mềm không chỉ khó đạt được mà còn khó đo đạc và chứng minh. Trong các sách phần mềm, các báo thương mại, và các sản phẩm văn học, sự bàn luận về chủ đề này được quấy đảo bởi những biệt ngữ trái ngược, những đơn vị đo đạc trái ngược, sự bất đồng giữa các chuyên gia, và những lời cường điệu không dứt. Nếu chúng ta thẩm tra bất kỳ một khía cạnh nào của sự cải tiến kinh tế phần mềm, thì chúng ta cũng kết thúc trên những kết luận khá hẹp và một sự quan sát được giới hạn giá trị. Tương tự, nếu một tổ chức chú trọng quá nhiều vào cải tiến một khía cạnh của tiến trình phát triển phần mềm, thì sẽ không nhận ra bất kỳ một sự cải tiến kinh tế có ý nghĩa nào thậm chí nó chỉ cải tiến một khía cạnh đẹp mắt đó.

Chìa khóa để chứng minh sự cải tiến là sự tấn công cân bằng vào một vài quan hệ định hướng. Tôi có cơ cấu trình bày các định hướng quan trọng xung quanh 5 tham số cơ bản của mô hình chi phí phần mềm đã trình bày trong chương 2.

1. Giảm kích thước hoặc sự phức tạp của những điều cần thiết cho phát triển.
2. Cải tiến tiến trình phát triển.
3. Sử dụng nhiều hơn những nhân viên lành nghề và đội ngũ lao động tốt hơn (không cần thiết những điều tương tự).
4. Sử dụng môi trường tốt hơn (công cụ để tự động hóa tiến trình).
5. Trả giá hoặc không lùi (hạ thấp) giới hạn điểm khởi đầu chất lượng.

Những tham số đưa ra theo thư tự ưu tiên đối với hầu hết các phần mềm. Bảng 3-1 là danh sách một vài công nghệ phát triển, nỗ lực cải tiến tiến trình và quản lý cách tiếp cận mục tiêu trong cải tiến kinh tế của sự phát triển phần mềm và sự tích hợp.

Hầu hết các chuyên gia phần mềm đều công nhận tính độc lập có ý nghĩa giữa các chiều hướng này. Ví dụ, các công cụ cho phép kích thước giảm xuống và sự cải tiến tiến trình,sự giảm kích thước sẽ dẫn tới tiến trình bị thay đổi và sự cỉa tiến tiến trình sẽ tác động đến công cụ yêu cầu. Xem như là miền của phần mềm giao diện người dùng. Trong 2 thập kỷ qua, các đội ngũ đang phát triển một giao diện người dùng đã tiêu tốn một thời gian phân tích thao tác

và nhân tố con người rộng lớn, cách bày hcí màn chắn, và màn chắn động. Tất cả những điều này đã được làm trên giấy, bởi vì nó cực kỳ đắt giá nếu phạm vào thiết kế, thậm chí nó không hợp lệ với các mẫu gốc, để có thể thực hiện mã hóa được. Vì thế tiến trình được nhấn mạnh một cách khá nặng nề trong các tạo tác sớm trên giấy và hoà hợp người dùng, vì vậy "những yêu cầu" này có thể bị đóng băng và các chi phí xây dựng cao có thể trở nên nhỏ nhất.

Bảng 3-1. Các chiều hướng quan trọng trong cải tiến kinh tế phần mềm.

Các tham số mô hình chi phí	Các chiều hướng
Kích thước Sự trừu tượng và các thành phần công nghệ phát triển cơ bản	Các ngôn ngữ trật tự bậc cao (C++, Ada 95, Java, Visual Basic ...) Hướng đối tượng (phân tích, thiết kế , lập trình) Tái sử dụng Các phần tử thương mại.
Tiến trình Các phương pháp và kĩ thuật	Lặp lại sự phát triển Tiến trình các mô hình chín muồi Kiến trúc phát triển đầu tiên Cải tổ đạt được
Đội ngũ nhân viên Các nhân tố con người	Đào tạo và phát triển đội ngũ lao động lành nghề Đội ngũ công nhân Chiến thắng các yếu tố văn hoá
Môi trường Các công nghệ và công cụ tự động hoá	Tích hợp các công cụ (mô hình trực quan, bộ dịch, người biên tập, người gỡ rối, quản lý thay đổi...) Các hệ thống mở Hiệu năng nền phần cứng Tự động hoá lập trình, tài liệu, kiểm thử, phân tích
Chất lượng Hiệu năng, độ tin cậy, tính đúng đắn	Hiệu năng nền phần cứng Luận chứng cơ sở đánh giá Điều chỉnh chất lượng thống kê

Công nghệ giao diện người dùng đồ hoạ (GUI - Graphical User Interface) là một ví dụ tốt về các công cụ cho phép một điều mới và một tiến trình khác. Khi công nghệ GUI đã chín muồi, quy trình giao diện người dùng truyền thống đã trở nên lỗi thời. Người xây dựng công nghệ GUI đã cho phép đội ngũ kỹ thuật xây dựng một giao diện người dùng có thể được thực hiện nhanh hơn và chi phí thấp hơn. Những mô tả trên giấy bây giờ đã không cần thiết nữa, trong thực tế chúng là một trở lực đối với hiệu năng của tiến trình. Các thao tác phân tích và nhân tố con người phân tích vẫn còn quan trọng, nhưng những hoạt động này hiện nay đã được làm trong môi trường mục tiêu thực tế sử dụng những cái nguyên thuỷ sẵn có và những nền tảng đang xây dựng. Kỹ thuật và các vòng phản hồi mà được sử dụng để làm trong hàng tháng trước đây thì nay có thể được làm trong ngày hoặc tuần. Tiến trình cũ đã ăn khớp theo hướng bảo đảm giao diện người dùng đã hoàn toàn được phân tích và thiết kế, bởi vì dự án chỉ có đủ cho một vòng xây dựng. Tiến trình đã ăn khớp theo hướng tạo nên giao diện người dùng qua một vài phiên bản thực tế, hợp nhất tất cả phản hồi người dùng theo quá trình và đạt được một hiểu biết ổn định về các yêu cầu và thiết kế phát sinh để cân bằng với một quá trình khác.

Có thể khẳng định rằng tiến trình tiên tiến (như sự cần thiết đối với sự lặp lại và sự thử nghiệm trong việc định nghĩa các giao diện người dùng) đã điều khiển sự phát triển của các công cụ, hay các công nghệ tiên tiến đã làm tiến trình thay đổi. Thực tế có thể là hỗn hợp của cả hai. Điều đó nói lên rằng năm tham biến cơ bản của sự cân bằng ước lượng chi phí là không qua lại riêng biệt, chúng cũng không độc lập với cái khác. Chúng có một sự quan hệ qua lại.

Một nhân tố quan trọng khác đã có ảnh hưởng đến sự cải tiến công nghệ phần mềm, ở một mặt khác là sự nâng cấp hiệu năng phần cứng. Sẵn có nhiều chu kỳ hơn, nhiều bộ nhớ hơn, và nhiều dãy tần số trong băng hơn đã loại bỏ nhiều tài nguyên phần mềm thực hiện phức tạp. Đơn giản hơn, những giải pháp bắt buộc mạnh mẽ giờ đây là có thể chấp nhận được, và sự cải tiến phần cứng có thể được cho phép nâng cấp sau hầu hết những cải tiến công nghệ phần mềm thực chất.

3.1 Giảm kích thước sản phẩm phần mềm

Cách có ý nghĩa nhất để có thể cải tiến và có thành quả đầu tư (ROI) là thường xuyên sản xuất một sản phẩm mà đạt được mục tiêu thiết kế với một tổng phí tài nguyên vật chất sinh ra của con người thấp nhất. Sự phát triển thành phần cơ bản được giới thiệu ở đây như giới hạn chung cho việc giảm kích thước ngôn ngữ "nguồn" cần thiết để đạt được giải pháp phần mềm. Việc tái sử dụng công nghệ hướng đối tượng, tự động hoá sự sản xuất chương trình và ngôn ngữ lập trình thứ tự bậc cao là toàn bộ trọng tâm trong việc đạt được hệ thống với ít dòng tài nguyên trực tiếp của con người hơn (lời phát biểu). Sự giảm kích thước này là động cơ thúc đẩy chính sau những sự cải tiến trong các ngôn ngữ thứ tự bậc cao (như là C++, Ada 95, Java, Visual Basic, và các ngôn ngữ thế hệ thứ 4), các bộ sinh mã tự động (các công cụ CASE (Computer Aid Software Engineering - Công nghệ phần mềm có máy tính hỗ trợ), các công cụ

làm mô hình trực quan, người xây dựng GUI (giao diện người dùng đồ hoạ)), tái sử dụng các thành phần thương mại (hệ điều hành, môi trường cửa sổ, hệ quản trị cơ sở dữ liệu, thiết bị trung gian, mạng máy tính) và công nghệ hướng đối tượng (ngôn ngữ làm mẫu thống nhất, các công cụ làm mẫu trực quan, kiến trúc các khung công việc).

Một sự báo trước đã được chứng nhận khi đang bàn luận sự giảm xuống của kích thước sản phẩm. Trên bề mặt, sự giới thiệu này xuất phát từ một sự quan sát đơn giản: Mã không ở đó thì không cần được phát triển và không thể bị hỏng. Nhưng đây là nhánh không toàn vẹn. Sự giảm xuống được định nghĩa trong giới hạn tài nguyên vật chất được sinh ra của con người. Nhìn chung khi các công nghệ làm giảm kích thước được sử dụng, thì chúng làm giảm số lượng dòng tài nguyên được sinh ra của con người. Tuy nhiên, tất cả chúng quay về tăng tổng số tiến trình có thể thực hiện mã hoá trên máy tính. Vì vậy phần đầu tiên của sự quan sát phải là đúng, nhưng phần hai thì không nhất thiết phải đúng. Dòng dưới cùng, như kinh ngiệm của nhiều đội làm dự án, là sự chín muồi và các công nghệ giảm kích thước đáng tin cậy là cực kỳ mạnh mẽ trong việc sản sinh lợi nhuận kinh tế. Những công nghệ giảm kích thước chưa chín muồi có thể giảm sự phát triển kích thước nhưng đòi hỏi nhiều sự đầu tư để đạt được các mức chất lượng và hiệu năng cần thiết đến nỗi mà chúng có một tác động phản tác dụng lên toàn bộ hiệu năng dự án.

3.1.1 Các ngôn ngữ

Toàn bộ các điểm chức năng (UFPs) là những nhà ước lượng hữu dụng đối với ngôn ngữ độc lập, các ước lượng vòng đời sớm. Đơn vị cơ bản của các điểm chức năng là các đầu vào của người dùng bên ngoài, các đầu ra bên ngoài, các nhóm dữ liệu logic bên trong, các giao diện dữ liệu bên ngoài và các yêu cầu bên ngoài. Các phương pháp đo SLOC là những nhà ước lượng hữa dụng đối với phần mềm sau một giải pháp ứng cử đã được làm thành công thức và một ngôn ngữ thực thi đã biết. Dữ liệu quan trọng đã được tài liệu hoá quan hệ SLOC với các điểm chức năng [Jones, 1995]. Một vài kết quả này được trình bày trong bảng 3-2.

Dữ liệu trong bảng minh hoạ tại saolại quan tâm đến các ngôn ngữ hiện đại như là C++, Ada 95, Java, và Visual Basic: Mức độ có thể biểu diễn được là rất hấp dẫn. Tuy nhiên, sự bảo

Bảng 3-2. Tính xúc cảm ngôn ngữ của một vài ngôn ngữ phổ biến ngày nay.

Ngôn ngữ	SLOC trên UFP (Điểm chức năng)
Assembly	320
C	128
FORTRAN 77	105

COBOL 85	91
Ada 83	71
C++	56
Ada 95	55
Java	55
Visual Basic	35

dưỡng phải được tạo ra để áp dụng vào các dữ liệu này bởi vì nhiều khả năng có thể bị lạm dụng. Trong khi tôi tin rằng dữ liệu đúng đắn trình bày lại một mối quan hệ quan trọng, những số nhiều một cách quá rõ ràng. (Chúng không thắc mắc sự rõ ràng trung bình của một vài số không rõ ràng). Mỗi một ngôn ngữ có một miền sử dụng. Visual Basic rất diễn cảm (expressive)và mạnh mẽ trong xây dựng các ứng dụng tương tác đơn giản, nhưng nó không phải là sự lựa chọn khôn ngoan đối với một chương trình thời gian thực, chương trình nhúng, chương trình khoa học điện tử áp dụng vào hàng không. Tương tự, Ada 95 có thể là ngôn ngữ tốt nhất đối với một hệ thống chi phí lỗi thảm khốc mà điều khiển một kế hoạch năng lượng hạt nhân, nhưng nó không phải là sự lựa chọn tốt nhất đối với các chương trình song song cao, chương trình khoa học, chương trình xử lý số chạy trên siêu máy tính. Dữ liệu công nghiệp phần mềm như vậy, cái mà mở rộng các miền ứng dụng, các tập đoàn, và những phát sinh công nghệ phải được diễn dịch và đươc sử dụng với sự bảo dưỡng rất lớn.

> *Hai sự quan sát thú vị trong dữ liệu có liên quan khác nhau và mối quan hệ giữa Ada 83 và Ada 95, và giữa C và C++. Điều thú vị của Cục bảo vệ (DOD- Department of Defense) trong phát triển Ada 83 đã thừa hưởng trong phần để tăng tính tính diễn cảm (expressiveness) của nó. (Những nguyên nhân khác bao gồm tính tin cậy, sự ủng hộ đối với chương trình thời gian thực, tính có thể bảo trì được, và cải tiến ROI qua ngôn ngữ tiêu chuẩn hoá). Một sự thúc đẩy kinh tế có ý nghĩa là khả năng để phát triển một chương trình trong thực tế với số dòng mã hóa ít hơn đòi hỏi lựa chọn trong ngôn ngữ truyền thống như là FORTRAN, COBOL, C, và hợp ngữ (asembly). Biểu hiện trong ngôn ngữ Ada là nhiều công nghệ kỹ thuật phần mềm tiên tiến, bao gồm ngôn ngữ làm cho được cấu hình điều khiển, sự chia cắt giao diện, và sự thực thi, kiến trúc điều khiển nguyên thuỷ, sự tóm tắt, sự ủng hộ đồng thời, và nhiều thứ khác. Ada 95 trình bày một sự nâng cấp ngôn ngữ có kế hoạch tốt để điều tiết công nghệ mới và kết hợp các bài đã học trong một khía cạnh ứng dụng. Điều khác biệt trong tính có thể xúc cảm giữa Ada 83 và Ada 95 là thừa hưởng chủ yếu để những nét đặc trưng được thêm vào để bổ xung chương trình hướng đối tượng. Theo đó, một sự ước lượng giá trị của chương trình hướng đối tượng có thứ tự đầu tiên mà nó cho phép các chương trình được viết giảm đi 30% dòng nguồn.*

Sự khác biệt giữa C và C++ thậm chí còn sâu sắc hơn. C++ kết hợp một vài (mặc dù không kết hợp toàn bộ) với sự hỗ trợ các tiến bộ tốt của Ada cho chương trình hướng đối tượng. Tuy nhiên, C++ cũng đã phát triển để hỗ trợ C như một bộ con. Yêu cầu này có chiều thuận và chiều nghịch. Trên một phương diện, tính tương hợp của C làm cho nó rất rễ đối với những nhà lập trình C chuyển sang C++. Trên một phương diện khác, một xu hướng đáng chú ý trong công nghiệp là một số lượng đông đảo các nhà lập trình sử dụng bộ dịch C++ nhưng bộ óc chương trình là C, vì thế thiếu sót này chấp nhận tính nhạy cảm của C++ hướng đối tượng. Sự tiến triển của Java đã loại trừ nhiều vấn đề của ngôn ngữ C++ (đặc trưng tự nhiên là hỗ trợ đối với C, hỗ trợ một vài chương trình thực hành nguy hiểm) trong khi bảo vệ các đặc trưng hướng đối tượng và cộng thêm những hỗ trợ xa hơn đối với tính khả chuyển và sự phân phối.

Toàn bộ các điểm chức năng có thể được sử dụng để chỉ ra kích thước chương trình quan hệ được yêu cầu được thực thi một chức năng đưa ra. Ví dụ, để đạt được một ứng dụng được đưa ra với một số lượng hỗn hợp các điểm chức năng, một kích thước chương trình như sau có thể đòi hỏi:

1.000.000 dòng hợp ngữ

400.000 dòng ngôn ngữ C

220.000 dòng ngôn ngữ Ada 83

175.000 ngôn ngữ Ada 95 hoặc C++.

Các giá trị biểu thị mối quan hệ nhạy cảm được đưa ra bởi nhiều ngôn ngữ khác nhau. Các thành phần thương mại và các bộ sinh mã tự động (như là công cụ CASE và những người xây dựng GUI) có thể làm giảm nhiều hơn kích thước chương trình nguồn sinh ra bởi con người, lần lượt giảm kích thước đội ngũ làm dự án và thời gian cần cho sự phát triển. Mở rộng ví dụ này ra, thêm vào một hệ quản trị cơ sở dữ liệu thương mại, người xây dựng GUI thương mại, và trung gian thương mại có thể làm giảm kích thước kết quả của sự phát triển xuống kích thước cuối cùng sau đây:

75000 dòng ngôn ngữ Ada 95 hoặc C++ cộng với sự tích hợp của một vài thành phần thương mại.

Bởi vì sự khác nhau giữa các dự án lớn và dự án nhỏ là lớn hơn đường tác động trong vòng đời chi phí, việc sử dụng ngôn ngữ bậc cao nhất và để dành các phần tử thương mại có một tiềm năng tác động lớn đối với chi phí. Hơn nữa, đơn giản hơn là tính tổng quan tốt hơn: Sự giảm kích thước thường làm tăng khả năng có thể hiểu được, tính có thể thay đổi được, và tính có thể tin cậy được. Một kết quả mặt trái điển hình là mức trừu tượng cao hơn các công nghệ hướng về sự suy biến hiệu năng, tăng sự tiêu thụ tài nguyên như là chu kì bộ xử lý, bộ nhớ, và bề dày các băng thông tin. Hầu hết những trở ngại này sẽ được vượt qua bởi sự cải tiến hiệu năng phần cứng và một cái nhìn lạc quan. Những cải tiến này kém xa hiệu quả trong các nền nhúng.

3.1.2.Các Phương pháp hướng đối tượng và mẫu trực quan

Đã có một sự chuyển động rộng lớn trong những năm 90 hướng tới công nghệ hướng đối tượng. Tôi sử dụng rất ít thời gian trong chủ đề này bởi vì công nghệ hướng đối tượng không thích hợp với hầu hết các chủ đề quản lý phần mềm được bàn luận ở đây, và có rất nhiều sách nói về công nghệ hướng đối tượng. Một vài nghiên cứu đã kết luận rằng sự xuất hiện các ngôn ngữ chương trình hướng đối tượng đã làm lợi cho cả các sản phẩm phần mềm và chất lượng phần mềm [Jones, 1994], nhưng lợi ích kinh tế vẫn chưa được chứng minh bởi vì chi phí sườn dốc của việc đào tạo trong các phương pháp thiết kế hướng đối tượng như Ngôn ngữ mẫu kết hợp (UML - Unified Modeling Language).

Từ việc cung cấp nhiều ký hiệu chính thức hoá hơn cho việc nắm được và hình dung được sự trừu tượng của phần mềm, tác động nền tảng của công nghệ hướng đối tượng là trong việc giảm toàn bộ kích thước của những gì cần được phát triển. Booch đã mô tả 3 lý do khác mà chắc chắn các dự án hướng đối tượng sẽ thành công [Booch, 1996]. Có một ví dụ thú vị về những mối tương quan giữa các chiều hướng cải tiến kinh tế phần mềm. (Các đoạn được trình bày in nghiêng.)

1. *Một vấn đề về mô hình hướng đối tượng và giải pháp của nó khuyến khích một từ vựng chung giữa những người dùng cuối cùng của hệ thống và những người phát triển nó, do đó việc tạo ra một sự chia sẻ kiến thức về vấn đề là cần được giải quyết.*

- Đây là một ví dụ về công nghệ hướng đối tượng nào đó cho phép những sự cải tiến phù hợp trong việc kết hợp (chung sức) và thông tin giữa các cá nhân với nhau.

1. *Việc sử dụng sự tích hợp tiếp theo tạo nên sự hài hoà để nhận ra rủi ro sớm và làm tăng những sửa chữa thiếu tính không ổn định toàn vẹn công sức phát triển.*

- Khía cạnh này của công nghệ hướng đối tượng cho phép một kiến trúc tiến trình đầu tiên, nơi mà sự tích hợp là một vòng đời hoạt động sớm và một vòng đời tiếp theo.

2. *Một kiến trúc hướng đối tượng đưa ra một sự phân chia rõ ràng về sự quan tâm lo lắng giữa các thành phần khác hẳn nhau của hệ thống, tạo nên bức tường lửa chống lại sự thay đổi trong từng phần của hệ thống từ việc phá nát kết cấu của kiến trúc toàn vẹn.*

- Đặc trưng này của công nghệ hướng đối tượng là tính quyết định cho việc ủng hộ các ngôn ngữ, và các máy tính sẵn có để thực thi các kiến trúc hướng đối tượng.

Booch cũng tổng kết 5 đặc điểm của một dự án hướng đối tượng thành công.

1. *Một trọng tâm tàn nhẫn (ruthless) trong sự phát triển hệ thống là đưa ra một sự lựa chọn kiến thức tốt của các đặc điểm tối thiểu cần thiết.*
2. *Sự tồn tại của một nền văn hoá là trung tâm trong các kết quả, khuyến khích thông tin, và vẫn chưa sợ thất bại.*
3. *Hiệu quả sử dụng mô hình hướng đối tượng.*
4. *Sự tồn tại của một sức nhìn kiến trúc mạnh.*
5. *ứng dụng của một quản lý lặp lại và việc gia tăng vòng đời phát triển tốt.*

Các đặc điểm này có ít điều phải làm với sự định hướng đối tượng. Tuy nhiên, các phương pháp hướng đối tượng, các kí hiệu và mẫu trực quan đưa ra công nghệ mạnh trợ giúp cho tiến trình khung công việc.

3.1.3.Tái sử dụng

Việc tái sử dụng các thành phần đang có và xây dựng các thành phần có thể tái sử dụng là các hoạt động kỹ thuật phần mềm tự nhiên kể từ khi có những sự cải tiến sớm nhất trong các ngôn ngữ chương trình, các phương pháp thiết kế phần mềm luôn luôn giải quyết một cách tuyệt đối việc tái sử dụng có trật tự để tối thiểu hoá các chi phí phát triển, trong khi đạt được tất cả các thuộc tính yêu cầu khác của hiệu năng, bộ đặc tính, và chất lượng. Việc tái sử dụng đạt được không chỉ đáng quan trọng trong cộng đồng (community) bởi vì chúng ta không làm nó tốt như chúng ta nên làm. Trong toàn bộ các kỹ thuật và kỉ luật sản xuất khác, việc tái sử dụng là nhiều hơn hay ít hơn một giả định ở mức dưới, không cần thiết phải phá bỏ cản trở công nghệ. Tôi cố gắng đề cập tái sử dụng như một phần trần tục của việc đạt được kết quả trong đầu tư. Các kiến trúc chung, các tiến trình chung, kinh nghiệm tiền lệ, và các môi trường chung là tất cả những thí dụ của việc tái sử dụng.

Một trong những trở ngại lớn nhất để tái sử dụng là sự vỡ ra từng mảnh của các ngôn ngữ, của hệ điều hành, các kí hiệu, các kiến trúc máy, các công cụ, và thậm chí các "chuẩn." Như một ví dụ trái ngược, mức độ tái sử dụng được tạo ra có thể do sự thành công của Microsoft trên máy PC đã rất rộng lớn.

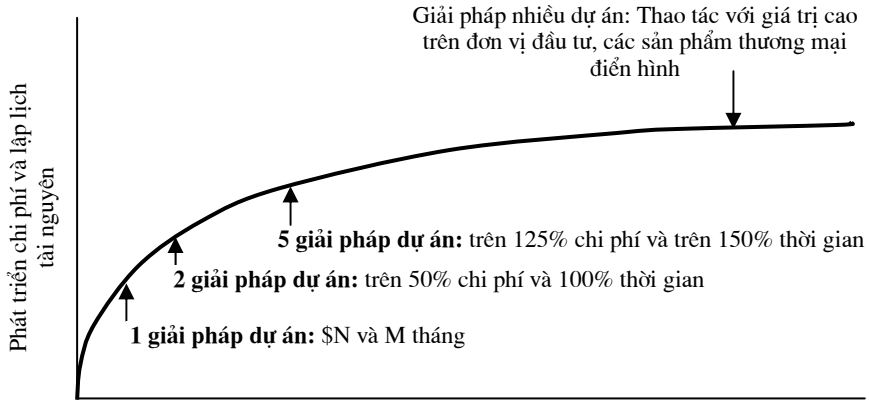
Nhìn chung, nhiều thứ cho phép được tái sử dụng vì các lý do kinh tế. Bởi thế, độ đo chính trong việc nhận diện hoặc một thành phần (hoặc một lớp các thành phần, hoặc một sản phẩm thương mại) đúng là có thể tái sử dụng là để thấy hoặc là một vài tổ chức đang tạo ra tiền bạc trên nó. Không có sự vận động kinh tế này, các thành phần tái sử dụng là rất hiếm. Chú ý việc "mở" các thư viện tái sử dụng được đỡ đầu bởi các tổ chức phi lợi nhuận. Chúng thiếu sự chuyển động kinh tế, tính chất đáng tin cậy, và trách nhiệm giải trình đối với chất lượng, sự ủng hộ, sự cải tiến, và tính sử dụng được. Hầu hết các giá trị của thành phần thực sự có thể tái sử dụng được là việc chuyển thành các sản phẩm thương mại được hỗ trợ bởi các tổ chức với các đặc điểm sau:

- Chúng có một sự chuyển động kinh tế đối với sự ủng hộ liên tục.

• Chúng làm chủ sở hữu sự cải tiến chất lượng sản phẩm, thêm các đặc tính mới, và chuyển sang các công nghệ mới.

• Chúng có một khách hàng rộng lớn đủ làm cơ sở để có thể có lợi nhuận.

Chi phí phát triển một thành phần có thể tái sử dụng là không tầm thường. Hình 3-1 kiểm tra việc trả giá. Đường cong dốc ban đầu minh hoạ trở lực kinh tế đối với sự phát triển các thành phần có thể tái sử dụng. Sẽ khó để phát triển một nhánh thương mại có tính thuyết phục đối với sự phát triển trừ khi mục tiêu là việc ủng hộ tái sử dụng qua nhiều dự án. Các nhánh thương mại lạc quan hiếm khi xảy ra trong các tổ chức phát triển phần mềm mà không được chú trọng trong việc bán các thành phần thương mại như đường thương mại chính của chúng. Hầu hết các tổ chức không thể tham gia tính kinh tế với các tổ chức thương mại đã thiết lập đầu tư của họ được hoàn trả dần qua người dùng cơ sở. Để đem lại thành công trên thương trường cho các thành phần thương mại, một tổ chức cần 3 yếu tố lâu dài: một nhóm phát triển, một cơ sở hạ tầng hỗ trợ, và một bày bán hướng sản phẩm và một thị trường cơ sở hạ tầng. Một sự ân cần khác là tính phức tạp và chi phí phát triển các thành phần có thể tái sử dụng được thường được đánh giá thấp một cách chất phác.



Hình 3-1.Chi phí và lịch trình đầu tư cần thiết để đạt được các thành phần có thể tái sử dụng được.

Mặc dù giá trị tái sử dụng có thể rất rộng lớn, tôi chưa bao giờ là một cổ động viên của việc định danh tái sử dụng như là một ngăn cách "công nghệ". Việc tái sử dụng là một kỷ luật quan trọng, nó có tác động vào khả năng của toàn bộ luồng công việc và chất lượng của hầu hết các tạo tác. Tôi nghĩ nó như sự đồng bộ đối với thành quả trong đầu tư, điều mà sẽ là một sự ân cần trong hầu hết mọi hoạt động và quyết định. Có một vài câu chuyện rất thành công trong việc tái sử dụng các thành phần phần mềm ngoại trừ đối với các sản phẩm thương mại như các hệ điều hành, hệ quản trị cơ sở dữ liệu, bộ trung gian, mạng, người xây dựng GUI, và các ứng

dụng văn phòng. Trên một phương diện khác, mọi câu chuyện thành công về phần mềm, có lẽ đã khai thác một vài cách tiếp cận chính của việc tái sử dụng (không có việc gọi như vậy) để đạt được những kết quả khả quan.

3.1.4.Các thành phần thương mại

Một cách tiếp cận chung đã được theo đuổi ngày nay trong nhiều miền là sự tối đa hoá sự tích hợp các thành phần thương mại và các sản phẩm không lỗi thời. Trong khi việc sử dụng các thành phần thương mại chắc chắn là đáng khao khát như một phương tiện của việc giảm sự phát triển theo tập quán cũ, nó đã không được chứng minh thẳng thắn trong thực hành. Bảng 3-3 định danh một vài ưu điểm và nhược điểm của việc sử dụng các thành phần thương mại (những trả giá này là là đặc thù khắc nghiệt trong các miền giới hạn nhiệm vụ.). Bởi vì trả giá thường xuyên có hiệu lực rộng lớn trong chất lượng, chi phí, tính có thể hỗ trợ được, và sự lựa chọn các thành phần thương mại qua sự phát triển của các thành phần tập quán (cũ-custom) có ý nghĩa tác động vào toàn bộ kiến trúc của một dự án. Thông báo cực cao ở đây (được bàn luận xâu hơn ở trong chương 7) là những quyết định này phải được tạo ra sớm trong vòng đời như một phần của kiến trúc thiết kế.

Bảng 3-3. *Các ưu điểm và nhược điểm của các thành phần thương mại chống lại phần mềm truyền thống.*

Cách tiếp cận	**Ưu điểm**	**Nhược điểm**
Các thành phần thương mại	Khả năng dự đoán đăng ký chi phí Được sử dụng rộng rãi, công nghệ chín muồi Sẵn có ngày nay Tổ chức ủng hộ tận tâm Phần cứng/ Phần mềm độc lập Sự phong phú của chức năng	Thường xuyên nâng cấp Trả trước đăng kí thù lao Định kỳ bảo trì thù lao Tính phụ thuộc vào người bán Hy sinh hiệu xuất thời gian chạy Sự ép buộc chức năng Sự tích hợp luôn không tầm thường Không điều khiển trên những nâng cấp và bảo trì Các đặc trưng không cần thiết tiêu thụ hết các tài nguyên phụ Thường không đầy đủ độ tin cậy và độ ổn định Tính không tương hợp đa người bán
Tập quán phát triển	Hoàn thiện thay đổi tự do Nhỏ hơn, thường đơn giản hơn sự thực thi Hiệu năng thường tốt hơn Điều khiển sự phát triển và nâng cao	Đắt, không có khả năng dự đoán sự phát triển Không có khả năng dự đoán ngày sẵn có Không định nghĩa được mô hình bảo trì Thường không chín muồi và dễ hỏng Tiêu phí nguồn tài nguyên chuyên môn

3.2.Cải tiến các tiến trình phần mềm

Tiến trình là một thời kỳ đặt quá tải. Đối với các tổ chức hướng phần mềm, có nhiều tiến trình và tiến trình con. Tôi sử dụng 3 tiến trình riêng biệt có triển vọng.

- *Siêu tiến trình* (metaprocess): những kiểm soát, thủ tục, và thực hành của một tổ chức đối với sự theo đuổi một tuyến phần mềm chuyên sâu trong thương mại. Trọng tâm của tiến trình này là trong tổ chức kinh tế, các chiến lược dài hạn, và một ROI phần mềm.
- *Đại tiến trình* (macroprocess): những kiểm soát, thủ tục, và thực hành của một dự án đối với việc sản xuất một phần mềm hoàn thiện, ở trong chi phí chắc chắn, lịch trình và sự cưỡng ép chất lượng. Trọng tâm của đại tiến trình là trong việc tạo ra một thí dụ đầy đủ của siêu tiến trình đối với một bộ ép buộc cụ thể.
- Tiểu tiến trình (microprocess): những kiểm soát, thủ tục, và thực hành của một đội làm dự án đối với việc đạt được một tạo tác của tiến trình phần mềm. Trọng tâm của tiểu tiến trình là trong việc đạt được một đường danh giới sản phẩm trung gian với đầy đủ chất lượng và đầy đủ chức năng về phương diện kinh tế và nhanh chóng đi vào thực hành.

Mặc dù 3 mức tiến trình chồng chéo lên nhau một phần nào đó, nhưng chúng có các mục tiêu, độc giả, độ đo, sự liên quan và nấc thang thời gian khác nhau như trình bày trong bảng 3-4. Đại tiến trình là tiến trình mức dự án mà hiệu quả mô hình chi phí ước lượng đã được bàn luận trong chương này.

Để đạt được sự thành công, hầu hết các dự án phần mềm đều yêu cầu một mớ phức tạp khó tin nổi của các bước nối tiếp và song song. Như tỉ lệ của một dự án tăng lên, nhiều bước trên phải bao gồm việc quản lý mớ phức tạp này. Tất cả các tiến trình dự án trùng khớp của các hoạt động sản xuất và hoạt động trên. Kết quả các hoạt động sản xuất trong các tiến trình thực tế hướng tới sản phẩm cuối cùng. Đối với các công sức phần mềm, các hoạt động này bao gồm tạo mẫu, mô hình hoá, mã hoá, gỡ rối, và cung cấp tài liệu người dùng. Các hoạt động trên có một tác động thực tế vào sản phẩm cuối cùng đã được yêu cầu trong sự chuẩn bị kế hoạch, sự cung cấp tài liệu, tiến trình giám sát, sự ước định rủi ro, sự ước định tài chính, điều khiển cấu hình, sự ước định chất lượng, tích hợp, kiểm thử, phế thải muộn (loại bỏ những thứ không tốt ở giai đoạn muộn) và sửa lại (thiết kế lại), quản lý, đào tạo nhân lực, điều hành thương mại, và nhiều nhiệm vụ khác. Các hoạt động trên bao gồm nhiều nỗ lực thêm các giá trị, nhưng nhìn chung, nỗ lực (công sức) nhỏ hơn đã cống hiến cho các hoạt động này, nỗ lực lớn hơn có thể được dùng (tiêu tốn) trong các hoạt động sản xuất. Mục tiêu của việc cải tiến tiến trình là tối đa hoá việc phân phối các tài nguyên cho các hoạt động sản xuất và tối thiểu hoá tác động của các hoạt động trên vào các tài nguyên như nguồn nhân lực, máy tính và lịch trình.

Bảng 3-4**.** *Ba mức tiến trình và các thuộc tính của nó.*

Thuộc tính	Siêu tiến trình	Đại tiến trình	Tiểu tiến trình
Chủ đề	Tuyến thương mại	Dự án	Tích hợp
Mục tiêu	Lợi nhuận tuyến thương mại Tính cạnh tranh	Lợi ích dự án Quản lý rủi ro Ngân sách dự án,lịch trình, chất lượng	Quản lý tài nguyên Sự giải quyết rủi ro Mốc (quan trọng) ngân sách, lịch trình, chất lượng
Độc giả	Giành được nhà cầm quyền, khách hàng Quản lý tổ chức	Những nhà quản lý dự án phần mềm Các kỹ sư phần mềm	Những nhà quản lý dự án con Các kỹ sư phần mềm
Độ đo	Khả năng dự đoán dự án Thu nhập, chia sẻ thị trường	Trong ngân sách, trong lịch trình Mốc thành công chính Phế thải dự án và sửa lại	Trong ngân sách, trong lịch trình Mốc tiến bộ chính Giải thoát/tích hợp phế thải và sửa lại
Lợi lộc (liên quan)	Thói quan liêu chống lại sự tiêu chuẩn hoá	Chất lượng chống lại hiệu năng tài chính	Nội dung chống lại lập lịch
Tỉ lệ thời gian	6 đến 12 tháng	1 đến nhiều năm	1 đến 6 tháng

Một vài người có thể bị làm bực dọc bởi sự phân loại của tôi về phế thải muộn, việc sửa lại và đào tạo nhân lực như các hoạt động trên cần được tối thiểu hoá. Tôi đã giảm bớt phế thải và sửa lại *muộn* để tạo sự khác biệt giữa nó với phế thải và sửa lại thường, phế thải và sửa lại thường là một sản phẩm tư nhiên của công sức tạo mẫu ban đầu (mẫu gốc). Phế thải và sửa lại sớm là một quá trình xảy ra tất yếu đối với hầu hết các dự án để giải quyết vô số những điều xa lạ (không ai biết) trong không gian giải pháp, nhưng rõ ràng là không ai ưa thích nó trong những pha muộn hơn của vòng đời. Với một tiến trình tốt, nó rõ ràng là không cần thiết.

Người ta sẽ khẳng định rằng đào tạo nhân lực không thể là một điều tồi tệ, nhưng đó là đối với dự án. Việc đào tạo là một trách nhiệm tổ chức, không phải là trách nhiệm dự án. Bất kì một nhà quản lý dự án nào mà mang gánh nặng đào tạo con người trong các tiến trình, các công nghệ, và các công cụ sẽ thấy tồi hơn một nhà quản lý dự án mà đã được đào tạo công việc đầy đủ. Việc cung cấp cho dự án một sự đào tạo công việc đầy đủ có thể là điều không thể thực hiện nhưng công việc đào tạo con người luôn tốt hơn việc không đào tạo con người, những điều khác giữ nguyên. Trong cảm nhận này việc đào tạo không được xem là một hoạt động thêm giá trị.

Chất lượng của tiến trình dự án khoẻ ảnh hưởng đến công sức đòi hỏi và vì thế ảnh hưởng tới lịch trình đối với việc sản xuất phần mềm, theo lệ thường sự khác biệt giữa một tiến trình tốt và và một tiến trình tồi sẽ ảnh hưởng trên toàn bộ chi phí ước lượng khoảng 50% đến 100%, và sự giảm nỗ lực sẽ cải tiến toàn bộ lịch trình. Hơn nữa, một tiến trình tốt hơn thậm chí có thể có một hiệu quả lớn hơn trong việc giảm thời gian nó sẽ mạng lại cho những người làm dự án có được cách nhìn sản phẩm với chất lượng yêu cầu. Tại sao điều này lại đúng?

Lập lịch cải tiến có ít nhất 3 hướng.

1. Chúng ta có thể làm một tiến trình có N bước và cải tiến hiệu xuất của từng bước.
2. Chúng ta có thể làm một tiến trình có N bước và loại bỏ một số bước và vì thế giờ đây nó chỉ còn M bước.
3. Chúng ta có thể làm một tiến trình có N bước và sử dụng nhiều hơn sự hài hoà trong các hoạt động được thực hiện hoặc các tài nguyên được ứng dụng.

Nhiều lần tổ chức chiến lược cải tiến thị trường đã nhấn mạnh hướng thứ nhất. Tuy nhiên, trọng tâm của hầu hết các cải tiến tiến trình được miêu tả trong cuốn sách này là đạt được theo hướng thứ 2 và thứ 3, là nơi có tiềm năng lớn hơn. Đặc biệt, trọng tâm chính của sự cải tiến tiến trình sẽ đạt được một giải pháp thoả đáng trong số tối thiểu của sự lặp lại và loại bỏ nhiều phế thải và sửa đổi xuôi dòng như có thể.

Mọi ví dụ của việc sửa đổi mở đầu một bộ có trình tự các nhiệm vụ phải làm lại. Ví dụ, giả sử rằng một nhóm hoàn thành trình tự các bước phân tích, thiết kế, lập trình, và kiểm thử các đặc tính, sau đó phát hiện một luồng thiết kế trong kiểm thử. Giờ đây một trình tự thiết kế lại, lập trình lại, và kiểm thử lại là một sự đòi hỏi. Dãy các nhiệm vụ này là sự cản trở chính đối với việc nén lịch trình. Mặc dù việc chọc thủng phòng tuyến công nghệ có thể loại bỏ các bước tiến trình hoàn thiện, nhưng tác động chính của sự cải tiến tiến trình sẽ làm cho phế thải và sửa chữa giảm xuống trong các pha muộn của vòng đời.

Trong một thế giới kỹ thuật phần mềm hoàn thiện với một sự miêu tả vấn đề thuần khiết, một không gian giải pháp rõ ràng, một đội ngũ phát triển có kinh nghiệm tài năng, một tài nguyên đầy đủ, và những cổ đông có các mục tiêu chung, chúng ta có thể thực hiện một tiến trình phát triển phần mềm bằng một lần lặp lại mà hầu như không có phế thải và sự sửa chữa. Bởi vì chúng ta làm việc trong một thế giới không hoàn thiện, tuy nhiên, chúng ta cần quản lý các hoạt động kỹ thuật vì vậy phế thải và sửa chữa sơ lược không có một tác động nào trong việc thắng các điều kiện của bất kì một cổ đông nào. Đây là giả thuyết cơ sở cho hầu hết các sự cải tiến tiến trình.

3.3.Cải tiến hiệu quả nhóm làm dự án

Đã có một sự hiểu biết sâu sắc khác nhau trong đội ngũ những người lao động đối với sự giao động trong năng xuất. Ví dụ trong mô hình COCOMO gốc đề xuất rằng hiệu qủa liên kết của đội ngũ lao động lành nghề và kinh nghiệm có thể có tác động vào năng xuất nhiều như

nhân tố thứ tư. Đây là điều khác biệt giữa một nhóm lao động nghiệp dư không có tay nghề cao với một nhóm lao động chuyên môn kì cựu. Trong thực hành, nó là tính rủi ro để đánh giá một nhóm đưa ra như làm hạ tỉ lệ trong cả 2 hướng. Một nhóm lớn, nói rằng, 50 người hoặc nhiều hơn, anh hầu như luôn giữ vững lập trường với danh nghĩa mọi người và kinh nghiệm. Không thể để nhân viên một dự án không tầm thường với toàn bộ nhân viên mà tất cả đều có kinh nghiệm tối ưu, đã được đào tạo đầy đủ về các công cụ, các công nghệ và có chỉ số IQs lớn hơn 130. Nếu bạn kéo điều này xuống thì nhóm có lẽ sẽ bị suy yếu chức năng. Vì vậy cách tiếp cận "vừa thuê nhân viên tốt" cũ cần được áp dụng một cách cẩn thận. Một cách tốt hơn để phát biểu đó là "Vừa trình bày rõ ràng một nhóm tốt."

Sự cân bằng và phạm vi bao hàm là 2 trong những lĩnh vực quan trọng nhất của những nhóm xuất sắc. Bất kì khi nào, một nhóm mất cân bằng, thì nó sẽ dễ bị tổn thương. Để sử dụng một môn thể thao tương tự, một đội bóng đá có một sự cần thiết cho những tay lành nghề thay đổi khác nhau, rất nhiều điều giống như một nhóm phát triển phần mềm. Hiếm khi một đội bóng lớn mà không có phạm vi bao hàm lớn: sự vi phạm, sự bảo vệ, và những nhóm chuyên biệt, sự huấn luyện và đội ngũ nhân lực, những xà ngang đầu tiên và các cầu thủ dự bị, chuyển qua và chạy. Những đội lớn cần phạm vi bao hàm trên các vị trí chính với những cá thể cầu thủ khoẻ mạnh. Nhưng một đội có các siêu sao, tất cả đều phấn đấu để đặt các hồ sơ cá nhân và tham gia đội ngũ dẫn dắt, có thể bị làm xấu hổ bởi một đội cân bằng của những cầu thủ vững vàng với một số người dẫn dắt trọng tâm vào kết quả chiến thắng của đội trong trò chơi (trận đấu).

Sự cộng tác quan trọng hơn nhiều so với tổng từng cá thể riêng lẻ. Với các nhóm làm phần mềm, một nhà quản lý dự án cần sắp xếp thành hình thể cân bằng một tài năng với những con người có tay nghề cao vào các vị trí đòn bẩy. Một vài châm ngôn của nhóm quản lý bao gồm như sau:

- Một dự án quản lý tốt có thể thành công với danh nghĩa nhóm kỹ thuật.
- Một dự án quản lý tồi sẽ hầu như không bao giờ thành công thậm chí với một nhóm các kỹ sư chuyên môn.
- Một hệ thống kiến trúc tốt có thể được xây dựng với danh nghĩa một nhóm những nhà xây dựng phần mềm.
- Một hệ thống kiến trúc nghèo nàn sẽ bị lúng túng thậm chí với một nhóm những nhà xây dựng chuyên môn.

Trong việc kiểm tra như thế nào đó đối với nhân viên một dự án phần mềm, Boehm đã đưa ra sau đây 5 nguyên tắc bố trí cán bộ nhân viên [Boehm, 1981]. (Đoạn trình bày in *nghiêng*.)

1. Nguyên tắc tài năng đỉnh cao: Sử dụng con người tốt hơn và ít hơn.

➢ Nguyên lý này là thiết yếu, nhưng nó có thể chỉ áp dụng cho đến nay. Đó là một kích thước nhóm "tự nhiên" cho hầu hết các công việc, và nó không tinh (thô) trên hoặc dưới kích thước này đều tồi đối với động lực của nhóm bởi vì nó cho kết quả quá ít hoặc quá nhiều áp lực lên các cá thể để thực hiện.

2. Nguyên tắc côngviệc phù hợp: Hoà hợp các nhiệm vụ với các tay nghề và thúc đẩy những con người sẵn có.

➢ Nguyên tắc này dường như rõ ràng. Trong một đội bóng đá bạn sử dụng một người dẫn dắt tốt như huấn luyện viên của bạn, một người biết vượt qua tốt như tiền vệ, một người chạy cực nhanh như là một người thu nhận rộng rãi, và một người béo nặng 300 pound như một người gác đường dây(trọng tài biên). Với các kỹ sư phần mềm càng khó hơn để phân biệt phần lớn sự vô hình trong toàn bộ nhân viên có tay nghề và sự phân phối nhiệm vụ một cách tối ưu. Những công việc phải làm riêng biệt cũng làm phức tạp thêm các nhiệm vụ. Trong bóng đá, một người gác đường dây 300 pound sẽ không bao giờ nghĩ về việc thăng chức thành tiền vệ, các khả năng tay nghề khác nhau quá rõ ràng. Tuy nhiên trong nhóm làm phần mềm, có cái chung cho những nhà lập trình tài năng để tìm sự thăng tiến thành những kiến trúc sư và những nhà quản lý. Tôi nghĩ khả năng tay nghề là sự bình đẳng khác nhau, bởi vì hầu hết các siêu sao lập trình đều thiếu khả năng bẩm sinh để làm những kiến trúc sư và những nhà quản lý, và những nhà thay thế đa năng. ấy thế mà những cá thể riêng biệt và thậm chí cả tổ chức của họ thường quan niệm sự thăng tiến như một cái có thể khao khát. Có vô số nhánh của các kỹ sư phần mềm lớn sẽ được thăng tiến trước hạn định (khi họ chưa đủ độ chín muồi) vào các vị trí mà họ không đủ tay nghề và thiếu khả năng. Việc quyết định đưa cầu thủ B ra thay thế cầu thủ A, nói rằng cầu thủ A ra ngoài để điều tiết vị trí đòn bẩy cao và cầu thủ B vào một vị trí đòn bẩy cao hơn. Đó là một tiếng vang gây bất ngờ gấp đôi.

3. Nguyên tắc sự tiến bộ nghề nghiệp: Một tổ chức làm tốt nhất trong cuộc chạy dài hơi bởi những sự giúp đỡ của những con người có tính hoạt động tự giác.

➢ Một người thực hiện tốt thường có tính hoạt động tự giác trong bất kì môi trường nào. Các tổ chức có thể giúp đỡ và cản trở sự tự giác của người lao động, nhưng năng lực tiềm tàng của tổ chức có lợi trung bình và dưới hầu hết những người thực hiện trung bình. Các chương trình đào tạo tổ chức là chiến lược công cuộc kinh doanh điển hình với giá trị giáo dục. Các chương trình đào tạo dự án hoàn toàn là chiến thuật, đã có ý định hữu ích và ứng dụng ngay sau khi việc đào tạo kết thúc.

4. Nguyên tắc cân bằng nhóm: Chọn người mà sẽ bổ sung và làm hài hoà với người khác.

➢ Mặc dù nguyên tắc này báo hiệu có một chút tính nhỏ rọt, nhưng tinh thần của nó là nhân tố tối cao trong sự phối hợp tốt. Sự cân đối của nhóm làm phần mềm có nhiều hướng, và khi một nhóm bất kỳ nào của dự án không cân đối, thì dự án trở nên nghiêm trọng trong rủi ro. Các định hướng này bao gồm:

Những tay nghề thô: sự thông minh, tính mục đích, tính tạo lập, sự tổ chức, sự suy nghĩ phân tích

Tính chất tâm lý: những người dẫn dắt và những người theo dõi, những người đánh cuộc và những người bảo thủ rủi ro, những người nhìn xa trông rộng và những người soi mói, những người hoài nghi và những người lạc quan.

Các mục tiêu: tài chính, thiết lập đặc tính, chất lượng, tính chất hợp thời.

5. Nguyên tắc rút lui dần dần: Việc giữ lại một sự không phù hợp trong nhóm không có lợi cho bất kỳ cái gì.

➢ Đây thực sự là nguyên tắc con của nguyên tắc thứ tư. Một sự không phù hợp đưa cho anh một lý do để tìm người tốt hơn hoặc để sống với ít người hơn. Một sự không phù hợp sẽ không thúc đẩy các thành viên nhóm khác, sẽ không hoạt đông tự giác, và phá vỡ nhóm cân bằng trong một vài hướng. Sự không phù hợp là rõ ràng, và nó hầu như không bao giờ có quyền làm trì hoãn việc loại bỏ chúng đi.

Sự phát triển phần mềm là một đội thể thao. Những nhà quản lý phải nuôi dưỡng văn hóa cộng tác có kết quả tốt hơn thành tựu cá nhân. Trong 5 nguyên tắc, cân bằng nhóm và sự phù hợp nghề sẽ là mục tiêu chính. Nguyên tắc tài năng đỉnh cao và nguyên tắc rút lui dần dần là mục tiêu thứ hai bởi vì chúng phải được ứng dụng trong hoàn cảnh của nhóm cân bằng. Cuối cùng, mặc dù sự tiến bộ nghề nghiệp cần địa chỉ như một công việc thực tập, các cá nhân hay tổ chức mà nhấn mạnh nó qua sự thành công của nhóm thì sẽ không kéo dài trên các vùng thị trường.

Người quản lý dự án phần mềm cần nhiều sự lãnh đạo có chất lượng trong việc sắp xếp để nâng cao hiệu quả của nhóm. Mặc dù chất lượng nói trên là vô hình và ở bên ngoài phạm vi của cuốn sách này, tôi sẽ sao lãng nếu tôi không đề cập đến chúng. Sau đây là một vài thuộc tính quyết định của những nhà quản lý dự án phần mềm thành công mà đáng được chú ý nhiều hơn:

1. Các kỹ năng thuê mướn: Một số ít các quyết định quan trọng như việc thuê quyết định. Sắp xếp đúng con người vào đúng công việc dường như là rất rõ ràng nhưng là sự vất vả đáng ngạc nhiên để đạt được nó.

2. Kỹ năng giao tiếp khách hàng: Việc tránh các mối quan hệ đối đầu giữa các cổ đông là một điều kiện tiên quyết cho sự thành công.

3. Kỹ năng ra quyết định: Hàng nghìn cuốn sách viết về quản lý đã có thiếu sót để đưa ra một định nghĩa rõ ràng về thuộc tính này. Tất cả chúng ta đều biết một người lãnh đạo tốt khi chúng ta chạy tới một, và kỹ năng ra quyết định dường như rõ ràng mặc dù sự định nghĩa của nó là vô hình.

4. Kỹ năng xây dựng nhóm: sự cộng tác yêu cầu một nhà quản lý thiết lập sự tin cậy, thúc đẩy tiến bộ, khai thác không đúng trọng tâm (phân chia vai trò cho mỗi người không tốt), chuyển những con người bình thường thành những con người có hiệu năng đỉnh cao, loại bỏ những cái không phù hợp, và củng cố các ý kiến khác nhau thành một nhóm giám sát.

5. Kỹ năng bán: Những nhà quản lý dự án thành công phải bán toàn bộ các cổ đông (bao gồm cả chính họ) trong các quyết định và các ưu tiên, bán những người ứng cử vào các vị trí công việc, bán từ sự thay đổi đến hiện trạng bên trong bề mặt của sự chống đối, và bán cả các mục tiêu chống đối đạt được. Trong thực hành, việc bán yêu cầu tiếp tục sự đàm phán, sự thoả hiệp và sự đồng cảm.

3.4.Cải tiến kỹ thuật tự động hoá qua các môi trường phần mềm

Các công cụ và môi trường được sử dụng trong tiến trình phần mềm nhìn chung có một đường thẳng kết quả trong năng xuất của tiến trình. Các công cụ kế hoạch, các công cụ yêu cầu quản lý, các công cụ mẫu trực quan, các bộ dịch, những người biên tập, những người gỡ rối, các công cụ phân tích bảo đảm chất lượng, các công cụ kiểm thử, và giao diện người dùng cung cấp sự hỗ trợ kỹ thuật tự động hoá quyết định đối với sự tiến triển của các tạo tác kỹ thuật phần mềm. Trên tất cả, các môi trường quản lý cấu hình cung cấp nền móng cho việc thi hành và dụng cụ trong tiến trình. ở cách sắp xếp đầu tiên, sự cô lập tác động của các công cụ và kỹ thuật tự động hoá nói chung cho phép cải tiến 20% đến 40% công sức. Tuy nhiên, các công cụ và các môi trường phải được nhìn nhận như phương tiện phân phối chính cho tiến trình kỹ thuật tự động hoá và cải tiến vì thế tác động của chúng có thể cao hơn nhiều.

Phần 2.3 tập trung vào sự cải tiến tiến trình nhằm làm giảm phế thải và việc sửa lại, bởi vậy đã loại bỏ các bước và tối thiểu hoá số lần lặp lại trong tiến trình. Dạng thức khác của cải tiến tiến trình là để tăng hiệu xuất của các bước chắc chắn. Đây là một trong những sự đóng góp chính của môi trường: để tự động hoá các nhiệm vụ làm bằng tay mà không có hiệu lực hoặc các lỗi dấu mặt. Sự chuyển đổi thành một tiến trình phần mềm chín muồi đã đưa vào một thách thức và cơ hội mới đối với việc điều hành quản lý các hoạt động đồng thời (concurrent-hợp lý) và đối với những tiến bộ hữu hình và sự ước định chất lượng. Kinh nghiệm dự án làm rõ ra rằng một môi trường tích hợp cao là cần thiết cho cả việc làm cho thuận tiện và bắt tuân theo sự điều hành quản lý của tiến trình. Một môi trường mà cung cấp ngữ nghĩa của sự tích hợp (nơi mà môi trường hiểu được chi tiết biểu lộ sự phát triển của các tạo tác) và tiến trình tự động hoá có thể cải tiến năng xuất, cải tiến chất lượng phần mềm,và thúc đẩy sự chấp thuận các kỹ thuật hiện đại. Một môi trường mà hỗ trợ sự gia tăng các chương trình dịch, sự tự động

hóa xây dựng hệ thống, và tích hợp kiểm thử thụt lùi có thể đưa ra sự chuyển đổi hướng một cánh nhanh chóng cho sự phát triển lặp lại và cho phép các nhóm phát triển lặp lại một cách tự do hơn.

Một sự nhấn mạnh quan trọng của một cách tiếp cận hiện đại là để định nghĩa sự phát triển và môi trường bảo dưỡng như một lớp tạo tác đầu tiên của tiến trình. Một cách thiết thực, sự tích hợp môi trường phát triển phải ủng hộ kỹ thuật tự động hoá của tiến trình phát triển. Môi trường này sẽ bao hàm các yêu cầu quản lý, tài liệu kỹ thuật tự động hoá, các công cụ chương trình chủ/đích, tích hợp kiểm thử thụt lùi, sự kế tiếp và tích hợp thay đổi quản lý, và dấu vết đặc trưng/thiếu sót. Một sợi dây chung trong các dự án phần mềm thành công là họ đã thuê được những con người tốt và cung cấp cho họ những công cụ tốt để hoàn thành các công việc của họ. Sự tự động hoá của tiến trình thiết kế đưa ra kết quả trả lại trong chất lượng, khả năng để ước lượng các chi phí và lập lịch, và toàn bộ năng xuất sử dụng một nhóm nhỏ hơn. Các bộ công cụ tích hợp đóng một vai trò quan trọng ngày càng tăng trong sự phát triển tăng lên/lặp lại bằng cách cho phép các nhà thiết kế đi tắt qua một cách nhanh chóng giữa sự phát triển các tạo tác và việc giữ chúng hợp thời (được cập nhật).

Kỹ thuật khứ hồi là một kỹ thuật được sử dụng để mô tả khả năng chính của các môi trường mà hỗ trợ sự phát triển lặp lại. Như chúng ta có sự chuyển dịch sang những kho thông tin bảo trì khác nhau đối với các tạo tác kỹ thuật, chúng ta cần kỹ thuật tự động hoá hỗ trợ để bảo đảm hiệu quả và lỗi chuyển đổi tự do của dữ liệu từ tạo tác này sang tạo tác khác. Kỹ thuật ở phía trước là sự tự động hoá của một kỹ thuật tạo tác từ một cái khác, sự trình bày trừu tượng hơn. Ví dụ, các chương trình dịch và các chương trình liên kết đã được cung cấp kỹ thuật tự động hoá chuyển đổi từ chương trình nguồn thành chương trình có thể thực hiện được. Kỹ thuật dự trữ là sự sinh ra hoặc sự sửa đổi của một sự trình bày trừu tượng hơn từ một tạo tác đã tồn tại (ví dụ, việc tạo một mô hình thiết kế trực quan từ một sự trình bày chương trình nguồn).

Kỹ thuật khứ hồi miêu tả môi trường hỗ trợ cần thiết để thay đổi một tạo tác một cách tự do và những tạo tác khác tự động hoá thay đổi vì vậy tính nhất quán được duy trì trong các bộ toàn vẹn của các yêu cầu, thiết kế, thực hiện, triển khai các tạo tác. Khái niệm này đã được phát triển một cách đầy đủ hơn trong chương 12.

Các kiến trúc đã bắt đầu được sử dụng những thành phần, các nền, và các ngôn ngữ hỗn tạp (không đồng nhất), sự phức tạp của việc xây dựng, việc điều hành, và duy trì các mớ tỉ lệ lớn của các thành phần mới được xuất hiện cần thiết đối với điều khiển cấu hình và tự động hoá quản lý xây dựng. Tuy nhiên, các môi trường ngày nay không đóng đối với việc hỗ trợ tự động hoá đến phạm vi có thể. Ví dụ, tự động hoá kiểm tra trường hợp xây dựng từ việc sử dụng trường hợp và kịch bản mô tả vẫn chưa tiến triển cái gì đó ngoài trường hợp tầm thường nhất, như khối kiểm tra kịch bản.

Một lời cảnh báo là cần thiết trong việc mô tả sự cải tiến kinh tế kết hợp với các công cụ và các môi trường. Đó là cái chung đối với công cụ người bán để tạo nên mối liên quan các

đánh giá cá nhân đúng đắn trong vòng đời hoạt động để hỗ trợ các yêu sách về tác động kinh tế tiềm ẩn trong các công cụ của họ. Ví dụ, dễ dàng tìm thấy những trình bày như sau từ nhiều đoàn (công ty) trong một công cụ đặc biệt phù hợp:

- Phân tích các yêu cầu và các hoạt động phát triển tiêu tốn 40% chi phí của vòng đời.
- Các hoạt động thiết kế phần mềm có một tác động trong hơn 50% các tài nguyên.
- Các hoạt động lập trình và khối kiểm thử tiêu tốn khoảng 50% công sức và lịch trình phát triển phần mềm.
- Các hoạt động kiểm tra có thể tiêu tốn nhiều như 50% các tài nguyên của dự án.
- Điều khiển cấu hình và quản lý thay đổi là các hoạt động tới hạn mà có thể tiêu tốn nhiều bằng 25% các tài nguyên trong một dự án lớn.
- Các hoạt động tài liệu có thể tiêu tốn hơn 30% các tài nguyên kỹ thuật của dự án.
- Quản lý dự án, sự quản lý thương mại, và đánh giá tiến bộ có thể tiêu tốn nhiều bằng 30% ngân sách của dự án.

Xem xét một cách riêng biệt không có một khẳng định nào thực sự đúng; chúng quá đơn giản (Đưa ra khẳng định này, không quá ngạc nhiên khi nó chiếm tới 275% nguồn ngân sách và lịch trình để hoàn thành hầu hết các dự án!). Khi xem xét cùng với nhau các khẳng định có thể rất dối trá. Coi chừng kiểu kết luận này:

> *Công cụ kiểm thử này sẽ cải tiến năng xuất kiểm thử của bạn lên khoảng 20%. Bởi vì các hoạt động kiểm thử trên tốn 50% vòng đời, nó là một lợi ích năng xuất thực 10% cho toàn bộ dự án. Với 1.000.000$ ngân sách, bạn có đủ khả năng để tiêu 100.000$ trong các côngcụ kiểm thử.*

Các mối quan hệ qua lại của tất cả các hoạt động và công cụ phát triển phần mềm là khá phức tạp đối với những điều quyết đoán đơn giản như vậy là chấp nhận được. Theo kinh nghiệm của tôi, việc kết hợp tất cả các hiệu quả của tất cả các công cụ có xu hướng nhỏ hơn khoảng 40%, và hầu hết lợi ích này không được hiểu cặn kẽ là thiếu một vài thay đổi tương xứng trong tiến trình. Không có lẽ rằng bất kì công cụ riêng lẻ nào sẽ cải tiến năng xuất của một dự án lên nhiều hơn 5%. Nhìn chung bạn đã làm tốt hơn tiêu chuẩn hoá của hầu hết những người bán khẳng định thực tế đến 275% tổng số hơn 100% tổng số mà bạn phải xử lý trong thực tế.

3.5.Đạt được yêu cầu chất lượng

Bảng 3-5. Những sự cải tiến chất lượng chung với một tiến trình hiện đại.

Bộ điều khiển chất lượng	Tiến trình truyền thống	Tiến trình lặp lại hiện đại
Sự hiểu lầm các yêu cầu	Bị phát hiện muộn	Được giải quyết sớm
Sự phát triển rủi ro	Không biết đến khi đã muộn	Sự hiểu biết và giải quyết sớm
Các thành phần thương mại	Vòng đời muộn, sự hỗn loạn, và ác tính	Vòng đời sớm, sự thẳng thắn, và ôn hoà
Các lỗi thiết kế	Bị phát hiện muộn	Được giải quyết sớm
Sự tự động hoá	Hầu hết lỗi dấu mặt trong những thủ tục làm bằng tay	Hầu hết tự động hoá, lỗi tự do tiến triển trong các tạo tác
Tài nguyên tương xứng	Không có khả năng dự đoán	Có khả năng dự đoán
Các lịch trình	Bị cưỡng ép quá	Có thể hoà được chất lượng, hiệu năng, và công nghệ
Hiệu năng đích	Cơ sở phân tích hay sự mô phỏng tách biệt trên giấy	Các mẫu gốc có thể thực thi, phản hồi hiệu năng sớm, số lượng kiến thức
Tiến trình phần mềm nghiêm ngặt	Cơ sở tài liệu	Quản lý, đo đạc, và công cụ được hỗ trợ

Nhiều chấp nhận nào đó ngày nay như phần mềm thực hành tốt nhất được bắt nguồn từ tiến trình phát triển và tổng kết các công nghệ trong chương này. Những thực tế này có tác động trong việc thêm vào sự cải tiến hiệu xuất chi phí. Nhiều trong số chúng cũng cho phép cải tiến chất lượng đối với các chi phí tương tự. Bảng 3-5 tổng kết một vài hướng cải tiến chất lượng.

Những thực hành cơ bản để cải tiến toàn bộ chất lượng phần mềm gồm:

- Trọng tâm vào việc điều chỉnh các yêu cầu và giới hạn sử dụng các trường hợp sớm trong vòng đời, trọng tâm vào việc hoàn thành các yêu cầu và tính theo dõi muộn trong vòng đời, và trọng tâm xuyên suốt vòng đời trong một sự cân bằng giữa sự phát triển các yêu cầu, phát triển thiết kế, và sự phát triển các kế hoạch.
- Sử dụng các độ đo và các chỉ báo để đo tiến trình và chất lượng của một kiến trúc như nó được giải quyết từ một mẫu gốc mức cao sang một sản phẩm hoàn thiện đầy đủ.
- Việc cung cấp các môi trường tích hợp vòng đời mà hỗ trợ sớm và tiếp tục điều hành cấu hình, quản lý thay đổi, các phương pháp thiết kế nghiêm ngặt, tài liệu tự động hoá, và tự động hoá kiểm thử thụt lùi.
- Sử dụng các mẫu trực quan và các ngôn ngữ bậc cao hơn mà hỗ trợ điều khiển kiến trúc, sự trừu tượng, chương trình đáng tin cậy, việc tái sử dụng, và tài liệu tự giải nghĩa (tài liệu hướng dẫn).
- Khả năng thấu hiểu bên trong sớm và tiếp tục đến hiệu năng phát ra qua việc trình bày các định mức cơ sở.

Cải tiến khả năng thấu hiểu bên trong sớm đến chạy thử nghiệm hiệu xuất phát ra thậm chí còn quan trọng hơn các dự án kết hợp hỗn hợp các thành phần thương mại và các thành phần phát triển truyền thống. Tiến trình phát triển truyền thống đã nhấn mạnh ước lượng kích thước và thời gian sớm của việc lợi dụng tài nguyên chương trình máy tính. Tuy nhiên các bộ môn nghiên cứu theo niên đại tiêu biểu trong việc đánh giá hiệu năng như sau:

- Sự khởi đầu dự án. Thiết kế được đề nghị đã được xác nhận là rủi ro thấp với đầy đủ hiệu năng tới hạn.
- Xem lại thiết kế ban đầu. Những đánh giá lạc quan của giới hạn thiết kế xứng đáng là cơ sở cho hầu hết các phân tích hoặc sự mô phỏng phác thảo trên giấy về các đường giới hạn. Trong hầu hết các trường hợp các thuật giải ứng dụng hoạt động và kích thước cơ sở dữ liệu đã được hiểu một cách khá tốt. Tuy nhiên, kết cấu hạ tầng bao gồm tổng phí hệ điều hành, tổng phí hệ quản trị cơ sở dữ liệu, và tổng phí tiến trình qua lại và mạng thông tin và toàn bộ những chuỗi thứ cấp đã bị hiểu lầm một cách điển hình.
- Xem lại thiết kế giữa vòng đời. Những sự đánh giá đã bắt đầu gọt giũa dần đi ở đường biên (mép lề) như những tiêu chuẩn sớm và những kiểm thử ban đầu đã bắt đầu bộc lộ niềm lạc quan cố hữu trong những ước lượng sớm.
- Sự tích hợp và kiểm thử. Những vấn đề hiệu năng nghiêm trọng đã được tiết lộ ra, cần phải thay đổi cơ bản trong kiến trúc. Cấu trúc hạ tầng thấp thường là người đứng mũi chịu sào, nhưng thủ phạm thực sự là việc sử dụng cơ sở hạ tầng chưa chín muồi, các

giải pháp kiến trúc chưa chín muồi, hoặc sự hiểu biết nghèo nàn đã sớm phải trả giá cho thiết kế.

Trình tự này đã xuất hiện bởi vì khả năng thấu hiểu hiệu năng bên trong sớm đã làm cơ sở duy nhất trong việc phán đoán kỹ thuật non nớt của vô số những tiêu chí. Trong hầu hết những hệ thống phân tán lớn đã hợp thành nhiều thành phần tác động qua lại với nhau, một sự biểu lộ cách tiếp cận cơ sở có thể cung cấp sự đánh giá đúng đắn có ý nghĩa hơn về hiệu năng sinh ra. Những sự biểu lộ sớm này phải được đặt trên nền chủ hoặc nền đích hoặc từng phần của cấu hình mạng. Trong bất cứ trường hợp nào, chúng ta có thể đặt kế hoạch và quản lý để đưa ra một kỹ thuật thực hành thành công. Sinh ra hiệu năng sớm là một điển hình. Chúng ta thậm chí vẫn mạnh khoẻ, bởi vì chúng có xu hướng phơi bày hết những vết nứt kiến trúc hoặc những yếu kém của các thành phần thương mại sớm trong vòng đời khi những quyền trả giá có thể được tạo nên.

3.6.Chú ý vào việc kiểm tra: một quan điểm thực dụng

Chú ý vào việc kiểm tra là một sự vận động thái quá như khía cạnh chính của một hệ thống chất lượng. Theo kinh nghiệm của tôi, chú ý xem lại là rất có giá trị như một cơ cấu thứ cấp, nhưng chúng là những đóng góp quan trọng hiếm hoi để chất lương được so sánh với những cơ cấu và chỉ báo chất lượng cơ bản sau, những cái được nhấn mạnh trong tiến trình quản lý:

- Việc chuyển thông tin kỹ thuật từ một bộ tạo tác sang một bộ khác, bởi vậy việc đánh giá tính nhất quán, tính khả thi, tính có thể nhận thức được, và công nghệ ràng buộc cố hữu trong các kỹ thuật tạo tác.
- Sự biểu lộ những mốc chính mà buộc các tạo tác được đánh giá chống lại tiêu chuẩn hữu hình trong hoàn cảnh của những trường hợp sử dụng xác đáng.
- Các công cụ môi trường (các bộ dịch, những người gỡ rối, những người phân tích, tự động hoá kiểm thử hoàn toàn) mà đảm bảo sự trình bày nghiêm ngặt, nhất quán, sự hoàn thiện, và điều khiển thay đổi.
- Vòng đời kiểm thử đối với chi tiết bên trong đến giới hạn trả giá, tiêu chuẩn chấp thuận và sự phục tùng các yêu cầu.
- Việc thay đổi quản lý các độ đo đối với mục tiêu bên trong đến những xu hướng thay đổi có triển vọng và sự hội tụ hoặc phân tuyến từ chất lượng và mục tiêu tiến bộ.

Mặc dù tôi tin tưởng rằng việc kiểm tra là đã quá nhấn mạnh, trong các trường hợp chắc chắn chúng cho ra một kết quả có ý nghĩa. Một trong những giá trị của việc kiểm tra là trong sự phát triển chuyên môn của đội làm dự án. Nó là sự hữu ích chung để có những sản phẩm của những thành viên ít tuổi hơn của đội được xem lại bởi những thành viên nhiều tuổi hơn có khinh nghiệm. Đưa những sản phẩm của những tay nghiệp dư sang tay của những chuyên gia

và ngược lại là một cơ cấu tốt để thúc đẩy toàn bộ đội ngũ lao động mới đạt được kiến thức và kỹ năng tay nghề. Những sai lầm thô ngớ ngẩn (blunders) có thể bị bắt và việc phản hồi có thể giành được kênh truyền, vì vậy những thực hành tồi sẽ không sống được lâu dài (perpetuated). Đây là một trong những cách tốt nhất để những kỹ sư phần mềm ít tuổi hơn học tập.

Việc kiểm tra cũng là một phương tiện tốt đối với việc bắt các tác giả phải chịu trách nhiệm về chất lượng các sản phẩm. Tất cả các tác giả và tài liệu hướng dẫn phần mềm đều chú ý xem xét kỹ lưỡng các sản phẩm của họ như một tiến trình tự nhiên của sản phẩm. Do đó, việc gộp vào kiểm tra sẽ được thông qua tất cả các tác giả hơn là thông qua các thành phần. Những tác giả ít tuổi hơn cần có một thành phần ngẫu nhiên đã được kiểm tra một cách định kì, và họ có thể học tập qua việc kiểm tra các sản phẩm của những tác giả có thâm niên cao hơn. Những mức khác nhau của việc kiểm tra không chính thức đã được thực hiện tiếp tục khi những nhà phát triển đọc hoặc tích hợp phần mềm với những phần mềm của tác giả khác, và trong suốt quá trình kiểm thử của những nhóm kiểm thử độc lập. Tuy nhiên, việc "kiểm tra" này là hữu hình nhiều hơn việc trọng tâm vào các khía cạnh tích hợp và khả năng thực thi của toàn bộ hệ thống.

Cuối cùng, một thành phần giới hạn xứng đáng được kiểm tra bởi một vài người, tốt nhất là những người mà họ có nguyên tắc trong chất lượng, hiệu năng, hay bộ đặc trưng của nó. Một sự kiểm tra trọng tâm vào việc giải quyết một cái tồn tại phát sinh có thể là một cách hiệu quả để xác định nguyên nhân hay đi đến một giải pháp mà nguyên nhân có thể hiểu được.

Mặc dù có nhiều lợi ích của việc kiển tra, nhưng nhiều tổ chức lại quá nhấn mạnh vào các cuộc họp và những cuộc kiểm tra chính thức và đòi hỏi phạm vi bao hàm qua toàn bộ kỹ thuật của các sản phẩm. Cách tiếp cận có thể cực kỳ phản tác dụng (có tác dụng ngược lại). Chỉ 20% các tạo tác kỹ thuật (như là việc sử dụng các trường hợp, các mô hình thiết kế, mã hoá nguồn và các trường hợp kiểm thử) xứng đáng được khảo sát tường tận chi tiết khi đem so sánh với cái khác, bảo đảm các hoạt động hữu ích hơn. Một tiến trình mà nhấn mạnh vào sự bảo đảm vào chất lượng chính trong quá trình kiểm tra thì sẽ không có được lợi nhuận. Một vài nghiên cứu đã xuất bản nhấn mạnh vào tầm quan trọng và kết quả đầu tư (ROI) cao của việc kiểm tra. Tôi nghi ngờ rằng nhiều trong những nghiên cứu này đã được viết bởi những người bảo đảm chất lượng chuyên nghiệp người mà đã phóng đại sự cần thiết đối với luật lệ của họ. Tôi thường xuyên một mình nói về chủ đề này, nhưng đây là cơ sở hợp lý căn bản của tôi.

Những lỗi thiết kế quan trọng hoặc có giá trị hoặc các kiến trúc phát ra hiếm khi rõ ràng từ một sự xem xét thiển cận trừ khi sự kiểm tra trong phạm vi hẹp chú trọng vào một sự phát ra đặc biệt. Và hầu hết những sự kiểm tra đều thiển cận. Những hệ thống ngày nay đã phức tạp khá cao, với vô số thành phần, sự hài hoà trong thực thi, việc phân chia các tài nguyên, và nhiều hướng đòi hỏi cân bằng phức tạp khác. Nó đã đem đến cho con người khả năng lý luận giống như những lớp người chơi cờ vua thế giới để thấu hiểu được động lực tác động trong một số hệ thống phần mềm đơn giản dưới một vài trường hợp sử dụng đơn giản. Bởi vậy, sự kiểm tra ngẫu nhiên của con người có xu hướng thoái hoá thành những bình luận trong phong

cách và sự phát ra trật tự ngữ nghĩa ban đầu. Chúng hiếm khi cho kết quả trong việc khám phá những chỗ làm đình trệ hiệu năng thực sự, những sự phát sinh điều khiển nghiêm trọng (như những sự đình trệ hoàn toàn, những cuộc đua tài hay tranh giành tài nguyên), hoặc kiến trúc yếu ớt (như là những vết nứt trong tính vô hướng, tín tin cậy, hay tính thao tác qua lại). Nhưng trong tất cả các trường hợp tầm thường, những phát sinh kiến trúc chỉ phơi bày ra qua những hoạt động kỹ thuật nghiêm ngặt hơn như sau:

- Việc phân tích, tạo mẫu gốc, hoặc sự thử nghiệm.
- Việc xây dựng các mô hình thiết kế.
- Việc phó thác các trạng thái hiện tại của các mô hình thiết kế cho một chương trình có thể thực hiện được có hiệu lực.
- Việc chứng minh các mặt mạnh và các mặt yếu của việc thực thi hiện tại trong ngữ cảnh của các tập hợp con tới hạn trong việc sử dụng các trường hợp và các kịch bản.
- Tập hợp những bài đã học lại thành những mô hình, sử dụng các trường hợp, sự thực thi, và những kế hoạch.

Việc đạt được chất lượng kiến trúc là điều cố hữu (sẵn có) trong một tiến trình lặp lại mà phát ra các bộ tạo tác cùng nhau trong sự cân bằng. Các điểm kiểm tra dọc theo quá trình là khá nhiều, bao gồm việc xem lại của con người và những sự kiểm tra chú trọng vào những sự phát sinh tới hạn. Nhưng những sự kiểm tra này không phải là những điểm kiểm tra chính. Các vòng đời tạo tác sớm chắc chắn phụ thuộc nhiều hơn vào mục tiêu xem lại của con người so với các vòng đời tạo tác muộn hơn. Việc chú trọng vào tỉ lệ phần trăm lớn của các tài nguyên của một dự án trong việc kiểm tra của con người là các thực tập tồi và chỉ làm sống mãi sự tồn tại của các giá trị thấp được thêm vào bởi những ngươi kiểm tra những người mà có một tác động nhỏ vào sự thành công của dự án. Hãy nhìn vào bất cứ công sức phần mềm thành công nào và hỏi những nhà thiết kế chính, những người kiểm tra, hoặc những nhà phát triển về sự khác biệt của sự thành công của họ. Không có lẽ rằng bất kì điều gì trong chúng cũng sẽ được trích dẫn trong các cuộc họp, trong kiểm tra hay tài liệu.

Bảo đảm chất lượng là trách nhiệm chung của mọi người và sẽ thuộc về toàn bộ hầu hết tất cả các tiến trình hoạt động thay vì một luật lệ tách biệt được thực hiện bởi những chuyên gia bảo đảm chất lượng. Việc đánh giá và ước định chất lượng của sự tiến hoá kỹ thuật đường cơ sở sẽ là công việc của một đội kỹ thuật đó là sự độc lập của kiến trúc và sự phát triển của đội. Sự đánh giá vòng đời của họ về sự tiến triển các tạo tác sẽ là tiêu biểu bao gồm sự quản lý thay đổi, sự phân tích các xu hướng, và việc kiểm thử, tốt như sự kiểm tra.

Chương 4
Cách cũ và cách mới

Những điểm chính:

- Kỹ thuật phần mềm truyền thống đã thiết lập rất nhiều nguyên tắc. Nhiều nguyên tắc vẫn còn giá trị; những nguyên tắc khác đã lỗi thời.
- Một tiến trình quản lý phần mềm hiện đại sẽ tập hợp nhiều nguyên tắc truyền thống nhưng cũng sẽ chuyển sang một vài cách tiếp cận mới vững chắc.

Hơn hai thập kỷ qua đã có một kỹ thuật lại (re-engineering) có ý nghĩa của tiến trình phát triển phần mềm. Nhiều sự quản lý và kỹ thuật thực hành truyền thống đã được sửa đổi lại theo những cách tiếp cận mới mà kết hợp những chủ đề lặp lại những kinh nghiệm dự án thành công với sự nâng cao của kỹ thuật công nghệ phần mềm. Sự chuyển đổi này đã được thúc đẩy bởi không thể đáp ứng được đòi hỏi đối với những đặc trưng phần mềm đã được sản xuất một cách nhanh chóng hơn dưới áp lực cạnh tranh nhiều hơn để giảm chi phí. Trong nền công nghiệp phần mềm thương mại, sự kết hợp của các áp lực cạnh tranh, tính lợi nhuân, tính đa dạng của nhiều khách hàng, sự thay đổi công nghệ một cách nhanh chóng là nguyên nhân khiến nhiều tổ chức bắt đầu khai sinh một mẫu (paradigm) quản lý mới để đáp ứng những áp lực ngân sách, động lực và môi trường đe doạ thay đổi khác nhau, thời gian sống vận hành lâu dài của các hệ thống, và tỉ lệ dễ ảnh hưởng lớn, các ứng dụng phức tạp.

4.1 .Các nguyên tắc của kỹ thuật phần mềm truyền thống

Có nhiều sự mô tả về kỹ thuật phần mềm "theo cách cũ." Sau những năm kinh nghiệm phát triển phần mềm, nền công nghệp phần mềm đã học được nhiều bài học và được công thức hoá thành nhiều nguyên tắc. Phần này miêu tả một quan điểm của các nguyên tắc kỹ thuật phần mềm ngày nay như một tiêu chuẩn đối với việc giới thiệu những chủ đề chính được bàn đến xuyên suốt trong phần còn lại của cuốn sách. Tiêu chuẩn mà tôi đã chọn lựa là một mục ngắn gọn có tiêu đề: "Mười lăm nguyên tắc của kỹ thuật phần mềm" [Davis, 1994]. Mục đã được tiếp tục phát triển thành thành một cuốn sách [Davis, 1994] đã liệt kê 201 nguyên tắc. Mặc dù tiêu đề, mục của nó miêu tả 30 nguyên tắc đỉnh cao và nó tốt như bất kỳ sự khôn ngoan truyền thống nào trong nền công nghiệp phần mềm. Trong khi tôi tán thành với nhiều trong số những sự khôn ngoan này, tôi tin rằng một số trong chúng đã lỗi thời. 30 nguyên tắc đỉnh cao của Davis được trích dẫn tiếp sau đây, bằng chữ in nghiêng. Đối với mỗi nguyên tắc tôi đều bình luận về những triển vọng được đưa ra muộn hơn trong cuốn sách này sẽ được tán thành hoặc thay đổi nó. Tôi đã một vài sự khẳng định ở đây mà đã rời đi không có căn cứ đến tận các chương sau.

1. ***Tạo chất lượng số 1.*** *Chất lượng phải được xác định định lượng và các cơ cấu phải sắp xếp vào những vị trí nhằm thúc đẩy nó đạt được.*

➢ Việc định nghĩa chất lượng ứng với dự án có thể với tới được là quan trọng nhưng nó không dễ dàng được thực hiện ở ngay thời điểm ban đầu của một dự án. Kết quả là một tiến trình khung công việc hiện đại cố gắng để biết được các trả giá giữa các đặc trưng, chất lượng, chi phí, và lịch trình sớm trong vòng đời như có thể. Đến tận khi tận khi những hiểu biết này đạt được, thì nó lại không thể định rõ hoặc quản lý chất lượng đạt được.

2. ***Phần mềm chất lượng cao là có thể.*** *Các kỹ thuật mà đã được giải thích nhằm làm tăng chất lượng bao gồm việc thu hút khách hàng, tạo mẫu gốc, việc đơn giản hoá thiết kế, việc hướng dẫn kiểm tra, và việc thuê nhân công tốt.*

➢ Nguyên tắc này hầu như không cần thiết với những quá trình khác.

3. ***Đưa các sản phẩm cho khách hàng.*** *Không có gì khó khăn để anh cố gắng học nhứng cái người dùng cần trong suốt pha yêu cầu, cách hiệu quả nhất để xác định những điều thực sự cần là đưa cho người dùng và cho phép họ sử dụng sản phẩm.*

➢ Đây là nguyên lý chính của tiến trình khung công việc hiện đại, và đó có thể là một vài cơ cấu để thu hút khách hàng trong suốt vòng đời. Việc phụ thuộc vào các miền, các cơ cấu này có thể bao gồm việc có thể chứng minh các mẫu gốc, cơ sở của sự chứng minh các mốc, và sự giải phóng alpha/beta.

4. ***Xác định vấn đề trước khi viết các yêu cầu.*** *Khi đối mặt với một cái gì đó họ tin rằng đó là một vấn đề, hầu hết các kỹ sư đều vội vàng đưa ra một giải pháp. Trước khi anh cố gắng giải quyết một vấn đề, hãy chắc chắn rằng đã thăm dò (khảo sát) tất cả các khả năng, và đừng bị mờ mắt bởi giải pháp rõ ràng.*

➢ Nguyên tắc này là sự chỉ rõ của nhứng phát sinh có liên quan với tiến trình yêu cầu đặc điểm kỹ thuật truyền thống. Các tham số của vấn đề trở nên rõ ràng hơn như một giải pháp phát triển. Một tiến trình khung công việc hiện đại phát triển vấn đề và giải phát cùng với nhau cho đến khi vấn đề ddã được hiểu đủ cặn kẽ để giao cho việc sản xuất đầy đủ.

5. ***Đánh giá các khả năng thiết kế khác nhau.*** *Sau các yêu cầu đã được đồng ý như trên, anh phải kiểm tra các kiến trúc và giải thuật khác nhau. Anh chắc chắn không muốn sử dụng một "kiến trúc" đơn giản bởi vì nó đã được sử dụng trong các yêu cầu đặc điểm kỹ thuật.*

➢ Nguyên tắc này dường như được neo chặt trong tinh thần thác nước theo 2 cách: (1) Các yêu cầu ưu tiên kiến trúc hơn là giải quyết cùng nhau. (2) Kiến trúc đã kết hợp trong các yêu cầu đặc điểm kỹ thuật. Trong khi một tiến trình hiện đại rõ ràng thúc đẩy việc phân tích các khả năng thiết kế khác nhau, các hoạt động này đã thực hiện đồng thời với yêu cầu các đặc điểm kỹ thuật, và các ký hiệu và các tạo tác đối với các yêu cầu và kiến trúc đã được phá vỡ cặp (decoupled) một cách rứt khoát.

6. ***Sử dụng một mô hình tiến trình thích hợp.*** *Mỗi dự án phải chọn một tiến trình mà tạo nên hầu hết các cảm nhận đối với dự án đó trên cơ sở của văn hoá đoàn thể, sự sẵn sàng mang các rủi ro, vùng ứng dụng, sự không ổn định của các yêu cầu, và phạm vi mà các yêu cầu được hiểu cặn kẽ.*

➢ Đúng là không một tiến trình riêng lẻ nào là phổ quát. Tôi sử dụng giai đoạn tiến trình khung công việc để trình bày một lớp các tiến trình mềm dẻo hơn một thí dụ cứng nhắc đơn giản. Chương 14 bàn luận về cấu hình và đồ may của tiến trình đối với những điều cần thiết khác nhau của dự án.

7. ***Sử dụng những ngôn ngữ khác nhau đối với những pha khác nhau.*** *Sự khao khát không ngừng của nền công nghiệp của chúng ta đối với những giải pháp đơn giản cho những vấn đề phức tạp đã được điều chỉnh nhiều nhằm đưa ra phương pháp phát triển tốt nhất là một phương pháp mà sử dụng những kí hiệu tương tự trong suốt vòng đời. Tại sao các kỹ sư phần mềm lại sử dụng Ada cho các yêu cầu, thiết kế, và mã hoá phải chăng Ada đã là một giải pháp tối ưu đối với tất cả những pha này?*

➢ Đây là một nguyên tắc quan trọng. Chương 6 miêu tả một tổ chức thích hợp và giới thiệu các ngôn ngữ/ các kí hiệu cho các tạo tác nguyên thuỷ của tiến trình.

8. ***Tối thiểu hoá khoảng cách trí tuệ.*** *Để tối thiểu hoá khoảng cách trí tuệ, cấu trúc của phần mềm phải gần như có thể đối với cấu trúc thế giới thực.*

➢ Nguyên tắc này là động lực thúc đẩy chính đối với sự phát triển của các kỹ thuật hướng đối tượng, sự phát triển các thành phần cơ bản, và mẫu trực quan.

9. ***Đặt các kỹ thuật trước các công cụ.*** *Một kỹ sư phần mềm vô kỷ luật với một công cụ sẽ trở thành một kỹ sư phần mềm vô kỷ luật nguy hiểm.*

➢ Mặc dù nguyên tắc này có giá trị, nhưng thiếu vắng hai điểm quan trọng: (1) Một kỹ sư phần mềm có kỷ luật với những công cụ tốt sẽ sản sinh ra những chuyên gia phần mềm có kỷ luật nhưng không có công cụ. (2) Một cách tốt nhất để khuyến khích, tiêu chuẩn hoá, và phân phối các kỹ thuật tốt là chuyển qua kỹ thuật tự động hoá.

10. ***Làm cho nó đúng trước khi anh làm nó nhanh hơn.*** *Khá dễ dàng để tạo một chương trình làm việc chạy nhanh hơn là việc tạo một chương trình làm việc nhanh. Đừng lo lắng về sự tối ưu trong suốt quá trình lập trình ban đầu.*

➢ Đây là một một sự trình bày thấu hiểu cặn kẽ bên trong. Nó đã được phát biểu sai bởi một vài chuyên gia phần mềm nhiều hoặc ít hơn như sau: "Các vấn đề hiệu năng sớm trong một hệ thống phần mềm là một dấu hiệu chắc chắn của rủi ro xuôi dòng." Mọi thành công, dự án phần mềm không tầm thường mà tôi biết đã có hiệu năng phát ra xuất hiện sớm trong vòng đời. Tôi sẽ chỉ rõ rằng hầu như tất cả các

kiến trúc chưa chín muồi (đặc biệt là những kiến trúc có tỉ lệ lớn) có hiệu năng phát ra trong sự lặp lại của các chương trình có thể thực thi được đầu tiên của chúng. Có một vài việc thực thi (làm việc) sớm là điều kiện tiên quyết để hiểu được các trả giá hiệu năng phức tạp. Đó cũng là điều rất khó để có được khả năng thấu hiểu bên trong qua việc phân tích.

*11. **Kiểm tra kỹ việc mã hoá.** Việc kiểm tra thiết kế chi tiết và việc mã hoá (lập trình) là một cách tốt hơn nhiều để tìm ra lỗi so với việc kiểm thử.*

➢ Giá trị của nguyên tắc này là sự vận động thái quá nhưng đối với toàn bộ các hệ thống phần mềm đơn giản nhất. Các tài nguyên phần cứng, các ngôn ngữ lập trình, và các môi trường tự động hoá ngày nay cho phép từ động hoá phân tích và kiểm thử được thực hiện một cách có hiệu quả trong suốt vòng đời. Việc tiếp tục và tự động hoá vòng đời kiểm thử là một sự cần thiết trong bất kỳ một sự phát triển lặp lại hiện đại nào. Nhìn chung, sự kiểm tra gián tiếp (tương phản với những sự kiểm tra trọng tâm vào những phát sinh đã biết) hiếm khi lộ ra sự phát sinh kiến trúc hoặc các trả giá thiết kế tổng thể. Điều này không nói rằng tất cả những sự kiểm tra là cực kỳ hiệu quả trong việc giải quyết các vâns đề. Nhưng nguyên tắc này không ở trong 15 nguyên tắc đỉnh cao, đặc biệt xét cho cùng thì các thực hành ngầm định của nền công nghiệp là để xem xét một cách quá kỹ lưỡng.

*12. **Sự quản lý tốt quan trọng hơn công nghệ tốt.** Công nghệ tốt nhất sẽ không bù lấp cho một cho sự quản lý nghèo nàn, và một nhà quản lý tốt có thể sản xuất ra một kết quả lớn thậm chí với nguồn tài nguyên đạm bạc. Một sự quản lý tốt sẽ thúc đẩy con người làm việc với khả năng tốt nhất của họ, nhưng đó không phải là một điều "đúng" phổ biến của sự quản lý.*

➢ Sự tin tưởng của tôi vào nguyên tắc này là nguyên nhân khiến tôi viết cuốn sách này. Tôi chỉ muốn tranh luận ở đây đó là thuật ngữ *nguồn tài nguyên nghèo nàn*, nó còn mập mờ. Một điều rất lớn, một đội được quản lý tốt có thể làm được những điều lớn với một nguồn ngân sách và lịch trình nghèo nàn. Một sự quản lý tốt và một đội nghèo nàn về chất lượng, ở một mặt khác, là sự qua lại riêng biệt, bởi vì một nhà quản lý tốt sẽ thu hút, sắp xếp thành hình thể, và giữ lại đội chất lượng.

*13. **Con người là chìa khoá để thành công.** Con người có kỹ năng tay nghề cao với sự chiếm lĩnh kinh nghiệm, tài năng, và việc đào tạo sẽ là chìa khoá. Những con người giỏi giang dù thiếu các công cụ, ngôn ngữ, tiến trình sẽ thành công. Những con người kém cỏi dù có đầy đủ các công cụ, ngôn ngữ, và tiến trình vẫn sẽ có thể thất bại.*

➢ Nguyên tắc này quá thấp trong danh sách.

*14. **Đi theo sau sự thận trọng.** Bởi vì mọi người đang làm gì đó mà không tạo nên những cái đúng cho anh. Nó có thể đúng, nhưng anh phải cẩn thận đánh giá tính thích hợp của nó đối với môi trường của anh. Một sự hướng đối tượng, sự đo đạc, việc tái sử dụng, sự cải tiến tiến trình, CASE, việc tạo mẫu gốc - tất cả những điều này phải làm tăng chất lượng, giảm chi phí, và làm tăng sự thoả mãn của khách hàng. Tiềm năng của các kỹ thuật như vậy thường là có giá bán rất cao (oversold), con lợi nhuận không có nghĩa là đã được bảo đảm hay phổ biến.*

➢ Đây là một lời khuyên nhủ khôn ngoan, đặc biệt trong một nền công nghiệp đang trưởng thành nhanh chóng, nơi mà công nghệ có khái niệm kỳ cục rất khó để phân biệt từ những sự cải tiến công nghệ. Việc trả giá cho các đặc trưng, các chi phí, và các lịch trình luôn không ưng thuận các công nghệ hiện đại nhất.

***15. Có tinh thần trách nhiệm.** Khi một cây cầu sụp đổ chúng ta hỏi: "Những kỹ sư nào đã làm hỏng cây cầu?" Thậm chí khi phần mềm sụp đổ, chúng ta hiếm khi hỏi câu này. Thực tế là trong bất kỳ kỷ luật kỹ thuật nào, các phương pháp tốt nhất có thể được sử dụng để sản sinh ra các thiết kế dở, và các phương pháp cổ xưa để sản sinh ra những thiết kế quí giá.*

➢ Đây là một hệ quả tất yếu lớn của nguyên tắc 14. Nó có những phương pháp, những công cụ, và những thành phần tốt hơn để thành công. Nó cũng có những con người giỏi, sự quản lý tốt, và một sự học tập có văn hoá mà trọng tâm hướng vào sự cải tiến thậm chí khi phải đối đầu với số đông và không tránh khỏi thất bại ngay lập tức.

*16. **Hiểu biết những ưu tiên của khách hàng.** Có thể khách hàng sẽ chịu được 90% chức năng giao nộp muộn nếu họ có thể có 10% đã đúng giờ.*

➢ Việc hiểu biết ác ưu tiên của khách hàng là quan trọng, nhưng chỉ trong sự cân bằng với những ưu tiên của các cổ đông khác. "Khách hàng luôn luôn đúng" là một tinh thần có thể thu được kết quả là việc tiêu hoang phí tiền hơn bất kỳ nhận thức sai lệch nào khác. Đặc biệt trong sự điều chỉnh lĩnh vực ký hợp đồng, nhưng thông thường hơn bất kì khi nào một khách hàng kí kết hợp đồng với một người hợp nhất hệ thống, khách hàng thường xuyên thất bại.

***17. Nhiều hơn những gì họ thấy, nhiều hơn những gì họ cần.** Nhiều chức năng (hoặc hiệu năng) hơn anh cung cấp cho một người dùng, nhiều chức năng (hoặc hiệu năng) hơn những gì người dùng muốn.*

➢ Nguyên tắc này rất đúng, nhưng nó đề nghị rằng anh sẽ chưa bao giờ muốn trình bày cho khách hàng bất cứ thứ gì. Nó sẽ đọc: "Nhiều hơn những gì người dùng cần, tốt hơn những gì người dùng biết." Không phải toàn bộ các cổ đông đều 100% bị điều chỉnh bởi sự tham lam. Họ biết rằng họ có giới hạn tài nguyên và những nhà

phát triển cũng có sự tự chủ. Biểu hiện ngay lập tức kết quả là một sự tích cực cao có thể thấy rõ, nó cần thiết cho việc đồng bộ hoá những sự trông đợi của cổ đông. Chi nhánh (hệ quả) của nguyên tắc này trong một tiến trình hiện đại là người quản lý dự án phần mềm cần có mục tiêu dữ liệu để mà tranh luận những những yêu cầu thay đổi không tránh khỏi và bảo dưỡng một sự cân bằng của tính có đủ khả năng, các đặc trưng, và rủi ro.

18. Có kế hoạch để ném một cái đi. *Một trong những nhân tố thành công tới hạn quan trọng là có hay không có một sản phẩm hoàn toàn mới. Như những ứng dụng, kiến trúc, giao diện, hoặc thuật giải mới tinh hiếm khi làm việc được ngay lần đầu tiên.*

➢ Anh không có kế hoạch để ném một cái đi. Đúng hơn, anh có kế hoạch để tạo ra một sản phẩm từ một mẫu gốc chưa chín muồi đến một đường cơ sở chín muồi. Nếu anh có để ném nó đi, đồng ý, nhưng đừng lập kế hoạch cho nó từ một sự bắt đầu. Đây có thể là lời khuyên khôn ngoan đối với 100% tập quán, việc dẫn dắt các dự án phát triển phần mềm mũi nhọn trong quá khứ. Tuy nhiên, trong các hệ thống phần mềm ngày nay, nhiều trong số các thành phần đã có (ít nhất là hệ điều hành, hệ quản trị cơ sở dữ liệu, giao diện đồ hoạ người dùng, mạng, và thiết bị trung gian), và nhiều trong số những gì đã xây dựng trong thời kì đầu tiên đã qua có thể được bảo thủ.

19. Thiết kế đối với sự thay đổi. *Các kiến trúc, các thành phần, và sự định rõ các kỹ thuật mà anh sử dụng phải điều tiết sự thay đổi (thích ứng với sự thay đổi).*

➢ Đây là một sự thực hiện rất đơn giản mà đã được chứng minh là cực kỳ phức tạp để nhận biết. Một cách cơ bản, có thể nói rằng chúng ta phải dự đoán tương lai và xây dựng một khung công việc mà có thể điều tiết sự thay đổi mà nó vẫn chưa được định nghĩa tốt. Tuy nhiên tôi tán thành nguyên tắc này một cách toàn tâm toàn ý bởi vì nó là tới hạn để thành công. Rất khó để dự đoán tương lai một cách chính xác, nhưng việc cố gắng để dự đoán được các loại thay đổi mà có lẽ để xuất hiện trong một vòng đời của hệ thống mà một thực hành hữu ích trong quản lý rủi ro và một chủ đề tuần hoàn của các dự án phần mềm thành công.

20. Thiết kế mà không có tài liệu là không thiết kế. *Tôi thường nghe các kỹ sư phần mềm nói, "Tôi đã hoàn thành thiết kế. Tất cả chúng đã rời khỏi tài liệu."*

➢ Nguyên tắc này cũng được néo chặt trong tài liệu-cách tiếp cận điều chỉnh trong quá khứ, nơi mà tài liệu đã phân tách từ chính phần mềm. Với mẫu trực quan và các ngôn ngữ có thứ tự bậc cao hơn, nó thường phản tác dụng để bảo dưỡng tài liệu chia tách đối với mục đích mô tả thiết kế phần mềm. Các tài liệu kiến trúc mức cao có thể cực kỳ hữu ích nếu chúng được viết một cách sinh động và súc tích, nhưng các tạo tác chính được sử dụng bởi đội ngũ kỹ thuật là việc thiết kế các ký hiệu, mã

nguồn, và kiểm thử các đường cơ sở. Tôi sẽ sửa chữa nguyên tắc này như sau, để khai thác công nghệ nâng cao ngày nay tốt hơn: "Các tạo tác phần mềm sẽ hầu như tự đưa ra tài liệu." Nguyên tắc này được miêu tả rất dài trong chương 6.

21. Sử dụng các công cụ, nhưng phải có óc thực tế. *Các công cụ phần mềm làm cho những người dùng của chúng nhiều hiệu quả hơn.*

➢ Nguyên tắc này làm tầm thường hoá một khía cạnh quyết định của kỹ thuật phần mềm hiện đại: tầm quan trọng của môi trường phát triển. Một tiến trình chín muồi phải là sự thiết lập, tự động hoá, và dụng cụ hoá tốt. Các dự án phát triển lặp lại yêu cầu sự tự động háo rộng rãi. Nó không khôn ngoan để đầu tư vào môi trường chủ chốt.

22. Tránh các thủ đoạn gian trá. *Nhiều lập trình viên yêu thích việc tạo những chương trình với những việc xây dựng các thủ đoạn gian trá mà việc thực hiện một chức năng một cách đúng đắn, nhưng theo một cách mù mờ. Hãy xem thế giới đau đớn ra sao bởi vậy bạn nên tránh chương trình quỷ quyệt.*

➢ Tôi tìm thấy nó rất khó khăn để tin rằng đây là một trong 30 nguyên tắc đỉnh cao. Rất khó để vẽ đường thẳng giữa một "thủ đoạn gian trá" và một giải pháp có tính chất sáng tạo. Tôi biết chính xác Davis đang với tới điều gì, nhưng tôi không muốn ban hành một nguyên tắc mà có bất kỳ ý nghĩa nào về sự cách tân ngột ngạt. Việc làm khó hiểu các kỹ thuật mã hóa sẽ được tránh trừ khi đó là những lý do hấp dẫn để sử dụng chúng. Không may, những lý do hấp dẫn như vậy là chung trong các dự án không tầm thường.

23. Hãy tóm lược lại. *Việc ẩn thông tin là một điều dơn giản, các khái niệm đã chứng minh rằng các kết quả trong phần mềm là dễ dàng hơn để kiểm thử và dễ dàng hơn nhiều để bảo quản.*

➢ Thiết kế thành phần cơ bản, thiết kế hướng đối tượng, và thiết kế hiện đại và các ký hiệu lập trình đã nâng cao nguyên tắc này thành dòng thực hành chính. Sự tóm lược như là nền tảng kỹ thuật đối với một kỹ sư phần mềm như toán học đối với một nhà vật lý. Nó sẽ là chủ đề duy nhất của một học kỳ trong các trường đại học mà dạy kỹ thuật phần mềm.

24. Sử dụng vật kết nối và sự cố kết. *Vật kết nối và sự cố kết là những cách tốt nhất để đo tính bảo trì và tính tương hợp cố hữu của phần mềm.*

➢ Nguyên tắc sống còn này rất khó được ứng dụng. Vật kết nối và sự cố kết là những sự miêu tả trừu tượng của các thành phần cho điều mà tôi biết là nó không được thiết lập tốt, sự định nghĩa mục tiêu. Vật kết nối và sự cố kết là khó khăn bởi vậy để đo đạc. Các độ đo hiện đại đối với đề địa chỉ tính có thể bảo trì và tính tương hợp là trung tâm trong việc đo đạc tổng số phế thải và việc làm lại phần mềm. Dính

liền các thành phần với vật kết nối nhỏ li ti là dễ dàng hơn việc kết nối với phế thải và việc làm lại ít hơn. Chúng ta có thể lý do về căn bệnh (quá nhiều vật kết nối và quá ít sự cố kết) chỉ bởi khả năng quan sát và đo đạc các triệu chứng (phế thải và việc làm lại).

25. Sử dụng cách đo phức hợp McCabe. *Mặc dù có nhiều độ đo sẵn có để báo cáo sự phức tạp cố hữu của phần mềm, không như là trực giác và dễ dàng sử dụng như của Tom McCabe.*

➢ Những độ đo phức tạp rất quan trọng đối với việc định danh một vài thành phần tới hạn mà cần sự chăm sóc đặc biệt. Tuy nhiên, theo kinh nghiệm của tôi, điều nhãi nhép phức tạp thực sự là rõ ràng, và nó hiếm khi thấy được những đo đạc phức tạp này được sử dụng trong lĩnh vực ứng dụng để quản lý một dự án hoặc ra quyết định. Những độ đo này khá thú vị từ một lý thuyết suông có triển vọng (siêu dự án nghiên cứu và chiến lược ra quyết định) và có thể hữu ích trong quản lý dự án (nếu đã tự động hoá), nhưng chúng không thuộc về những nguyên tắc đỉnh cao.

26. Đừng kiểm thử phần mềm của chính bạn. *Những nhà phát triển phần mềm không bao giờ nên là những người kiểm thử chính của phần mềm của chính họ.*

➢ Nguyên tắc này thường được thảo luận. Một mặt, một đội kiểm thử độc lập đề xuất một mục tiêu có triển vọng. Trên một mặt khác, những nhà phát triển phần mềm cần tạo ra chủ nhân của chất lượng trong những sản phẩm của họ. Trong chương 11, tôi tán thành với cả 2 triển vọng: Những nhà phát triển sẽ kiểm thử phần mềm của chính họ, và vì vậy sẽ tách ra một đội.

27. Phân tích các nguyên nhân của các lỗi. *Đó là kết quả có giá trị hơn để giảm hiệu lực của một lỗi chống lại nó so với việc tìm ra và sửa chữa lại nó. Một cách để làm điều này là phân tích nguyên nhân của các lỗi khi chúng đã được tìm ra.*

➢ Trên bề mặt, đây là một nguyên tắc tốt, đặc biệt trong việc xây dựng pha. Khi các lỗi có lẽ bị lặp lại. Nhưng việc phân tích lỗi trong các hệ thống phần mềm phức tạp đã tìm ra một trong những nguồn tài nguyên tới hạn là quá nhiều phân tích và quá nhiều thiết kế trên giấy trong giai đoạn sớm của một dự án. Đến một vài cấp, các hoạt động này là nỗ lực "sự chống chọi lại lỗi ". Chúng cho kết quả trong một kết quả đầu tư thấp hơn so với việc đã được hiểu thấu từ sự uỷ thác đối với việc tạo mẫu gốc và việc xây dựng các hoạt động, điều sẽ làm cho các lỗi rõ ràng và hữu hình hơn. Bởi vậy tôi sẽ trình bày lại những điểm chính của hai nguyên tắc này: (1) Đừng sợ tạo ra các lỗi trong giai đoạn kỹ thuật. (2) Phân tích nguyên nhân đối với lỗi trong giai đoạn sản xuất.

28. ***Nhận ra rằng phép đo năng lượng (entrôpy) của phần mềm tăng lên.*** *Bất kỳ hệ thống phần mềm nào mà phải chịu đựng tiếp tục sự thay đổi sẽ trưởng thành trong sự phức tạp và sẽ trở nên ngày càng phá hoại tổ chức.*

- Đây là một sự bỏ sót lại khác của kiến trúc phần mềm truyền thống. Hầu như tất cả các hệ thống phần mềm đều tiếp tục chịu đựng sự thay đổi, và dấu hiệu của một kiến trúc nghèo nàn là phép đo năng lượng của nó tăng lên theo một cách mà rất khó quản lý. Phép đo năng lượng có xu hướng tăng lên một cách nguy hiểm khi giao diện bị thay đổi cho những lý do chiến thuật. Sự nhất quán của một kiến trúc là tính chiến lược và cố hữu cơ bản trong giao diện của nó, và nó phải được điều chỉnh với sự khảo sát kỹ lưỡng có cường độ cao. Các công cụ quản lý thay đổi buộc một dự án tôn trọng và tuân theo giao diện nhất quán. Một kiến trúc chất lượng là một kiến trúc mà phép đo năng lượng tăng lên một cách tối thiểu và sự thay đổi có thể được điều tiết ổn định khả năng có thể dự đoán được kết quả. Một kiến trúc lý tưởng sẽ cho phép có sự thay đổi mà không có bất kỳ sự tăng lên nào của phép đo năng lượng.

29. ***Con người và thời gian không thể thay đổi cho nhau.*** *Việc đo đạc một dự án duy nhất bởi cá nhân hàng tháng trời tạo nên một ít giác quan.*

- Nguyên tắc này không bị ảnh hưởng bởi thời gian (bất tận).

30. ***Hãy mong đợi sự xuất sắc.*** *Những người lao động của anh sẽ làm tốt hơn nhiều nếu anh có một sự trông đợi cao đối với họ.*

- Nguyên tắc này áp dụng cho tất cả các luật lệ, không áp dụng đối với quản lý phần mềm.

Tôi đã sử dụng một vài từ gây tranh luận trong những bình luận của tôi. Mục đích của tôi là không tán thành cũng không phản bác đặc biệt là các nguyên tắc của Davis, nhưng tôi nhưng sẽ hay hơn nếu tôi bộc lộ ra những thành kiến và những suy nghĩ kích động của tôi. Trong khi tôi thấy giá trị khác thường (rất tốt) trong khoảng một nửa các nguyên tắc, một nửa kia cũng cần một sự thay đổi ưu tiên hoặc chúng đã bị lỗi thời bởi những công nghệ mới.

4.2.Các nguyên tắc quản lý phần mềm hiện đại

Mặc dù các nguyên tắc quản lý phần mềm hiện tại miêu tả trong phần 4.1 được phát triển và được cải tiến từ các kỹ thuật truyền thống, chúng vẫn không nhấn mạnh các nguyên tắc hiện đại trong cuốn sách này làm cơ sở. Được xây dựng trên định dạng của Davis đây là 10 nguyên tắc đỉnh cao của tôi về quản lý phần mềm hiện đại. (Năm nguyên tắc đầu, là những chủ đề chính trong sự định nghĩa của tôi về một tiến trình lặp, đã được tổng kết trong hình 4-1.) Các nguyên tắc được sắp xếp theo thứ tự ưu tiên, và những từ ***mặt đậm-in nghiêng (bold-faced italicized)*** đã được sử dụng trong suốt cuốn sách như phép tốc ký (shorthand) đối với những định nghĩa đã được mở rộng này.

1. Cơ sở của tiến trình trong một ***cách tiếp cận kiến trúc đầu tiên***. Những yêu cầu này mà một sự cân bằng có thể giải thích được đã giành được giữa việc điều chỉnh các yêu cầu, những quyết định thiết kế có ý nghĩa về mặt kiến trúc, và vòng đời các kế hoạch trước khi nguồn tài nguyên được giao phó cho nấc thang phát triển đầy đủ.

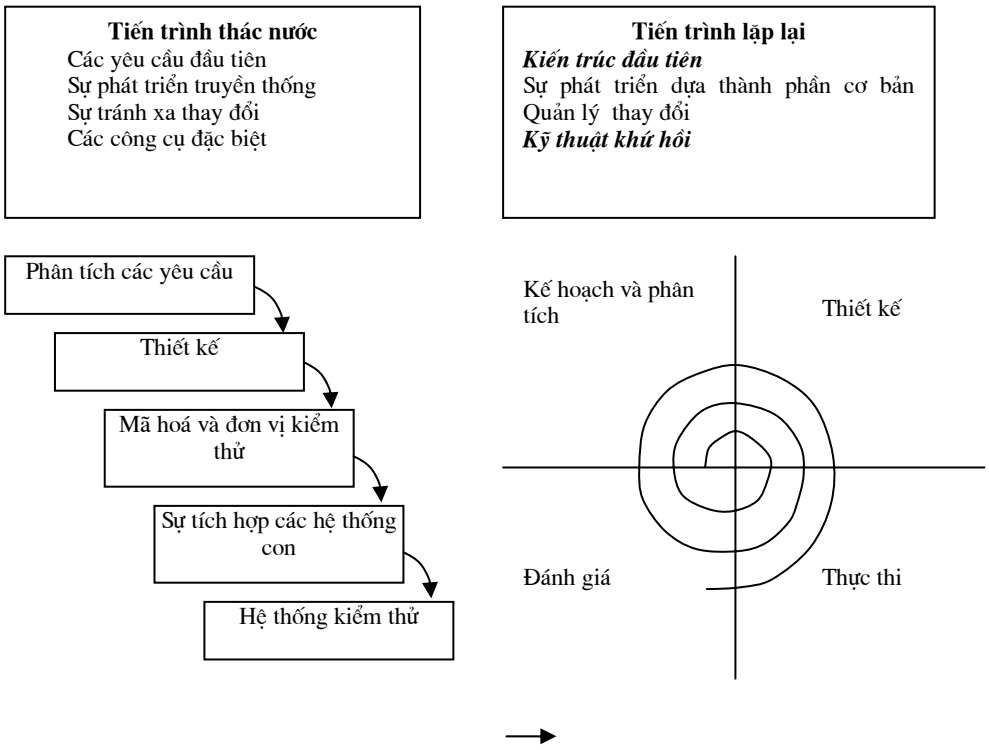
2. Thiết lập một tiến trình vòng đời lặp lại mà đối mặt với những rủi ro sớm. Với những hệ thống phần mềm tinh vi ngày nay, nó không thể định nghĩa vấn đề toàn vẹn, thiết kế giải pháp toàn vẹn, xây dựng phần mềm, sau đó kiểm thử và kết thúc sản phẩm trong sự liên tục. Thay vì, một tiến trình lặp mà tinh lọc vấn đề hiểu biết, một giải pháp hiệu quả, và một kế hoạch hiệu quả qua một vài sự lặp lại đã khuyến khích một cách xử lý (cư xử) cân bằng khách quan của tất cả các cổ đông. Những rủi ro chính phải được đề địa chỉ sớm để tăng khả năng dự đoán và tránh phải trả giá đắt cho phế thải và làm lại xuôi dòng.

3. Truyền các phương pháp thiết kế để nhấn mạnh sự phát triển dựa vào các thành phần cơ bản. Việc chuyển từ một dòng mã mang tính tinh thần sang một thành phần cơ bản mang tính tinh thần là sự cần thiết để giảm tổng số mã nguồn được sinh ra bởi con người và sự phát triển truyền thống. Một thành phần (phần tử) là một bộ cố kết của những dòng mã đã tồn tại trước đó, cũng trong nguồn hoặc định dạng có thể thực hiện được, với một giao diện và cách xử trí đã được định nghĩa.

4. Thiết lập một một môi trường quản lý thay đổi. Động lực của sự phát triển lặp lại, bao gồm các luồng công việc đồng quy trong công việc của các nhóm khác nhau trên các tạo tác đã được chia sẻ, cần phải có mục tiêu điều chỉnh các đường cơ sở.

5. Đề cao sự thay đổi tự do qua các công cụ mà hỗ trợ kỹ thuật khứ hồi. Kỹ thuật khứ hồi là môi trường hỗ trợ cần thiết để tự động hoá và đồng bộ hoá thông tin kỹ thuật trong những định dạng khác nhau (như là các đặc điểm kỹ thuật yêu cầu, các mô hình thiết kế, mã nguồn, mã có thể thực hiện được, các trường hợp kiểm thử). Thiếu sự tự động hoá quan trọng của kế toán, sự quản lý thay đổi, sự cung cấp tài liệu, và việc kiểm thử này, thì sẽ rất khó để giảm các chu kỳ lặp lại để có thể quản lý cấu trúc thời gian nơi mà sự thay đổi được khuyến khích hơn là được né tránh. Sự thay đổi tự do là một sự cần thiết trong tiến trình lặp lại, và việc thiết lập một môi trường đã được tích hợp là điều cốt yếu.

6. Các tạo tác thiết kế giành được một cách nghiêm ngặt, mô hình kí hiệu cơ bản. Một mô hình tiếp cận cơ bản (như UML) hỗ trợ sự phát triển của đồ hoạ phong phú ngữ nghĩa và các kí hiệu thiết kế nguyên bản. Mẫu trực quan với các kí hiệu nghiêm ngặt và một máy móc có thể xử lý ngôn ngữ hình thức cung cấp cho phép đo khách quan hơn so với cách tiếp cận truyền thống trong sự cân nhắc của con người và sự kiểm tra thiết kế đặc biệt trình bày trong các tài liệu giấy.



Cách tiếp cận kiến trúc đầu tiên → **Thành phần thiết kế trung**

Thiết kế và tích hợp đầu tiên, sau đó sản xuất và kiểm thử

Tiến trình vòng đời lặp → **Thành phần quản lý rủi ro**

Điều khiển rủi ro qua việc tăng chức năng, hiệu năng, chất lượng từ

Phát triển thành phần → **Thành phần công nghệ**

Các phương pháp hướng đối tượng, các ký hiệu ngiêm ngặt, mẫu trực

Môi trường quản lý thay đổi → **Thành phần điều khiển**

Các độ đo, các xu hướng, sự trang bị dụng cụ tiến trình

Kỹ thuật khứ hồi → **Thành phần tự động hoá**

Bổ sung hoàn thiện các công cụ, tích hợp các môi trường

Hình 4-1. *Năm nguyên tắc đỉnh cao của một tiến trình hiện đại.*

7. Dụng cụ tiến trình cho ***điều khiển chất lượng mục tiêu*** và sự đánh giá tiến triển. Đánh giá vòng đời về sự tiến triển và chất lượng của tất cả các sản phẩm ngay lập tức phải được tích hợp thành tiến trình. Các cơ cấu đánh giá tốt nhất là những cách đo được định nghĩa tốt bắt nguồn trực tiếp từ việc phát triển các tạo tác kỹ thuật và được tích hợp thành tất cả các hoạt động và các nhóm làm dự án.

8. Sử dụng một cách tiếp cận đã có cơ sở chứng minh để đánh giá các tạo tác ngay lập tức. Việc chuyển các sản phẩm tạo tác trong giai đoạn phát triển hiện tại (các tạo tác là một mẫu gốc sớm, một kiến trúc đường cơ sở, hay một khả năng bêta) sang một sự chứng minh có thể thực hiện được của những kịch bản thích ứng kích thích sự hội tụ sớm hơn của việc tích hợp, một hiểu biết hữu hình hơn về các trả giá thiết kế và việc loại bỏ sớm hơn các kiến trúc không hoàn hảo.

9. Có kế hoạch giải phóng ngay lập tức các nhóm sử dụng kịch bản với việc phát triển các mức chi tiết. Đó là điều thiết yếu mà tiến trình quản lý phần mềm điều chỉnh hướng tới sớm và tiếp tục sự chứng minh trong ngữ cảnh vận hành của hệ thống, tên trong các trường hợp sử dụng của nó. Sự phát triển từng bước của dự án sự tăng lên và sự sinh ra phải ứng với mức độ hiểu biết hiện tại của các yêu câù và kiến trúc. Cố kết cách sử dụng kịch bản khi đó là cấu trúc chính đối với việc tổ chức các yêu cầu, việc định nghĩa nội dung sự lặp lại, việc đánh giá sự thực thi, và việc tổ chức sự thừa nhận kết quả kiểm thử.

10. Thiết lập một tiến trình có thể sắp xếp thành cấu hình (configurable) đó có thể là kinh tế vô hướng. Không tiến trình đơn giản nào có thể phù hợp đối với tất cả sự phát triển phần mềm. Một khung tiến trình thực dụng phải có khả năng sắp xếp thành hình thể đối với toàn bộ phạm vi khái quát các ứng dụng. Tiến trình phải đảm bảo có tỉ lệ kinh tế và kết quả trong đầu tư bằng việc khai thác một khuynh hướng (spirit) tiến trình chung, sự tự động hoá tiến trình bao quát, và các mẫu vẽ kiến trúc và các thành phần chung.

Mười nguyên tắc đỉnh cao của tôi không có cơ sở khoa học. Tuy nhiên, chúng thực hiện, nắm bắt một quan điểm cân bằng về các chủ đề lặp lại được trình bày xuyên suốt cuốn sách này. Bảng 4-1 vẽ lên bản đồ những gì mà tôi coi là 10 rủi ro đỉnh cao của tiến trình truyền thống đối với thuộc tính khoá và các nguyên tắc của một tiến trình hiện đại. Mặc dù bảng còn chứa các nguyên tắc thô (chưa được tinh lọc) chung chung, nhưng ở một mức cao nó đưa ra một sự giới thiệu (tiến cử) các nguyên tắc của một tiến trình hiện đại.

Bảng 4-1. *Những cách tiếp cận tiến trình hiện đại để giải quyết vấn đề truyền thống.*

Tiến trình truyền thống: 10 rủi ro đỉnh cao	**Tác động**	**Tiến trình hiện đại: các đặc trưng giải quyết rủi ro cố hữu**
1. Đoạn vỡ muộn và phế thải/ làm lại quá mức.	Chất lượng, chi phí, lịch trình	Kiến trúc cách tiếp cận đầu tiên Sự phát triển lặp lại Tự động hoá quản lý thay đổi Tiến trình đối mặt rủi ro

2. Sự tiêu hao đội ngũ lao động chính	Chất lượng, chi phí, lịch trình	Sự lặp lại thành công, sớm Sự quản lý và kế hoạch đáng tin cậy
3. Nguồn tài nguyên phát triển không tương xứng	Chi phí, lịch trình	Các môi trường như là lớp tạo tác đầu tiên của tiến trình Môi trường công nghiệp mạnh, đã được tích hợp Mô hình tạo tác kỹ thuật cơ sở Kỹ thuật khứ hồi
4. Các cổ đông đối thủ	Chi phí, lịch trình	Xem lại cơ sở sự chứng minh Sử dụng các yêu cầu/kiểm thử trường hợp hướng đối tượng
5. Sự bổ xung công nghệ cần thiết	Chi phí, lịch trình	Kiến trúc cách tiếp cận đầu tiên Phát triển các thành phần cơ sở
6. Các yêu cầu khiếp đảm (creep)	Chi phí, lịch trình	Sự phát triển lặp lại Xem lại cơ sở sự chứng minh
7. Phân tích tình trạng tê liệt (Analysis paralysis)	Lịch trình	Xem lại cơ sở sự chứng minh Sử dụng các yêu cầu/kiểm thử trường hợp hướng đối tượng
8. Hiệu năng không tương xứng	Chất lượng	Đánh giá hiệu năng cơ sở sự chứng minh Sự phản hồi hiệu năng kiến trúc sớm
9. Quá nhấn mạnh vào các tạo tác	Lịch trình	Đánh giá cơ sở chứng minh Điều khiển chất lượng mục tiêu
10. Chức năng không tương xứng	Chất lượng	Sự phát triển lặp lại Tạo mẫu gốc sớm, sự giải phóng tăng lên

4.3.Chuyển sang một tiến trình lặp

Các tiến trình phát triển phần mềm hiện đại đã được chuyển đến từ mô hình thác nước truyền thống, nơi mà trong mỗi giai đoạn của tiến trình phát triển phụ thuộc vào hoàn toàn vào giai đoạn trước đó. Mặc dù có những sự biến đổi, những cách tiếp cận hiện đại nhìn chung đòi hỏi một phiên bản ban đầu của hệ thống phải được nhanh chóng xây dựng sớm trong tiến trình phát triển, với việc nhấn mạnh vào việc đề địa chỉ các vùng rủi ro cao, việc ổn định kiến trúc cơ sở, và việc tinh chế các yêu cầu điều khiển (với người dùng rộng rãi vào những nơi có thể). Sự phát triển lúc đó như một chuỗi lặp lại, xây dựng trên kiến trúc hạt nhân cho đến khi các mức mong muốn về chức năng, hiệu năng, và sự tráng kiện đạt được. (Những sự lặp lại này đã được gọi là sự xoắn ốc, sự gia tăng, sự sinh ra, hay giải phóng.) Một tiến trình lặp lại nhấn mạnh toàn bộ hệ thống hơn là từng phần riêng lẻ. Rủi ro đã giảm xuống sớm trong vòng đời qua việc tiếp tục tích hợp và sự tinh chế các yêu cầu, kiến trúc và các kế hoạch. Những sự ngạc nhiên xuôi dòng mà quấy rầy các dự án phần mềm truyền thống đã được tránh.

Những lợi nhuận kinh tế cố hữu trong sự chuyển đổi từ mô hình thác nước truyền thống sang một tiến trình phát triển lặp lại là có ý nghĩa nhưng khó xác định chất lượng. Như một tiêu chuẩn tác động kinh tế mong đợi của sự cải tiến tiến trình, coi như tiến trình diễn giải các thông số của mô hình COCOMO II. (Phụ lục B trình bày chi tiết hơn về mô hình COCOMO.) Sự diễn giải này có thể sắp thứ tự từ 1.01 (hầu như không tỉ lệ phi kinh tế) đến 1.26 (có ý nghĩa tỉ lệ phi kinh tế). Các tham số mà điều hành giá trị của tiến trình diễn giải là có tiền lệ ứng dụng, tiến trình mềm dẻo, sự phát triển rủi ro kiến trúc, đội cố hữu, và tiến trình phần mềm trưởng thành.

Các đoạn văn sau đây vẽ bản đồ tiến trình diễn giải các tham số của COCOMO II với 10 nguyên tắc đỉnh cao của tôi về một tiến trình hiện đại.

- **Có tiền lệ ứng dụng.** Miền kinh nghiệm là một nhân tố tới hạn trong việc hiểu biết như thế nào đó để có kế hoạch và thực thi một dự án phát triển phần mềm. Đối với những hệ thống chưa có tiền lệ, một trong những mục tiêu chính là đối mặt với các rủi ro thiết lập các tiền lệ sớm, thậm chí nếu chúng chưa hoàn thiện hoặc dựa trên thí nghiệm. Đây là một trong những lý do chính mà nền công nghiệp phần mềm đã chuyển sang một ***tiến trình vòng đời lặp lại.*** Những sự lặp lại sớm trong vòng đời thiết lập các tiền lệ từ sản phẩm, tiến trình, và các kế hoạch có thể được thí nghiệm trên ***các mức tiến triển chi tiết.***

- **Tiến trình mềm dẻo.** Sự phát triển của phần mềm hiện đại đã được biểu thị đặc điểm bằng một không gian giải pháp rộng lớn như vậy và rất nhiều quan hệ có liên quan mà nó là một sự tối cần đối với việc tiếp tục hợp nhất của các thay đổi. Những thay đổi này có thể cố hữu trong việc hiểu biết vấn đề, không gian giải pháp, hoặc các kế hoạch. Các tạo tác dự án phải được hỗ trợ bởi ***sự quản lý thay đổi*** có hiệu quả xứng với những điều dự án cần. Cả tiến trình cứng nhắc và tiến trình thay đổi hỗn độn đều để dành cho sự thất bại ngoại trừ với hầu hết các dự án tầm thường. Một ***tiến trình có khả năng sắp xếp thành hình thể*** cho phép một khung công việc chung được kết hợp qua một sự sắp xếp của các dự án là cần thiết để đạt được một kết quả đầu tư phần mềm.

- **Tiến triển từng bước rủi ro kiến trúc.** Sự phát triển kiến trúc đầu tiên là một chủ đề quyết định nằm dưới một tiến trình phát triển lặp lại thành công. Một đội làm dự án phát triển và ổn định một kiến trúc trước khi phát triển tất cả các thành phần mà tạo nên sự phù hợp toàn vẹn cảu các thành phần ứng dụng. Một ***kiến trúc đầu tiên*** và ***cách tiếp cận phát triển thành phần cơ sở*** buộc cơ sở hạ tầng, các cơ cấu chung, và các cơ cấu điều khiển được thử nghiệm sớm trong vòng đời và điều khiển tất cả các thành phần ra/mua quyết định thành tiến trình kiến trúc. Cách tiếp cận này khởi đầu hoạt động tích hợp sớm trong vòng đời như hoạt động thẩm tra của tiến trình thiết kế và các sản phẩm. Nó cũng buộc môi trường phát triển dành cho vòng

đời kỹ thuật phần mềm được xếp đặt cấu hình và thực hành sớm trong vòng đời, bởi vậy đảm bảo sự chú ý sớm đối với hoạt động kiểm thử và một nền tảng cho sự đánh giá ***cơ sở chứng minh.***

- **Đội cố kết.** Các đội thành công là cố kết, và các đội cố kết là thành công. Tôi không chắc chắn nó vừa là nguyên nhân vừa là kết quả, nhưng các đội thành công và các đội cố kết chia sẻ các mục tiêu và ưu tiên chung. Các đội cố kết tránh các nguồn của dự án hỗn loạn và phép đo năng lượng mà có thể cho kết quả từ những khó khăn trong việc đồng bộ những sự mong đợi của các cổ đông của dự án. Trong khi có rất nhiều lý do cho đối với sự hỗn loạn như vậy, một trong những lý do chính là không truyền thông, đặc biệt trong việc thay đổi thông tin một cách duy nhất qua các tài liệu giấy mà thông tin kỹ thuật trình bày một cách khách quan. Việc nâng cao công nghệ (như các ngôn ngữ lập trình, UML, và mẫu trực quan) có thể được cho phép nghiêm ngặt hơn và có khả năng hiểu được các ký hiệu cho việc truyền thông thông tin kỹ thuật phần mềm, đặc biệt trong việc yêu cầu và thiết kế các tạo tác mà trước đó là đặc biệt và được cơ sở một cách hoàn thiện trên những thay đổi trên giấy. Các định dạng ***mô hình cơ sở*** này cũng cho phép ***kỹ thuật khứ hồi*** hỗ trợ cần thiết cho việc thiết lập sự thay đổi tự do thích đáng cho việc trình bày thiết kế.
- **Tiến trình phần mềm trưởng thành.** Mẫu trưởng thành khả năng (CMM-Capability Maturity Model) của Viện kỹ thuật phần mềm (The Sofware Engineering Intitute) là một tiêu chuẩn chấp nhận tốt cho việc đánh giá tiến trình phần mềm. Miền kinh nghiệm như vừa rồi là điều quyết định cho việc tránh các rủi ro ứng dụng và việc khai thác miền tài sản sẵn có và những bài đã học, tiến trình phần mềm trưởng thành là điều cốt yếu đối với việc tránh các rủi ro phát triển phần mềm và việc khai thác các tài sản phần mềm của tổ chức và những bài đã học. (Những lý luận tán thành và phản đối của CMM đã được miêu tả rất dài trong phụ lục E) Một trong những chủ đề chính là những tiến trình trưởng thành đích thực được cho phép qua một môi trường tích hợp mà cung cấp mức độ thích hợp của sự tự động hoá cho dụng cụ tiến trình đối với ***điều khiển chất lượng mục tiêu.***

Chương 5
Các giai đoạn của vòng đời

Các điểm trọng tâm :

Giai đoạn công nghệ (The engineeering stage) trong vòng đời sẽ phát triển kế hoạch, các yêu cầu cùng với kiến trúc giải quyết các rủi ro phát triển (development risk). Giai đoạn này kết thúc với vạch ranh giới kiến trúc có thể thực hiện được (executable architecture basline).

Giai đoạn sản xuất (The production stage) của vòng đời sẽ xây dựng các phiên bản thích hợp cho việc sử dụng dựa trên vạch ranh giới và kế hoạch, các yêu cầu và kiến trúc đã đạt được trong giai đoạn công nghệ.

Đặc điểm của tính biệt nhất của một quy trình phát triển phần mềm thành công đó là sự tách biệt rõ ràng giữa các hoạt động nghiên cứu và phát triển (research and development activities) với các hoạt động sản xuất (production activities). Khi một dự án phần mềm không thành công, lý do đầu tiên thường là sự thiếu khả năng trong việc xác định rõ ràng và thực hiện 2 giai đoạn trên với sự cân bằng và sự nhấn mạnh phù hợp. Điều này không những đúng cho các quy trình truyền thống mà cũng đúng cho các quy trình lặp. Hầu hết các dự án phần mềm không thành công đều có một trong những đặc điểm sau:

- *Sự quá nhấn mạnh vào quá trình nghiên cứu và phát triển. Có quá nhiều phân tích và nghiên cứu trên giấy tờ, hoặc là sự chậm trễ trong việc xây dựng vạch danh giới công nghệ. Sự nhấn mạnh này thường xẩy ra trong các qui trình phần mềm truyền thống.*

- Sự nhấn mạnh quá mức vào quá trình sản xuất, điển hình là sự thiết kế vội vã (rush-to-judgment designs), vội vàng viết mã.

Những dự án thành công hiện đại, thậm chí cả dự án thành công phát triển dưới các quy trình truyền thống đều có khuynh hướng có một mốc dự án quan trọng (well-defined project mislestone) khi có sự chuyển tiếp đáng chú ý từ giai đoạn công nghệ sang giai đoạn sản xuất. Giai đoạn nghiên cứu đặt trong tâm vào việc đạt được các chức năng. Trong khi đó, giai đoạn

sản xuất xoay quanh việc đạt được một sản phẩm mà có thể chuyển giao cho khách hàng. Trong giai đoạn này, hiệu năng (performance), sự hài hoà (fit), sự hoàn thiện (finish), và khả năng (robustness) của sản phẩm được đặc biệt chú ý đến. Sự cân bằng vòng đời này đôi khi là điều gì đó tinh vi và không rõ ràng, nhưng vẫn là một trong những nền tảng cơ sở cho việc quản lý thành công một dự án phần mềm.

Một quy trình phát triển phần mềm hiện đại phải được định rõ để hỗ trọ các công việc sau:

- Sự phát triển kế hoạch, yêu cầu, kiến trúc cùng với các vấn đề về đồng bộ hoá.
- Quản lý rủi ro, và các phương pháp khách quan để ước lượng tiến trình và chất lượng.
- Sự phát triển năng lực của hệ thống thông qua sự chỉ dẫn của việc gia tăng chức năng.

5.1.Giai đoạn công nghệ và giai đoạn sản xuất

(engineering and production stages)

Các cơ sở kinh tế được giới thiệu ở chương trước đã cung cấp một khung công việc đơn giản để có được mô tả của vòng đời (life-cycle description). Để giảm bớt chi phí sản xuất (economies of scale) và đạt lợi nhuận cao hơn từ việc đầu tư, chúng ta phải thay đổi quy trình sản xuất công nghệ phần mềm

(sofware manufacturing process) bằng cách cải tiến công nghệ trong tự động hoá quy trình (process automation) và phát triển các thành phần cơ sở (component-based development).

Hai giai đoạn đầu tiên của vòng đời:

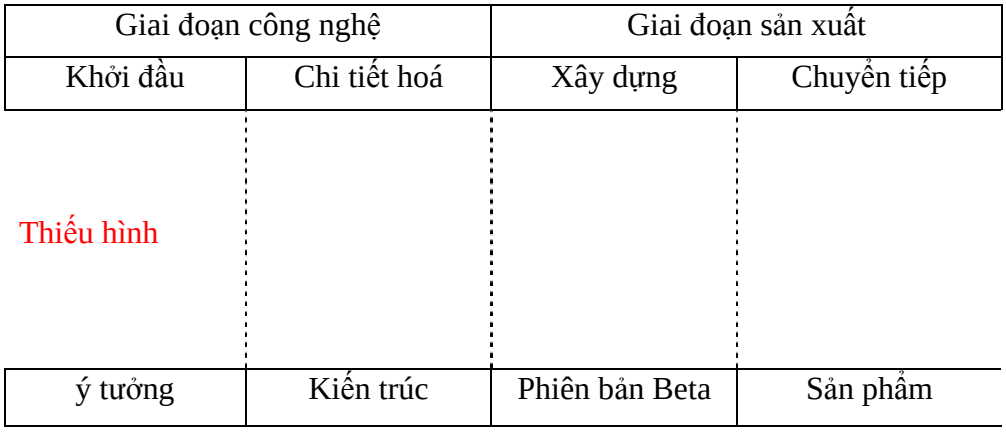
1. *Giai đoạn công nghệ, phát triển bởi các nhóm nhỏ, ít có khả năng dự đoán trước (smaller and less predictable team) làm các công việc thiết kế và tổng hợp (design and synthesis activities).*
2. *Giai đoạn sản xuất, phát triển bởi các nhóm lớn hơn có nhiều khả năng dự đoán trước (larger and more predictable team) làm các công việc xây dựng, kiểm tra và các hoạt động phát triển (construction, test, and deployment activities).*

Bảng 5.1. Tổng kết các sự nhấn mạnh khác nhau giữa 2 giai đoạn

Các mặt của vòng đời	*Nhấn mạnh trong giai đoạn công nghệ*	*Nhấn mạnh trong giai đoạn sản xuất*
Giảm bớt rủi ro	*Kế hoạch làm việc, khả thi công nghệ*	*Chi phí*
Sản phẩm	*Vạch ranh giới kiến trúc*	*Vạch ranh giới phát hành sản phẩm*
Hoạt động	*Phân tích, thiết kế, lập kế hoạch*	*Thực hiện, thử nghiệm*
Đánh giá	*Dẫn chứng kiểm tra, phân tích*	*Thử nghiệm*
Kinh tế	*Giải quyết sự tăng chi phí sản xuất*	*Khai thác việc giảm bớt chi phí sản xuất*
Quản lý	*Lập kế hoạch*	*Thực hiện*

Sự chuyển tiếp giữa giai đoạn công nghệ và giai đoạn sản xuất là một sự kiện quan trọng đối với các cổ đông (stakeholders). Kế hoạch sản xuất đã được chấp thuận và có sự hiểu biết đầy đủ về vấn đề, cách giải quyết mà các cổ đông có thế đưa ra một cam kết để đi đến giai đoạn sản xuất. Phụ thuộc vào các đặc điểm riêng biệt của từng dự án mà thời gian và tài nguyên (time and resources) giành cho 2 giai đoạn này là rất khác nhau. Tuy nhiên, đối với hầu hết các dự án, nếu coi vòng đời chỉ bao gồm 2 giai đoạn công nghệ và sản xuất thì có thế là một sự đơn giản thái qúa. Do đó, giai đoạn công nghệ lại được chia ra làm 2 giai đoạn phân biệt: Giai đoạn khởi đầu (inception phase), giai đoạn cụ thể hoá (elaboration phase) và giai đoạn sản xuất cũng được phân ra làm 2 giai đoạn: Giai đoạn xây dựng (construction phase), giai đoạn chuyển tiếp (transition phase).

Bốn giai đoạn này của vòng đời được vẽ một cách không chính xác lắm dưới mô hình xoắn ốc (spiral model) [Boehm, 1988] như trên hình vẽ 5.1



Hình 5-1, Các giai đoạn của vòng đời

Trong hình này, kích thước của các hình xoắn ốc tương ứng với mức độ ổn định (inertia) về chiều rộng và chiều sâu (breadth and depth) của mẫu (artifact) đang được phát triển. Mức độ cố định này biểu hiện trong việc duy trì tính nhất quán (consistency) của mẫu, kiểm tra ngược (regression testing), sự cung cấp tài liệu (documentation), phân tích (quality analyses) chất lượng và điều khiển cấu hình (configuration control). Tuy nhiên thời gian để đáp ứng lại những sự thay đổi trong kiến trúc, những thay đổi chính trong yêu cầu, và sự thay đổi kế hoạch (planning shift) hoặc sự lộn xộn trong tổ chức (organizational perturbation) rõ ràng sẽ gia tăng theo trong các giai đoạn kế tiếp (subsequent phases).

Trong hầu hết các vòng đời truyền thống (conventional life-cycle), các giai đoạn được đặt tên theo hoạt động chính trong giai đoạn đó: Phân tích yêu cầu, thiết kế, viết mã, kiểm tra unit, kiểm tra tích hợp (integration test), kiểm tra toàn hệ thống (system test). Các nỗ lực phát triển phần mềm truyền thống (conventional software development effort) nhấn mạnh vào quy trình tuần tự (sequential process), trong đó một hoạt động này được yêu cầu phải hoàn thành trước khi một hoạt động khác được bắt đầu. Trong quy trình lặp

(iterative process), mỗi một giai đoạn bao gồm mọi hoạt động, theo rất nhiều tỷ lệ rất khác nhau (proportions). Các cấp độ hoạt động (activity level) trong 4 giai đoạn của vòng đời sẽ được thảo luận trong chương 8. Các mục tiêu chủ yếu (primary objectives), hoạt động căn bản (essential activities), và các tiêu chuẩn đánh giá chung (general evaluation criteria) cho mỗi giai đoạn sẽ được thảo luận ở đây trong chương này.

5.2.Giai đoạn khởi đầu

(Inception phase)

Mục tiêu quan trọng nhất (overriding goal) của giai đoạn khởi đầu là nhằm đạt được sự tán thành (concurrence) giữa các cổ đồng về các mục tiêu của vòng đời cho dự án.

Các mục tiêu chính (primary objectives):

- Thiết lập phạm vi phần mềm của dự án và các điều kiện dự án (project's sofware scope and boundary condition) bao gồm cả khái niệm thao tác (operational concept), các tiêu chuẩn chấp nhận (acceptance criteria), và một sự hiểu biết rõ ràng về điều gì có và không có trong sản phẩm.
- Phân biệt các lớp sử dụng quan trọng (critical use cases) của hệ thống (system) và kịch bản công việc (primary scenarios of operation) mà sẽ đưa đến sự thoả thuận về các thiết kế chính (major design trde-offs).
- Đưa ra ít nhất một kiến trúc ứng cử (candidate architecture) đối lập với các kịch bản chính (primary scenarios).
- Ước lượng chi phí và thời gian biểu (cost and schedule) cho toàn bộ dự án (bao gồm những ước lượng cụ thể cho giai đoạn cụ thể hoá (elaboration phase)).
- Ước lượng các rủi ro tiềm tàng (potential risks) (đưa ra nguồn của các sự kiện không dự đoán được (sources of unpredictability)).

Các hoạt động chủ yếu (essential activities):

- Thành lập được phạm vi của dự án (formulating the scope of the project). Hoạt động này liên quan đến việc nắm bắt được các yêu cầu và khái niệm thao tác từ nguồn thông tin (information repository) miêu tả quan điểm của người sử dụng về các yêu cầu. Nguồn thông tin này phải đủ để xác định được không gian của bài toán (problem space) và tìm ra các tiêu chuẩn chấp nhận được (acceptance criteria) cho sản phẩm cuối cùng (end product).
- *Tổng hợp kiến trúc. Đánh giá các thoả thuận về thiết kế (design trade-offs), sự mô hồ của không gian bài toán (problem space ambiguities), các tài nguyên sẵn có cho không gian lời giải (available solution-space assets) (công nghệ và các thành phần đã có sẵn). Một nguồn thông tin được tạo ra phải đầy đủ để có thể đưa ra tính khả thi (feasibility) của ít nhất một kiến trúc đề cử và vạch ranh giới ban đầu để mà có thể đưa ra các ước lượng về chi phí, thời gian biểu và tài nguyên.*

- Lập kế hoạch và chuẩn bị khung công việc (business case). Chọn lựa cách quản lý rủi ro (alternatives for risk management), bố trí nhân sự (staffing), và sự cân bằng chí phí/thời gian biểu/lợi nhuận (cost/schedule/profitability trade-offs). Xác định cơ sở hạ tầng (công cụ, các quy trình, trợ giúp tự động) (infrastructure-tools, processes, automation support) để đáp ứng được nhiệm vụ phát triển của vòng đời (life-cycle development task).

Tiêu chuẩn đánh giá chủ yếu (evaluation criteria):

- Tất cả các cổ đông đều đồng ý về cách xác định của phạm vi (scope definition), các ước lượng về chi phí và thời gian biểu (cost and schedule estimates) không?
- Các yêu cầu đã được hiểu rõ chưa?
- Các ước lượng về chi phí, thời gian biểu, các quyền ưu tiên (prioriries), rủi ro, quy trình phát triển (development processes) đã chấp nhận được hay chưa?

- *Một mẫu kiến trúc (architecture prototype) đã đạt được tiêu chuẩn trước đó chưa (ý nghĩa chủ yếu của một mẫu kiến trúc là đưa ra một phương tiện (vehicle) nhằm nhận biết phạm vi và đánh giá mức độ chấp nhận được (credibility) của nhóm phát triển (development group) trong việc giải quyết những vấn đề công nghệ cụ thể (particular technical problem)).*
- Việc sử dụng các tài nguyên hiện tại (actual resource expenditures) so với kế hoạch có chấp nhận được không.

5.3. Giai đoạn cụ thể hoá

(Elaboration phase)

Dễ dàng chỉ ra được rằng giai đoạn chi tiết hoá là giai đoạn quan trọng nhất trong 4 giai đoạn của vòng đời. Vào giai đoạn cuối của giai đoạn này, phần công nghệ (engineering) được xem là đã hoàn thành và dự án đối mặt với vấn đề (reckoning): Quyết định đi hay không đi tiếp đến giai đoạn sản xuất. Đối với hầu hết các dự án, quyết định này tương ứng với sự chuyển tiếp từ một quy trình hoạt động (operation) ít rủi ro (low cost risk) đến một qui trình hoạt động với rủi ro cao hơn (higher cost risk). Trong khi tiến trình (process) phải luôn thích nghi với sự thay đổi, các hoạt động của giai đoạn cụ thể hoá phải đảm bảo rằng kiến trúc, các yêu cầu, kế hoạch phải đủ ổn định, rủi ro phải được làm giảm đi một cách thích hợp, phải đảm bảo rằng chi phí, thời gian biểu cho toàn bộ sự phát triển của dự án được ước lượng trong một khoản chấp

nhận được (acceptable range). Độ tin cậy phải phù hợp với mức độ cần thiết cho tổ chức để chuyển tới giai đoạn xây dựng với chi phí cố định (fixed-price construction phase). Trong giai đoạn cụ thể hoá, một mẫu kiến trúc thực hiện được (executable arrchitecture prototype) phải xây dựng một vài lần, tuỳ thuộc vào phạm vi, độ lớn, mức độ rủi ro, và tính mới lạ của dự án. Sự cố gắng này tập trung tới ít nhất một lớp sử dụng quan trọng (critical use case) được xác định trong giai đoạn khởi đầu (inception phase), điều này thường bộc lộ những rủi ro công nghệ cao nhất của dự án (top technical risks). Mặc dù một mẫu phát triển của các thành phần số lượng-chất lượng (prototype of production- quality components) thường là mục tiêu, nó cũng không loại trừ việc phát triển các mẫu mang tính thăm dò, thông báo (exploratory and throw-away prototypes) hơn nhằm làm giảm các rủi ro đặc trưng (specific risks) như sự cân bằng về thiết kế/yêu cầu (design/requirements trade-offs), sự phân tích tính khả thi thành phần (component feasibility analyses), hoặc sự thuyết phục các nhà đầu tư.

Các mục tiêu chủ yếu (primary objectives):

- Định vạch ranh giới kiến trúc một cách nhanh chóng cũng như thiết thực (baselining the architecture as rapidly as practical) (thiết lập configuration-managed snapshot trong đó các thay đổi được hợp lý hoá, theo dõi và bảo trì (rationalized, tracked and maintained)).
- Định vạch ranh giới viễn cảnh (vision).
- Định vạch ranh giới cho kế hoạch với độ tin cậy cao (high-fidelity) cho giai đoạn xây dựng (construction phase).
- Chứng minh được kiến trúc ranh giới (baseline architecture) sẽ hỗ trợ viễn cảnh (vision) với một chi phí chấp nhận được và trong một khoảng thời gian chấp nhận được (reasonable cost and reasonable time).

Các hoạt động chủ yếu (essential activities):

- Cụ thể hoá viễn cảnh (elaborating the vision). Hoạt động này liên quan đến việc thiết lập sự hiểu biết ở mức độ tin cậy cao (high-fidelity) về các lớp sử dụng chính (critical usse cases) cái mà đưa đến các quyết định về kế hoạch hoặc kiến trúc (architectural or planning decisions).

- Chi tiết hoá qui trình và cơ sở hạ tầng. Thiết lập qui trình xây dựng, các công cụ, sự hỗ trợ tự động hoá qui trình, các mốc trung gian và các tiêu chuẩn đánh giá riêng cho từng cái (respective evaluation criteria).
- Chi tiết hoá kiến trúc và lựa chọn các thành phần. Các thành phần tiềm năng (potential components) phải được xem xét, phải có các quyết định đủ rõ ràng để có thế xác định được chi phí, thời gian biểu của giai đoạn xây dựng. Các thành phần kiến trúc được lựa chọn sẽ được tích hợp lại và đánh giá theo kịch bản theo các kịch bản chính (primary scenarios). Các bài học rút ra là có thể nhiều kết quả tốt trong việc thiết kế lại kiến trúc cũng như là chọn lựa việc thiết kế sẽ được cân nhắc hoặc yêu cầu trong các hoạt động này.

Các tiêu chuẩn đánh giá chủ yếu (primary evaluation criteria):

- Viễn cảnh (vision) có ổn định hay không?
- Kiến trúc có ổn định không?
- Luận chứng (executable demontration) có nêu ra được các yếu tố rủi ro chính đã được xem xét đến và đã được giải quyết chưa?
- Kế hoạch cho giai đoạn xây dựng đã đủ tin cậy chưa, và kế hoạch đã có đủ các ước lượng cơ bản chấp nhận được (back up with credible basis of estimate)?
- Tất cả các cổ đông đã đồng ý rằng sẽ có được viễn cảnh hiện tại (current vision) nếu kế hoạch được thực hiện và phát triển theo đúng kiến trúc?
- Các chi tiêu về tài nguyên hiện tại có phù hợp với kế hoạch không?

5.4. Giai đoạn xây dựng

(Construction phase)

Trong giai đoạn xây dựng (construction phase), tất cả các thành phần còn lại và các đặc điểm của ứng dụng (application features) được tích hợp, kiểm tra kỹ lưỡng. Phần mềm mới được phát triển (newly developed software) sẽ được tích hợp (intergrate) khi có yêu cầu. Giai đoạn xây dựng cũng đưa ra qui trình sản xuất, trong đó nhấn mạnh vào việc quản lý tài nguyên (resource management) và điều khiển quá trình hoạt động (coltronlling operations) nhằm tối ưu chi phí, thời gian biểu và chất lượng. Với ý nghĩa đó, việc quản lý sẽ trải qua một sự chuyển tiếp từ giai đoạn khởi đầu và giai đoạn cụ thể hoá sang giai đoạn xây dựng và giai đoạn chuyển tiếp. Rất nhiều dự án đủ lớn để làm xuất hiện việc gia tăng sự xây dựng song song (parallel). Các hoạt động song song làm tăng đáng kể các phiên bản triển khai được (deployable releases); nó cũng làm gia tăng sự phức tạp của quá trình quản lý tài nguyên và đồng bộ hoá các luồng công việc, các nhóm phát triển (synchronization of workflows and teams). Một kiến trúc mạnh (robust architecture) liên quan nhiều đến một kế hoạch dễ hiểu (understandable plan). Nói cách

khác, một trong những đặc trưng quan trọng cuả bất kỳ kiến trúc nào là sự dễ dàng cho xây dựng. Đây là môt lý do mà một sự phát triển cân bằng (balanced development) về kiến trúc và kế hoạch được nhấn mạnh trong giai đoạn cụ thể hoá.

Các mục tiêu chủ yếu (primary objectives);

- Tối thiểu hoá chi phí phát triển (development costs) bằng cách tối ưu tài nguyên (optimizing resources) và tránh làm những công việc vụn không cần thiết, làm lại (unnecessary scrap and rework)
- Đạt được chất lượng thoả đáng nhanh cũng như là thiết thực (adequate quality)
- Đạt được các phiên bản thích hợp cho sử dụng (Alpha, Beta và các phiên bản kiểm tra khác (alpha, beta and other test releases)) nhanh cũng như là thiết thực.

Các hoạt động chủ yếu (essential activities):

- Quản lý tài nguyên (resource management), điều khiển, tối ưu qui trình (control, process optimization).
- Phát triển hoàn chỉnh các thành phần (complete component development) và kiểm tra so với các tiêu chuẩn ước lượng (evaluation criteria).
- Đánh giá các phiên bản sản phẩm (product releases) so với các tiêu chuẩn chấp nhận của viễn cảnh (acceptance crteria of the vision).

Các tiêu chuẩn ước lượng chủ yếu (evaluation criteria):

- Vạch ranh giới sản phẩm (product baseline) đã đủ hoàn thiện chưa để có thể triển khai nó trong cộng đồng người sử dụng (user community) (Việc tồn tại các lỗi (defects) không phải là trở ngại (obstacles) để đạt được mục đích của các phiên bản tiếp theo)?
- Vạch ranh giới sản phẩm đã ổn định chưa để triển khai trong cộng đồng người sử dụng (các thay đổi sắp có (pending changes) cũng không phải là trở ngại để đạt được các mục tiêu của phiên bản tiếp theo (next release))?
- Tất cả các cổ đông đã sẵn sàng cho việc chuyển giao (transition) cho cộng đồng người sử dụng chưa?
- Chi phí tài nguyên hiện tại so với chi phí theo kế hoạch (planned expenditures) có chấp nhận được không?

5.5. Giai đoạn chuyển tiếp

(Transition phase)

Giai đoạn chuyển tiếp (transition phase) được bắt đầu khi vạch ranh giới đã đủ hoàn thiện để được triển khai cho người sử dụng cuối cùng (end–user domain). Điều này thường yêu cầu một phần của hệ thống (usable subset of the system) đã phải đạt được chất lượng chấp nhận được (acceptable quality levels) và đã có sự cung cấp tài liệu cho người sử dụng (user documentation) do đó sự chuyển giao tới người sử dụng sẽ cho các kết quả tích cực (positive results).

Giai đoạn này gồm những hoạt động sau :

1. Kiểm tra phiên bản beta (beta testing) để thông qua (validate) một hệ thống mới so điều mong đợi của người dùng (user expectations).
2. Kiểm tra phiên bản beta và việc thực hiện song song tương xứng với hệ thống tài sản mà nó đang thay thế (relative to legacy system it is replacing).
3. Sự chuyển đổi cơ sở dữ liệu thi hành (conversion of operational databases).
4. Đào tạo người dùng và những người bảo trì (training users and maintainers).

Giai đoạn chuyển giao sẽ kết thúc khi vạch ranh giới triển khai (deployment baseline) đạt được toàn bộ viễn cảnh (complete vision). Đối vơí một vài dự án, điểm kết thúc vòng đời này (life-cycle end point) có thể trùng với điểm bắt đầu (starting point) của phiên bản tiếp theo của sản phẩm. Đối với những dự án khác, nó lại trùng với việc phân phối toàn bộ thông tin (information sets) cho các nhà sản xuất thứ ba (third party) có trách nhiệm trong việc thực hiện, bảo trì, nâng cấp (operation, maintenance, enhancement).

Giai đoạn chuyển giao nhấn mạnh vào hoạt động đưa phần mềm đến tận tay người sử dụng (place software into the hands of the users). Thông thường, giai đoạn này bao gồm một số lần lặp (iterations), bao gồm phiên bản beta (beta releases), các phiên bản sẵn có (availability releases), các phiên bản sửa lỗi (bug-fix releases), và các phiên bản nâng cấp (enhancement releases). Đã có rất nhiều nỗ lực trong việc cung cấp tài liệu hướng người dùng (user-oriented documentation), đào tạo người dùng, hỗ trợ người dùng trong lần sử dụng sản phẩm đầu tiên (initial product use), đáp ứng lại các phản hồi từ người dùng. (Tại giai đoạn này trong vòng đời, các phản hồi của người sử dụng phải được hạn chế (confined) trong việc điều chỉnh của sản phẩm, định dạng, cài đặt (product tuning, configuring, installing and usability issues)).

Các mục tiêu chủ yếu (primary objectives):

- Đạt được khả năng tự hỗ trợ người dùng (user self-supportability).
- Đạt được sự tán thành các cổ đông về vạch ranh giới triển khai (deployment baseline) phù hợp với các tiêu chuẩn đánh giá của viễn cảnh (evaluation criteria of the vision).
- Đạt được vạch ranh giới sản phẩm cuối cùng nhanh (rapidly), chi phí hiệu quả (cost-effectively) cũng như là thiết thực (practical).

Các hoạt động chủ yếu (essential activities):

- Sự đồng bộ hoá và tích hợp (synchronization and integration) của sự gia tăng xây dựng (concurrent construction increments) vào các vạch ranh giới triển khai phù hợp (consistent deployment baselines) .
- Chuyển giao các công nghệ cụ thể (deployment-specific engineering) (cutover, đóng gói và sản xuất (commercial packaging and production), phát triển công cụ giới thiệu sản phẩm (sales rollout kit development), đào tạo nhân sự (field personnel trainning)).
- Đánh giá vạch ranh giới triển khai so với toàn bộ viễn cảnh và các tiêu chuẩn chấp nhận trong tập các yêu cầu (requirement set).

Các tiêu chuẩn đánh giá (evaluation criteria):

- Người dùng đã hài lòng chưa?
- Tất cả các chi phí về tài nguyên hiện tại so với chi phí cho kế hoạch có chấp nhận được không?

Tổng kết:

Mỗi một trong bốn giai đoạn kể trên bao gồm một hay nhiều lần lặp (iterations) trong đó vài khả năng công nghệ (technical capability) được tạo ra dưới dạng có thể dẫn chứng được (demonstrable form) và được đánh giá so với tập các tiêu chuẩn. Một vòng lặp (được giới thiệu trong chương 8) sẽ biểu diễn một chuỗi các hoạt động (sequence of activites) trong đó có một sự kiện trung gian (một mốc (milestone) sẽ được giới thiệu trong chương 9); phạm vi và kết quả của lần lặp (iteration) có được thông qua các mẫu rời rạc (discrete artifacts) (được thảo luận ở chương 6). Trong khi đó, hầu hết các mốc ở cuối mỗi giai đoạn sử dụng các tiêu chuẩn đánh giá chuẩn (formal versions of evaluation criteria), một số ít thì sử dụng các tiêu chuẩn

đánh giá không chuẩn (informal). Mỗi giai đoạn tương ứng với việc hoàn thành một số lần lặp (completion of iterations) để đạt được trạng thái cho trước của toàn bộ dự án (given overall project state). Sự chuyển tiếp từ một giai đoạn này sang giai đoạn khác thì là một quyết định kinh doanh hơn là sự hoàn thành của một hoạt động phát triển phần mềm cụ thể. Sự chuyển tiếp các giai đoạn trung gian là điểm tựa chính của qui trình phần mềm, khi mà các viễn cảnh công nghệ và quản lý (technical and management perspectives) được đồng bộ hoá và sự thống nhất giữa các cổ đông đạt được với sự liên quan đến sự hiểu biết hiện tại về các yêu cầu, thiết kế và kế hoạch để hoàn thành.

Chương 6

Tạo tác qui trình

(Artifacts of the process)

Điểm lưu ý :

- Tạo tác một quá trình bao gồm năm phần : điều hành, yêu cầu, thiết kế, cài đặt, triển khai (deploymend).
- Mô hình điều hành chứa đựng thông tin để đồng bộ hoá những mong muốn của nhà đầu tư.
- Tạo tác yêu cầu, thiết kế, cài đặt, triển khai có sự trợ giúp của các công cụ ghi chép chuẩn xác kèm theo tính năng phân tích và lựa chọn tự động.

> *Các dự án phần mềm truyền thống thường đi theo mô hình phát triển phần mềm theo hướng tuần tự : xây dựng yêu cầu, kết cấu một mô hình thiết kế có thể đáp ứng được yêu cầu đó, tiến hành cài đặt theo mẫu thiết kế, dịch và thử kết quả cài đặt để tiến hành triển khai. Quá trình này có thể thu gọn hơn, hầu như chỉ trong trường hợp phát triển theo đơn dặt hàng trong đó việc mô tả thiết kế, mô tả cài đặt, và biểu diễn việc triển khai được xếp gần ngang hàng nhau. Ví dụ, một chương trình đơn lẻ nhằm chạy trên một máy tính đơn lẻ của một chủng loại máy cụ thể và được xây dựng hoàn toàn dựa trên các thành phần có chức năng riêng biệt theo thói quen có thể được kết cấu với một sự điều chỉnh định hướng giữa các bộ phận mô tả.*

Tuy nhiên, cách tiếp cận này không đáp ứng được yêu cầu của các hệ thống phần mềm ngày nay, trong đó độ phức tạp hệ thống (nhiều kích cỡ) dẫn đến nhiều sự mạo hiểm và những mối quan hệ tinh tế theo dõi được làm cho bạn không thể dễ dàng chuyển đổi chúng sang dạng tuần tự một cách giản dị thái quá. Phần lớn những hệ thống hiện đại cấu thành từ rất nhiều phần (một số là theo đơn đặt hàng, một vài được tái sử dụng, một vài là sản phẩm thương mại) có xu hướng thực thi trên các mạng hỗn tạp hay là môi trường phân bố. Chúng đòi hỏi không ngừng được cải tiến quá trình tạo mẫu và một cách tiếp cận hoàn toàn mới tới vấn đề tính theo dõi được .

Trải qua 20 năm, ngành công nghiệp phần mềm đã trưởng thành và thay đổi cách thức điều hành sang dạng có thể lặp lại. Thay vì được xây dựng tuần tự, các mẫu được tiến triển cùng với nhau; và các mối ràng buộc, các mức độ trìu tượng hóa khác nhau, các cấp độ độc lập của modul được cân đối giữa các sự lựa chọn cạnh tranh. Đề tài quay vòng qua những dự án thành công cho thấy quá trình tạo tác phần mềm có thể được cùng tiến triển đồng thời với nhau trong đó có sự cân đối giữa các mức độ chi tiết. Tạo tác không tiến triển theo một chiều, tuyến tính từ yêu cầu đến thiết kế rồi đến cài đặt sau đó đến triển khai. Sự lựa chọn việc cài đặt và triển khai ảnh hưởng đến cách phát biểu yêu cầu và phương pháp thiết kế. Thông tin và quyết định có thể được chuyển tiếp giữa các quá trình tạo tác theo nhiều cách khác nhau. Mục đích của một

phương pháp phát triển tốt là loại bỏ những ràng buộc không cần thiết hấp tấp trong khi thiết kế và điều tiết những ràng buộc kỹ thuật thực sự.

Sự phát triển quay vòng đã có ảnh hưởng như thế nào đến tiến trình tạo tác mở. Điểm khác biệt nổi bật với cách tiếp cận truyền thống là với mỗi pha của vòng đời công việc không tiến triển theo một chiều tuyến tính đơn giản, hay là tạo tác một cách đều đều từ mẫu này sang mẫu khác. Thay vì đó, trọng điểm của công việc luôn được di chuyển giữa các mẫu, tăng cường tính năng của toàn bộ hệ thống nhờ vào quá trình điều chỉnh chiều sâu và chiều rộng của thông tin.

6.1.Tập mẫu

Để có thể quản lý được sự phát triển phần mềm, mỗi tập hợp thông tin riêng biệt được tổ chức thành tập mẫu (tập tạo tác). Mỗi tập gồm các mẫu có liên quan chặt chẽ và được biểu diễn dưới dạng chuẩn (như là văn bản, C++, Visual Basic, Java, mẫu tài liệu hay bảng tính chuẩn, mô hình UML). Trong khi một tập đề cập đến khía cạnh hoàn chỉnh của hệ thống, tạo tác đề cập đến những thông tin cố kết tiêu biểu được phát triển và nhìn nhận như là một thực thể độc lập. Trong bất kỳ một tổ chức, dự án hay hệ thống nào, một số mẫu hay thậm chí một số tập là tầm thường và không cần thiết. Nói chung, một số thông tin cần được thu thập trong mỗi tập để làm hài lòng tất cả các nhà đầu tư.

Tạo tác vòng đời phần mềm tạm chia thành 5 tập tách biệt dựa theo ngôn ngữ cơ bản của tập đó: điều hành (định dạng phi thể thức), yêu cầu (văn bản có cấu trúc và mô hình của bài toán), thiết kế (mô hình không gian giải pháp), cài đặt (ngôn ngữ lập trình và các file nguồn liên quan), triển khai (ngôn ngữ máy và các file liên quan).

Sự nổi lên của các ký hiệu kỹ thuật chính xác và mạnh mẽ cho bước tạo tác yêu cầu và thiết kế trong khi xây dựng kiến trúc đầu tiên là một bước tiến kỹ thuật đáng kể. Nói chính xác, ngôn ngữ mẫu thống nhất (Unified Modelling Language - UML) đã đưa ra một định dạng trình bày thích hợp, gọi là mô hình trực quan chỉ rõ cú pháp và ngữ nghĩa của qui trình tạo tác yêu cầu và thiết kế. Mô hình trực quan sử dụng UML là những ghi chép sơ khai cho sự bắt đầu chu kỳ tạo tác. Tập mẫu được chỉ ra trong hình 6.1; mục đích và ghi chú về chúng sẽ được giải thích trong phần tiếp theo.

6.1.1.Tập điều hành

Tập điều hành gồm những mẫu liên quan đến việc lập kế hoạch và xúc tiến quy trình. Những mẫu này sử dụng văn phạm phi thể thức, gồm văn bản, hình ảnh, và bất cứ phương tiện biểu diễn nào cần để đi đến thỏa thuận giữa những cán bộ dự án (quản lý dự án, kiến trúc sư, người phát triển, người kiểm tra, bộ phận tiếp thị, nhà quản trị), giữa những nhà đầu tư (tổ chức đầu tư, người sử dụng, giám đốc dự án phần mềm, giám đốc công ty, người điều tiết trung gian), hay là giữa những cán bộ dự án và những nhà đầu tư. Những mẫu cụ thể trong tập này bao gồm : cấu trúc phân chia công việc (chia nhỏ công việc và cơ chế hiệu chỉnh tài chính), công việc kinh doanh (giá cả, thời hạn, lợi nhuận ước tính), tháo gỡ đặc tả (phạm vi, kế hoạch, mục đích của việc loại bỏ vạch ranh giới), kế hoạch phát triển phần mềm (tiến hành dự án cụ thể), tách lớp các mô tả (kết quả việc loại bỏ vạch ranh giới),

đánh giá tình hình (hình ảnh có tính chất định kỳ của sự diễn tiến dự án), trật tự thay đổi phần mềm (mô tả sự thay đổi các đường ranh giới rời rạc), tài liệu mô tả việc triển khai (kế hoạch cắt giảm, khoá đào tạo, công cụ giới thiệu sản phẩm), môi trường (phần cứng và phần mềm), công cụ (tự động hoá quy trình, chứng minh bằng tài liệu, việc đào tạo thêm cần thiết để xúc tiến quy trình mô tả trong kế hoạch phát triển phần mềm và sự trình bày mô hình kỹ thuật).

<table>
<tr>
<td>Tập yêu cầu
(Requirement set)
1. Tài liệu trực quan
2. Mô hình yêu cầu</td>
<td>Tập thiết kế
(Design set)
1. Mô hình thiết kế
2. Mô hình kiểm tra
3. Mô tả kiến trúc phần mềm</td>
<td>Tập cài đặt
(Implementation set)
1. Ranh giới mã nguồn
2. Tệp kèm theo trong khi biên dịch
3. Thành phần có thể thi hành được.</td>
<td>Tập triển khai
(Deployment set)
1. Ranh giới sản phẩm tích hợp thi hành được
2. File kết hợp khi chạy chương trình
3. Tài liệu hướng dẫn người sử dụng</td>
</tr>
<tr>
<td colspan="2">Tập điều hành (Management set)
Tạo tác kế hoạch
1. Cấu trúc phân chia công việc
2. Công việc kinh doanh
3. Tháo gỡ các đặc tả
4. Kế hoạch phát triển phần mềm</td>
<td colspan="2">Tạo tác quá trình hoạt động
5. Mô tả việc phát hành
6. Đánh giá tình hình
7. Cơ sở dữ liệu thay đổi trật tự phần mềm
8. Tài liệu quá trình triển khai
9. Môi trường</td>
</tr>
</table>

Hình 6.1 : *Tổng quan về tập mẫu*

Tạo tác tập điều hành được ước định, đánh giá và dự tính dựa trên các tiêu chí sau :

- Sự phê bình xác đáng của nhà đầu tư
- Phân tích những thay đổi giữa phiên bản tạo tác này so với phiên bản trước (xu hướng điều hành và những thay đổi đặc tính dự án về giá cả, thời hạn, và chất lượng).

- Sự thể hiện những cột mốc cân đối chính giữa các mẫu, ví dụ như sự chính xác của tác vụ kinh doanh và tạo tác trực quan.

6.1.2 .Tập công nghệ (The engineering sets)

Tập công nghệ gồm có tập yêu cầu, tập thiết kế, tập cài đặt, và tập triển khai. Cơ chế chủ đạo để đánh giá chất lượng tiến hóa của mỗi tập mẫu là sự di chuyển thông tin từ tập này sang tập khác, nhờ đó duy trì cân đối được sự thông hiểu lẫn nhau giữa các mẫu yêu cầu, thiết kế, cài đặt và triển khai. Mỗi một thành phần của bộ mô tả hệ thống đều tiến hóa theo thời gian.

Tập yêu cầu

Văn bản có cấu trúc dùng để trình bày trực quan, cung cấp tư liệu về phạm vi dự án, chứng thực cho hợp đồng giữa nhà đầu tư và tổ chức dự án. Định dạng phi thể thức có thể dùng để đặc tả thêm (như là yêu cầu điều tiết), sử dụng mô hình và nguyên mẫu diễn tả được yêu cầu. Ký hiệu UML dùng để biểu diễn kỹ thuật của mô hình yêu cầu (mô hình trong từng trường hợp, mô hình phạm vi). Tạo tác tập yêu cầu là khung cảnh kỹ thuật chính để ước định ba tập tạo tác khác và là cơ sở cho phần kiểm tra.

Tạo tác yêu cầu được ước tính, đánh giá, đo đạc dựa trên các tiêu chí sau :

- Phân tích sự phù hợp với các chi tiết kỹ thuật được tháo gỡ của tập điều hành
- Phân tích sự phù hợp giữa tầm nhìn trực quan và mô hình yêu cầu.
- Sắp đặt lại các tập thiết kế, cài đặt và triển khai để ước định được sự phù hợp và hoàn chỉnh về cân xứng ngữ nghĩa thông tin trong các tập khác nhau.
- Phân tích những thay đổi giữa phiên bản mẫu yêu cầu này so với phiên bản trước (phần thừa, làm lại, xu hướng loại bỏ sai sót).
- Sự phê bình chủ quan về chất lượng của các mặt khác.

Tập thiết kế

Ký pháp UML dùng để vạch ra mô hình thiết kế của giải pháp. Tập thiết kế chứa nhiều thành phần của không gian giải pháp với các mức độ trìu tượng hóa khác nhau (tính đồng nhất, thuộc tính, quan hệ tĩnh, tương tác động). Mô hình thiết kế kèm theo thông tin có cấu trúc mang tính hành vi đủ để xác định được một hóa đơn của vật chất (số lượng và đặc tả các thành phần ban đầu, nguyên liệu, nhân công, và các chi phí trực tiếp khác). Thông tin của mô hình thiết kế có thể rất minh bạch, và trong nhiều trường hợp tự động dịch chuyển thành một bộ phận con của tập tạo tác cài đặt và triển khai. Tạo tác thiết kế cụ thể bao gồm mô hình thiết kế, mô hình kiểm tra, và miêu tả kiến trúc phần mềm (đoạn thông tin trích từ mô hình thiết kế thích hợp để mô tả kiến trúc).

Tập thiết kế được ước định, đánh giá và đo đạc dựa trên nhưng tiêu chí sau :

- Phân tích tính thích ứng nội tại và chất lượng của mô hình thiết kế
- Phân tích sự phù hợp với mô hình yêu cầu
- Bản dịch sang tập cài đặt và triển khai cùng với các ghi chú (ví dụ : tính theo dõi được, sinh mã nguồn, biên dịch, liên kết) để ước định mức độ phù hợp và hoàn thiện của vấn đề cân xứng ngữ nghĩa giữa thông tin trong các tập.
- Phân tích những thay đổi giữa phiên bản mô hình thiết kế này so với phiên bản trước (phần thừa bị loại bỏ, phần làm lại, xu hướng loại bỏ sai sót).
- Sự phê bình chủ quan về chất lượng của các mặt khác

Vì khả năng phân tích tự động trong thiết kế mô hình còn bị hạn chế nên con người vẫn phải đảm nhận phần lớn công việc đó. Tình hình này có thể sẽ thay đổi trong vài năm tới với sự lớn mạnh của các công cụ phân tích mô hình thiết kế có hỗ trợ tập hợp ngữ điệu ngôn ngữ, phân tích độ phức tạp, phân tích kiểu dáng, phân tích đánh giá kinh nghiệm, và cả phân tích mức độ phù hợp.

Tập cài đặt

Tập cài đặt gồm có mã nguồn (ký pháp ngôn ngữ lập trình) - sự cài đặt rõ ràng cho một thành phần (hình thức, giao diện, những mối quan hệ phụ thuộc) và mọi yếu tố cần thiết để có thể kiểm tra sửa lỗi một thành phần riêng lẻ. Những bộ phận thi hành được này là những thành phần gốc cần thiết cho việc xây dựng nên sản phẩm sau cùng, gồm có thành phần chuyên dụng, giao diện lập trình ứng dụng (APIs) của một công cụ phát triển mang tính thương mại, và APIs hay là các thành phần tái sử dụng thừa kế từ các file chương trình nguồn được viết trong một ngôn ngữ lập trình nào đó (như Ada95, C++, Visual Basic, Java, Assembly). Tạo tác tập cài đặt có thể được dịch (biên dịch và kết nối) thành một bộ phận con của tập triển khai (mục tiêu cuối cùng có thể thi hành được). Tạo tác cụ thể gồm có cơ sở mã nguồn có chỉ dẫn và các file kèm theo (kịch bản biên dịch, cơ sở hạ tầng quản lý cấu hình, file cấu hình), cơ sở mã nguồn kiểm tra kèm theo chỉ dẫn và các file cần thiết (file dữ liệu kiểm tra, file kết quả kiểm tra), bộ phận thi hành được độc lập, bộ phận điều khiển thi hành kiểm tra.

Tập cài đặt có dạng con người có thể đọc được và được ước đoán, đánh giá, và tính toán qua các nhân tố sau :

- Phân tích mức độ phù hợp với mô hình thiết kế
- Dịch sang tập ký hiệu triển khai (ví dụ, biên dịch và kết nối) để ước đoán độ phù hợp và hoàn thiện giữa các tập mẫu.
- Đánh giá bộ phận nguồn và tệp tin thi hành qua tiêu chuẩn ước đoán tương ứng bằng cách kiểm tra, phân tích, chứng minh, thử nghiệm.
- Thi hành việc kiểm tra các bộ phận độc lập bằng các công cụ trợ giúp tự động so sánh kết quả lý thuyết với kết quả thực tế.

- Phân tích thay đổi giữa phiên bản cài đặt này với phiên bản trước (phần loại bỏ, làm lại, xu hướng loại bỏ sai xót)
- Quan điểm khách quan về chất lượng của những khía cạnh khác.

Tập triển khai

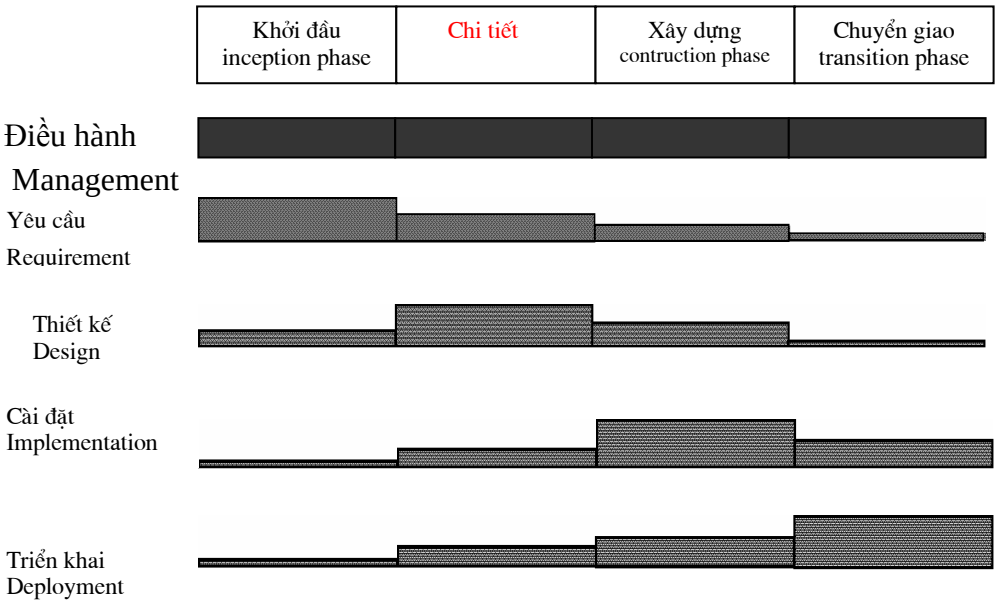
Tập triển khai gồm có phân phát cho người dùng, ký hiệu ngôn ngữ máy, phần mềm thi hành được, kịch bản xây dựng, kịch bản cài đặt, dữ kiệu cần thiết để thi hành trên môi trường đích. Ký hiệu ngôn ngữ máy này là thành phần sản phẩm ở dạng đích để phân phát tới người dùng. Thông tin tập triển khai có thể được cài đặt, thi hành trong một số tình huống giả định (thử nghiệm), và được tái cấu hình động để đáp các tính năng yêu cầu của sản phẩm. Tạo tác cụ thể gồm có ranh giới thi hành được và các file kèm theo khi chạy chương trình, tài liệu hướng dẫn người dùng.

Tập triển khai được ước đoán, đánh giá, và tính toán qua các nhân tố sau :

- Kiểm tra kịch bản sử dụng và thuộc tính chất lượng định nghĩa trong tập yêu cầu để đánh giá độ phù hợp và hoàn thiện và sự cân đối ngữ nghĩa giữa thông tin trong hai tập
- Kiểm tra sự phân chia, tái tạo, chiến lược phân bố đưa các thành phần của tập cài đặt vào các tài nguyên vật lý của hệ thống triển khai (kiểu nền tảng, kiểu số, cấu trúc liên kết mạng)
- Kiểm tra lại tình huống định nghĩa giả định trong tài liệu hướng dẫn người dùng như là cài đặt, tái cấu hình động hướng người dùng, xu hướng cách dùng, sự điều hành không bình thường
- Phân tích thay đổi giữa tập triển khai hiện hành so với tập cài đặt trước (xu hướng loại bỏ sai xót, thay đổi hiệu năng)
- Quan điểm khách quan về chất lượng của những khía cạnh khác.

Nhân tố căn bản để lựa chọn tập điều hành, yêu cầu, thiết kế, cài đặt, và triển khai là không có tính khoa học. Thành công là tối ưu việc trình bày hoạt động quy trình, tạo tác, và mục đích. Một số nhân tố căn bản ảnh hưởng đến khuôn khổ nhận thức sẽ được trình bày tiếp theo. Mặc dù có một số ngoại lệ nhỏ đối với sự tổng quát hóa, chúng rất hữu ích để hiểu toàn bộ tập tạo tác mẫu.

Mỗi tập tạo tác dùng các ký hiệu khác nhau để diễn tả tạo tác tương ứng. Ký hiệu tập điều hành (văn bản phi thể thức, hình ảnh, ký hiệu dùng trợ giúp phát triển phần mềm) mô tả kế hoạch, quy trình, mục tiêu, tiêu chuẩn chấp nhận. Ký hiệu yêu cầu (văn bản có cấu trúc và mô hình UML) mô tả khung cảnh kỹ thuật và khái niệm thi hành được. Ký hiệu thiết kế (dạng UML) mô tả chi tiết kỹ thuật (thiết kế kiến trúc, thiết kế bộ phận). Ký hiệu cài đặt (ngôn ngữ lập trình) mô tả cácc khối xây dựng của giải pháp ở dạng con người có thể đọc được. Ký hiệu triển khai (tệp dữ liệu và thi hành) mô tả giải pháp ở dạng ngôn ngữ máy.



Hình 6-2 : *Vòng đời tập trung trọng điểm vào tập mẫu*

Mỗi tập tạo tác chiếm một ưu thế nổi bật trong mỗi pha của vòng đời; những tập khác tạm dừng lại được cân bằng. Minh hoạ ở hình 6-2, mỗi pha có một trọng điểm nổi bật : yêu cầu là trọng điểm của giai đoạn khởi đầu thiết kế, giai đoạn dàn dựng tỉ mỉ; cài đặt, giai đoạn xây dựng; triển khai, giai đoạn chuyển giao. Tạo tác điều hành cũng tiến hóa, nhưng với một mức độ không đổi qua vòng đời.

Phần lớn những công cụ phát triển phần mềm hiện nay đáp ứng một trong năm tập mẫu sau :

1. Điều hành : lập lịch, phân luồng công việc, hiệu chỉnh sai sót, điều hành sự thay đổi, cung cấp tài liệu, bảng tính, quản lý tài nguyên, công cụ giải trình.
2. Yêu cầu : công cụ quản lý yêu cầu
3. Thiết kế : công cụ tạo mô hình trực quan
4. Cài đặt : chương trình biên dịch / gỡ rối, công cụ phân tích mã, công cụ phân tích thông tin của những thử nghiệm, công cụ điều hành việc thử nghiệm.
5. Triển khai : công cụ kiểm tra tự động và đánh giá mức độ bao trùm của thử nghiệm, thành phần thương mại (hệ điều hành, giao diện người dùng đồ hoạ - GUIs, hệ quản trị cơ sở dữ liệu - DBMSs, mạng, thành phần trung gian), và công cụ cài đặt.

Việc phân chia trách nhiệm trong nhóm dự án không khó thực hiện, sẽ được sắp đặt trong tiến trình luồng công việc trình bày ở chương 8.

Phân lập việc cài đặt và triển khai

Sự tách rời cài đặt (mã nguồn) ra khỏi triển khai (mã thi hành được) là rất quan trọng vì mỗi công việc có những mối liên quan riêng. Cấu trúc thông tin đưa đến người sử dụng (điển hình là tổ chức thử nghiệm) rất khác biệt so với cấu trúc mã nguồn. Những quyết định về kỹ thuật gây ảnh hưởng tới chất lượng triển khai nhưng không có quan hệ bao hàm đối với thiết kế và cài đặt bao gồm :

- Thiết lập thông số cấu hình động (kích thước bộ đệm, bảng màu, lượng máy phục vụ, số lượng máy khách đồng thời hoạt động, tệp dữ liệu, tham số chạy chương trình) .
- Hiệu quả của việc tối ưu biên dịch / kết nối (ví dụ tối ưu không gian nhớ làm giảm khả năng tối ưu tốc độ).
- Đặc tính trong một số chiến lược phân bổ (tập trung trái ngược với phân bố, mạch chính và mạch ngầm, cân đối động gánh nặng, sao lưu nóng trái ngược với cơ chế điểm đánh dấu / quay lui).
- Ràng buộc tài nguyên ảo (bộ miêu tả tệp, thu thập rác, kích thước vùng heap, kích thước tối đa bản ghi, sự luân phiên sử dụng các vùng nhớ trên đĩa)
- Vấn đề tranh chấp tài nguyên (bế tắc và một số tình huống ngoại lệ)
- Điểm khác biệt về sự thi hành và cách chạy chúng trên các nền tảng cụ thể

Những thông tin cấu hình này là nguồn dữ liệu kỹ thuật quan trọng lấy được trong tập cài đặt (nếu nó được nhúng vào trong mã nguồn) hay là trong tập triển khai (nếu nó được nhúng vào trong tệp dữ liệu, tệp cấu hình, kịch bản cài đặt, hay những yếu tố mục tiêu cá biệt khác). Trong những hệ thống có thể cấu hình động hay là bộ phận di động, nên tách rời công việc cài đặt mã nguồn với môi trường đích (vì lý do hiệu suất, thích ứng động, quản lý sự thay đổi mã nguồn). Với cách tiếp cận này, việc cài đặt có thể được tách riêng từ kiểu mẫu nền tảng thực tế và từ số lượng cũng như hình thái cấu trúc cơ sở hạ tầng thông tin, gồm có hệ điều hành, bộ phận chuyển tiếp, mạng, hệ quản trị cơ sở dữ liệu.

Ví dụ, xem xét kiến trúc phần mềm của một triệu dòng mã nguồn viết cho hệ thống báo động tên lửa (dự án này trình bày chi tiết trong phụ lục D) với yêu cầu chịu lỗi và khả năng xử lý dữ liệu cực kỳ khắt khe. Trong dự án này, có những khác biệt đáng kể giữa những cấu hình có thể thi hành được xây dựng từ cùng một tập nguồn.

- Phiên bản chỉ gồm luồng xử lý chính chạy trên máy chủ thử nghiệm để kiểm tra một tập con các thử nghiệm.
- Phiên bản gồm dòng xử lý cơ bản và xử lý sao lưu trên máy thử nghiệm, có thể thực thi một sự tái cấu hình logic giả định.
- Phiên bản có chức năng tương đương với hai cấu hình trên có thể thi hành trên các bộ xử lý đích để đánh giá số lượng vật liệu đưa vào theo yêu cầu và thời gian đáp ứng với luồng tới hạn giả định trên một cấu hình đích đề cử.

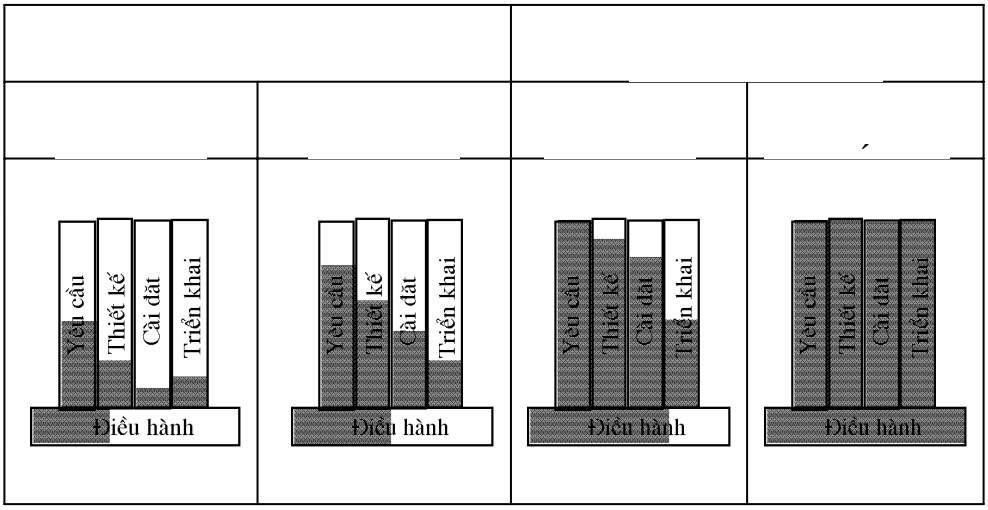
- Phiên bản có thể thi hành đồng thời một tiến trình phục vụ chính trên một bộ xử lý đích, và một tiến trình bóng được sao lưu lại trên một bộ xử lý độc lập, một tiến trình kiểm tra / thi hành cả hai máy đích, một tập hợp các tiến trình khách độc lập chạy trên các máy trạm. Phiên bản cuối cùng có thể hỗ trợ phạm vi tái cấu hình động rộng rãi, về cơ bản là cấu hình đích cuối cùng.

Triển khai các sản phẩm thương mại đến khách hàng có thể mở rộng ra thành một phạm vi thử nghiệm và triển khai cấu hình rộng. Ví dụ, những sản phẩm trung gian có tính năng tốt, sản phẩm đáng tin cậy đã được thử nghiệm trên nhiều nền tảng, gồm có hệ điều hành máy trạm, xử lý nhúng, hệ điều hành máy tính lớn, hệ điều hành xử lý thời gian thực. Cấu hình sản phẩm thích hợp với nhiều trình biên dịch và ngôn ngữ, có thể thi hành trên nhiều phần mềm mạng. Tính hỗn tạp của tất cả các cấu hình đích có thể dẫn tới một cấu trúc mã nguồn phức tạp và lượng khổng lồ các mẫu triển khai khác nhau.

6.1.3 Sự tiến hoá của quá trình tạo tác qua vòng đời của nó

Mỗi trạng thái của quá trình phát triển biểu diễn bởi mức độ mô tả kỹ lưỡng so với đặc điểm cuối cùng của toàn hệ thống. Bắt đầu vòng đời, độ chính xác còn thấp và chủ yếu là sự hình dung. Cuối cùng, sự trình bày đạt cấp độ rõ ràng chính xác cao và tất cả được đặc tả đến từng chi tiết. Tuy nhiên, chúng cần có mức độ chi tiết thích hợp có thể theo dấu được đối các thành phần khác. Tiến hành sớm sự phân tích khả năng theo dõi được chi tiết và phân tích độ ổn định trong vòng đời (khi mà độ chính xác còn thấp và có nhiều thay đổi thường xuyên) có thể tiết kiệm được vốn đầu tư. Khi quá trình phát triển tiếp diễn, với kiến trúc ổn định, duy trì mối quan hệ liên kết theo dõi được giữa các tập mẫu rất đáng chú ý.

Mỗi một giai đoạn của quá trình phát triển tập trung vào một tập mẫu riêng biệt. Cuối mỗi giai đoạn, toàn bộ hệ thống sẽ tiến triển trên tất cả các tập, như hình 6.3.



Hình 6-3 : *Vòng đời tiến hoá của tập tạo tác.*

Giai đoạn khởi đầu thường tập trung chủ yếu vào các yêu cầu cấp bách, thứ hai chú trọng vào quan điểm bước đầu về triển khai, ít tập trung vào việc cài đặt ngoại trừ lựa chọn ngôn ngữ và thành phần thương mại, tập trung cao độ vào thiết kế kiến trúc nhưng không phải là chi tiết thiết kế.

Giai đoạn phát sinh, các yêu cầu được phân tích kỹ hơn, thiết kế chi tiết hơn, các vấn đề của cài đặt và triển khai như là việc cân bằng sự thực hiện giữa viễn cảnh cơ bản và phân tích sản xuất / đi mua cũng tiến triển thêm một bước. Các tác vụ của giai đoạn phát sinh gồm cả việc sinh ra một biểu mẫu có thể thi hành được. Nguyên mẫu này gồm một tập con của quá trình phát triển đối với cả bốn tập, nó đánh giá chính xác liệu giao diện và sự hợp tác giữa các bộ phận có phù hợp và đáp ứng được yêu cầu chính của hệ thống hiện tại và viễn cảnh tương lai. Mặc dù nói chung có một sự thông hiểu rộng rãi giữa giao diện các thành phần, nhưng các thành phần chuyên dụng theo đơn đặt hàng lại không được tiến hành kỹ lưỡng (Các thành phần thương mại hay là các bộ phận có sẵn thường được trau truốt kỹ). Một phần của cả bốn tập phải được tiến hóa đến một cấp độ hoàn thiện trước khi vạch ranh giới kiến trúc được thiết lập. Quá trình tiến hoá này đòi hỏi ước định đủ sự tạo tác các tập thiết kế, tập cài đặt, tập triển khai, trái với cách làm đáng phê phán dùng tập yêu cầu để dự đoán rằng dự án có thể tiếp diễn trong tầm kiểm soát với những nguy hiểm được biết trước.

Điểm mấu chốt của giai đoạn xây dựng là thiết kế và cài đặt. Điểm chú trọng đầu tiên của giai đoạn này là chiều sâu (mức độ kỹ lưỡng) của tạo tác thiết kế. Sau đó, cần nhìn nhận mô hình thiết kế dưới dạng mã nguồn và các thành phần riêng lẻ có thể kiểm tra độc lập. Giai đoạn này cần hoàn thiện dần các tập yêu cầu, thiết kế, cài đặt. Các công việc trọng yếu của tập triển khai cũng phải được hoàn thành, ít nhất cũng để kiểm tra một vài mẫu của hệ thống chương trình, như cách làm phân phát bản alpha và beta thử nghiệm.

Giai đoạn chuyển giao tập trung vào việc hoàn thành tính nhất quán và hoàn thiện của tập triển khai trong mối quan hệ với các tập khác. Những thiếu sót còn lại được giải quyết, thông tin phản hồi từ bản alpha, beta và các kết quả thử nghiệm hệ thống được kết hợp chặt chẽ.

Quá trình phát triển diễn tiến, mỗi bộ phận đều tiến hóa đến mức chi tiết hơn. Khi hoàn thành hệ thống, cả bốn tập được trau truốt tinh vi và nhất quán với mỗi tập khác. Trái với cách làm truyền thống, bạn không vạch rõ yêu cầu, sau đó làm thiết kế, Thay vì đó bạn thúc đẩy toàn bộ hệ thống; quyết định về triển khai có thể ảnh hưởng đến yêu cầu, chứ không chỉ là một chiều ngược lại. Điểm nhấn mạnh ở đây là phá bỏ đường ranh giới truyền thống, mà trong đó một tập phải tiến hành trước tập khác. Trái lại, một trạng thái của toàn bộ hệ thống tiến triển lên một trạng thái tỉ mỉ hơn, kèm theo sự tiến hóa đối với mỗi phần cấu thành của bộ phận. Trong giai đoạn chuyển tiếp, duy trì tính theo dấu được giữa tập yêu cầu và tập triển khai là cực kỳ quan trọng. Tập yêu cầu ngày càng tiến triển và cuối cùng mô tả chính xác tiêu chuẩn chấp nhận của nhà đầu tư, tập triển khai thể hiện ở sản phẩm người dùng cuối thực tế. Vì vậy, trong suốt pha chuyển tiếp, tính hoàn chỉnh và nhất quán giữa hai tập này là rất quan trọng.

Tính theo dõi được giữa các tập khác chỉ cần thiết để thực hiện mục tiêu công nghệ hay là hoạt động quản lý.

6.1.4.Tạo tác kiểm tra

Kiểm tra phần mềm truyền thống cũng tiếp cận theo cách dựa trên tài liệu tương tự như cách áp dụng đối với công việc phát triển phần mềm. Nhóm phát triển xây dựng tài liệu nêu rõ yêu cầu, tài liệu thiết kế ở cấp cao nhất, và tài liệu thiết kế chi tiết trước khi xây dựng bất kỳ một file nguồn hay file thực thi nào. Tương tự, nhóm thử nghiệm xây dựng kế hoạch kiểm tra hệ thống , tài liệu ghi thủ tục kiểm tra hệ thống, tài liệu về kế hoạch kết hợp các bước kiểm tra , kế hoạch kiểm tra từng đơn vị bộ phận, tài liệu ghi chép thủ tục kiểm tra một bộ phận trước khi xây dựng bất cứ một công cụ kiểm tra nào. Cách tiếp cận theo hướng dựa vào tài liệu gây ra cho công việc kiểm tra vấn đề tương tự như vấn đề gặp phải khi phát triển phần mềm dựa trên tài liệu.

Một nguyên lý tách bạch của một tiến trình hiện đại là chỉ sử dụng duy nhất một tập hợp, ký pháp, và mẫu dùng cho việc kiểm tra và nó cũng được dùng cho phát triển sản phẩm. Thực chất, chúng ta quan niệm rằng bộ phận cấu thành nên hoạt động kiểm tra cũng là một bộ phận yêu cầu tất yếu của sản phẩm cuối cùng. Do đó, chúng ta phải thêm một số quy tắc kỹ thuật sau vào quy trình .

- Tạo tác kiểm tra phải được tiến hành đồng thời với sản phẩm từ lúc khởi đầu cho đến khi triển khai. Vậy thì, kiểm tra là một hoạt động đầy đủ trong suốt vòng đời, không phải là một công việc sau cùng của vòng đời.
- Tạo tác kiểm tra được truyền đạt, kiến tạo, và phát triển trong cùng một tập mẫu với phát triển sản phẩm.
- Tạo tác kiểm tra tiến hành dưới dạng có thể lập trình được và có thể lặp lại được (giống như chương trình phần mềm).
- Tạo tác kiểm tra được ghi thành tài liệu giống như đối với sản phẩm.
- Nhà phát triển tạo tác kiểm tra cũng sử dụng công cụ , kỹ thuật, khóa đào tạo như người kỹ sư phát triển sản phẩm phần mềm.

Quy luật này cho phép đưa ra mức đồng nhất hoá đáng kể qua việc phân luồng công việc (chương 8). Tất cả mọi người làm việc với các ký hiệu, và kỹ thuật của bốn tập dùng cho tạo tác công trình, không dùng trình tự thiết kế và tài liệu kiểm tra riêng lẻ. Trao đổi thông tin giữa mọi người, quan điểm nhà đầu tư, và phân tích kỹ thuật có thể được thi hành với ít định dạng hơn, ít ký hiệu đặc biệt hơn, giảm thiểu sự nhập nhằng, hiệu quả cao hơn.

Thử nghiệm chỉ là một phần của công việc đánh giá. Những khía cạnh khác như là kiểm tra tính kỹ lưỡng, phân tích, trình diễn. Thử nghiệm đề cập đến sự đánh giá tính chính xác qua kết quả thực thi của bộ phận triển khai. Kiểm tra thành công nếu đem so sánh kết quả lý thuyết

với kết quả thực tế nằm trong một cấp độ chính xác quy định trước. Thử nghiệm chính là một dạng đánh giá được tự động hoá.

Mặc dù tập hợp mẫu kiểm tra có những đặc trưng riêng tuỳ vào từng dự án, tuy nhiên ví dụ sau cho ta thấy mối liên quan giữa mẫu kiểm tra và những tập mẫu khác. Xem xét một dự án xử lý thông tin địa chấn phục vụ cho mục đích thăm dò dầu. Hệ thống này có ba hệ thống con cơ bản : (1) bộ phận cảm biến dò tìm thông tin địa chấn thô trong thời gian thực và chuyển dữ liệu này cho (2) bộ phận thao tác kỹ thuật chuyển dữ liệu thô sang cấu trúc cơ sở dữ liệu và quản lý nhưng truy vấn tới CSDL này (3) bộ phận hiển thị cho phép người vận hành máy trạm khảo sát dữ liệu địa chấn dưới dạng con người có thể đọc được. Hệ thống như thế có thể có những tạo tác kiểm tra như sau :

- Tập điều hành : Những đặc tính chi tiết và miêu tả chi tiết đưa ra tiêu chuẩn tương đối và mang tính chủ quan, những kết quả mốc trung gian. Mẫu này là kế hoạch kiểm tra và kết quả kiểm tra được thoả thuận giữa những thành viên của nhóm dự án. Trật tự thay đổi phần mềm có thể dựa vào kết quả kiểm tra (sai sót, thay đổi có thể kiểm tra được, sự nhập nhằng giữa các yêu cầu, sự làm nổi bật) và tiêu chuẩn kết thúc liên quan đến sự thay đổi đường ranh giới một cách trìu tượng.
- Tập yêu cầu : Cấp độ sử dụng hệ thống miêu tả khái niệm quá trình hoạt động và bộ miêu tả kiểm tra chấp nhận, gồm cả hành vi mong đợi của hệ thống và thuộc tính chất lượng của hệ thống. Tập yêu cầu tổng thể là một mẫu kiểm tra vì nó là cơ sở cho tất cả các hoạt động đánh giá trong vòng đời.
- Tập thiết kế : Mô hình kiểm tra cho những bộ phận không phân phối được cần phải kiểm tra vạch ranh giới sản phẩm được lấy ra từ tập thiết kế. Những thành phần này gồm cả mẫu tập thiết kế, như là sự mô phỏng hiện tượng địa chấn để tạo ra dữ liệu cảm biến hiện thực; một thao tác viên ảo có thể không cần có mặt, sau khi kiểm tra; bộ công cụ riêng để ban đầu trình diễn cách sử dụng tài nguyên ; tốc độ kinh doanh và thời gian đáp ứng; dùng bộ kiểm tra đã được đóng gói hay là dùng phần kiểm tra riêng lẻ.
- Tập cài đặt : Mã nguồn có ghi chú mô tả thành phần kiểm tra và phần điều khiển kiểm tra cung cấp tính năng tương đương với thủ tục kiểm tra hay là kịch bản kiểm tra. Những file nguồn này có thể kèm theo cả tệp dữ liệu mà con người có thể đọc được biểu diễn tập định nghĩa dữ liệu tĩnh mà có thể kiểm tra chính xác file nguồn.Tệp kết quả từ bộ phận điều khiển kiểm tra đưa ra chức năng tương đương với báo cáo kiểm tra.
- Tập triển khai : Phiên bản thi hành được của bộ phận kiểm tra, điều khiển kiểm tra, và các file dữ liệu kèm theo.

Với mọi sự tháo gỡ, tất cả tạo tác kiểm tra và tạo tác sản phẩm được duy trì dùng cùng một sự nhận dạng vạch ranh giới. Chúng được tạo ra, thay đổi, và teo đi thành một đơn vị nhất quán. Bởi vì tạo tác kiểm tra đạt được nhờ sử dụng cùng ký hiệu, phương pháp, và công cụ, cách tiếp cận đến kiểm tra cũng phải nhất quán với thiết kế và phát triển. Cách tiếp cận này buộc sự tiến triển tạo tác kiểm tra phải được duy trì để cho sự kiểm tra ngược trở lại có thể được tự động hoá.

6.2 Tạo tác điều hành

Tập điều hành gồm một số mẫu thể hiện được kết quả trung gian, thông tin phụ thuộc cần thiết để đưa ra tư liệu về sở hữu sản phẩm / phương pháp sản xuất, bảo dưỡng sản phẩn, cải tiển sản phẩm, cải tiến phương pháp. Tạo tác được trình bày tóm tắt sau đây và được thảo luận sâu hơn ở những chương tiếp theo, khi mà việc quản lý luồng phân chia công việc và các hoạt động khác được trau truốt. Sau đây dùng thuật ngữ tài liệu để mô tả tạo tác, ngụ ý rằng dữ liệu có thể được ghi ra giấy trắng mực đen hẳn hoi. Trong nhiều trường hợp, dữ liệu có thể được xử lý và trao đổi chỉ bằng phương tiện điện tử.

Tác vụ kinh doanh

Tạo tác công việc kinh doanh đưa ra thông tin cần thiết để dự tính xem liệu một dự án có tiềm năng kinh tế không. Cụ thể là dự đoán thu nhập, chi phí ước tính và kế hoạch quản lý, dữ liệu sao lưu dự phòng những sự nguy hiểm và thực tế mà dự án sẽ phải đối mặt. Đối với những hợp đồng mua bán lớn, tác vụ kinh doanh phải đưa ra những đề xuất với một lô dữ liệu kèm theo. ở mức độ tiếp cận một sản phẩm thương mại, nó được thi hành với một kế hoạch ngắn gọn với một bảng tính kèm theo. Mục đích chính là chuyển những dữ liệu trực quan sang lĩnh vực kinh tế để một tổ chức có thể đánh giá chính xác ROI. Sự dự báo tài chính cũng được tiến hoá, cập nhật những dự báo ngày càng chính xác theo tiến triển của vòng đời. Hình 6-4 đưa ra một phác thảo của tác vụ kinh doanh.

Khung cảnh (Lĩnh vực, Thị trường, Phạm vi)
Tiếp cận kỹ thuật
Kế hoạch đạt được những đặc điểm
Kế hoạch đạt được chất lượng
Rủi ro kỹ thuật và trả giá về công nghệ
Tiếp cận theo hướng điều hành
Lịch trình và đánh giá rủi ro lịch trình
Dự tính mức độ thành công một cách khách quan
Phụ lục tiến hóa
Dự báo tài chính
Dự đoán chi phí
Dự đoán thu nhập
Cơ sở của dự đoán

Hình 6-4 *Phác thảo tác vụ kinh doanh điển hình*

Kế hoạch phát triển phần mềm

Kế hoạch phát triển phần mềm (SDP) soạn thảo tỉ mỉ sườn của tiến trình thành một kế hoạch chi tiết. Tài liệu vạch rõ tiến trình dự án. Nó phải tuân theo bản hợp đồng, tuân theo chuẩn của công ty, tiến hoá cùng với thiết kế và yêu cầu, được sử dụng đồng nhất trong các tổ chức con cùng tham gia phát triển phần mềm. Hai thông số quan trọng của một kế hoạch phần mềm hữu ích là cập nhật định kỳ (không ứ đọng công việc) và sự thông hiểu và đồng tình của giám đốc với người thi hành là đều như nhau. Hình 6-5 vạch ra một khung của kế hoạch phát triển phần mềm.

I. **Khung cảnh (phạm vi, mục đích)**

II. **Quy trình phát triển phần mềm**

- A. Dự án nguyên thuỷ
  1. Giai đoạn vòng đời
  2. Tạo tác
  3. Luồng công việc
  4. Điểm kiểm soát
- B. Các điểm chính về phạm vi và nội dung
- C. Thủ tục cải tiến quy trình

III. **Môi trường phát triển phần mềm**

- A. Tự động hoá quy trình (cấu hình nguồn phần cứng và phần mềm)
- B. Thủ tục định vị tài nguyên (phân chia giữa các tổ chức, truy cập bảo mật)

IV. **Điều hành thay dôidr phần mềm**

- A. Kế hoạch và thủ tục bảng điều khiển cấu hình
- B. Thủ tục và định nghĩa trật tự thay đổi phần mềm
- C. Thủ tục và định nghĩa ranh giới cấu hình

V. **Đánh giá phần mềm**

- A. Chức năng báo cáo và tập hợp tiêu chuẩn
- B. Chức năng đánh giá rủi ro (định nghĩa rủi ro, theo dõi, và giải pháp)
- C. Kế hoạch đánh giá hiện trạng

D. Kế hoạch kiểm tra cấp nhận

VI. **Thủ tục và tiêu chuẩn**

A. Thủ tục và tiêu chuẩn cho tạo tác kỹ thuật

VII. **Phụ lục tiến hoá**

A. Các điểm nhỏ về phạm vi và nội dung

B. Nguồn nhân lực (tổ chức, kế hoạch lập đội ngũ, kế hoạch đào tạo)

Hình 6-5 : *Đề cương một kế hoạch phát triển phần mềm điển hình*

Cấu trúc phân chia công việc

Cấu trúc phân chia công việc (WBS) là phương tiện để dự thảo chi phí. Để quản lý và điều hành tài chính của dự án, giám đốc dự án phần mềm cần thấu hiểu được những chi phí của dự án và sự phát sinh của chúng. Cấu trúc của trách nhiệm giải trình chi phí là một mối ràng buộc không thể thiếu đối với kế hoạch dự án. Bài học từ những dự án không thành công cho thấy nếu WBS có cấu trúc không hợp lý, nó có thể dẫn quá trình thiết kế và cấu trúc sản phẩm đi chệch hướng. Giám đốc dự án không cần phải vạch rõ cấp độ thấp hơn của WBS (theo đó định nghĩa ranh giới cụ thể của việc giải trình) cho đến khi đã đạt được mức độ ổn định của cấu trúc sản phẩm. Một sự chia nhỏ chức năng trong WBS dẫn đến sự phân tách chức năng trong phần mềm. Khái niệm về một WBS tiến hoá sẽ được khảo sát kỹ hơn trong chương 10.

Cơ sở dữ liệu trật tự thay đổi phần mềm

Kiểm soát thay đổi là một yếu tố nguyên mẫu cơ bản của quá trình phát triển lặp. Với khả năng thay đổi thoải mái hơn, một dự án có thể lặp hiệu quả hơn. Sự mềm dẻo này mang lại nhiều nội dung, chất lượng, số lượng bước lặp hơn mà một dự án có thể đạt được trong cùng một lịch trình. Sự tự do thay đổi này thu được qua nghiên cứu tự động hoá, và ngày nay môi trường phát triển lặp đảm nhận hết gánh nặng của việc quản lý thay đổi . Điều hành tổ chức mà dựa vào kỹ thuật quản lý thay đổi thủ công đã chứng tỏ có những bất cập lớn. Kết quả là, dữ liệu quản lý thay đổi đã được nâng lên thành mẫu lớp quản lý thứ nhất được mô tả như là một CSDL thấm nhuần tư tưởng sự cần thiết phải tự động hóa. Một khi phần mềm đã đạt đến vạch ranh giới điều khiển, tất cả các thay đổi cần phải được kiểm soát và theo dõi cẩn thận. Bằng cách tự động hoá mục dữ liệu và duy trì khả năng thay đổi bản ghi trực tuyến, phần lớn công việc quản lý thay đổi và hoạt động báo cáo sẽ được tự động hoá. Trật tự thay đổi phần mềm được thảo luận kỹ hơn ở chương 12.

Nội dung lặp
Mục đích có thể xác định
Tiêu chuẩn ước đoán (đánh giá)
Tiếp cận theo đà
Kế hoạch giải trình (Demonstration planing**)**
Lịch trình hoạt động
Trách nhiệm nhóm
Viễn cảnh thực thi (trình diễn cách sử dụng)
Thủ tục trình diễn
Theo dấu được đối với tầm nhìn và tác vụ kinh doanh

Hình 6-6 : *Phác thảo đặc tả phát hành thông thường chi tiết điển hình*

Sự tách riêng các chi tiết

Phạm vi, kế hoạch, tiêu chuẩn đánh giá chủ quan cho mỗi vạch ranh giới tháo gỡ xuất phát từ sự trình bày trực tiếp hoặc là từ rất nhiều nguồn khác (phân tích việc chế tạo / đi mua, mối quan tâm về sự mạo hiểm, nghiên cứu kiến trúc, làm thử, ràng buộc thi hành, ngưỡng của chất lượng). Mẫu này tiến hoá cùng với cả quá trình, đạt được độ chính xác cao hơn khi vòng đời tiến triển và đã hiểu rõ yêu cầu. Hình 6.6 phác thảo một số nét chung khi tách riêng các chi tiết.

Có hai dạng yêu cầu quan trọng. Thứ nhất là dạng phát biểu trực tiếp (yêu cầu của người dùng), có trong bản hợp đồng giữa nhóm phát triển và người mua. Thông tin này cũng cần được hoàn thiện dần cùng với vòng đời, nhưng thường là rất chậm. Chúng cần được nêu lên ở dạng con người có thể hiểu được (dạng phi thể thức, có thể có cả văn bản, mô hình, cách dùng, bảng tính, hay là các dạng khác). Mẫu cách sử dụng trong văn cảnh phát biểu trực tiếp hướng dẫn cách thao tác ở dạng người sử dụng / người mua có thể hiểu được.

Tiêu chuẩn ước đoán, dạng yêu cầu thứ hai chứa trong sự tách riêng chi tiết, là một hình ảnh tức thời của mục tiêu trong một giai đoạn nào đó của vòng đời.Tiêu chuẩn ước đoán trong sự tách riêng chi tiết được coi là tạo tác điều hành chứ không phải là tập yêu cầu. Chúng nhận được từ dạng phát biểu trực tiếp và rất nhiều nguồn khác (phân tích việc sản xuất / đi mua, phân tích rủi ro, nghiên cứu kiến trúc, làm thử, ràng buộc thi hành, ngưỡng của chất lượng). Yêu cầu hướng điều khiển được diễn giải qua cách dùng, cách nhận thức, chú thích cách sử dụng, trình bày đề mục có cấu trúc.

Yêu cầu hệ thống (liên quan tới người sử dụng / người mua) thu được từ phát biểu trực tiếp. Yêu cầu ở mức độ chi tiết tổ chức như một tiến trình (tổ chức ở dạng lặp chứ không phải dạng gồm nhiều bộ phận cấu thành) ở dạng tiêu chuẩn ước đoán (thu được từ tập các cách dùng thông thường và theo cách trình bày chủ quan . Vì vậy, yêu cầu ở mức độ chi tiết có thể tiến hoá bằng cách tóm tắt những ví dụ thuộc quan niệm sau đây thuộc một dự án lớn:

1. Bước lặp khởi đầu. Thông thường, 10 đến 20 tiêu chuẩn ước đoán thu hút vấn đề điều khiển liên quan đến trường hợp dùng cá biệt gây ảnh hưởng lên sự lựa chọn kiến trúc và toàn bộ tác vụ kinh doanh.
2. Bước lặp khảo sát kỹ lưỡng. Những tiêu chuẩn ước đoán này (khoảng 50), khi phản bác lại những kiến trúc khác, chứng minh rằng trường hợp dùng cá biệt và yêu cầu cá biệt của phát biểu trực tiếp có thể được thỏa mãn với ít sự rủi ro nhất.
3. Bước lặp xây dựng. Những tiêu chuẩn ước đoán này (khoảng hàng trăm), liên quan đến một tập các ý nghĩa theo cách làm thông thường trong từng trường hợp, khi hoàn thành, các bộ phận con cấu thành sản phẩm được chuyển cho bộ phận kiểm tra dưới dạng bản alpha hay beta.
4. Bước lặp chuyển giao. Hoàn thành hướng dẫn sử dụng và các tiêu chuẩn ước đoán liên quan (khoảng hàng ngàn) kiến tạo nên tiêu chuẩn kiểm tra đạt yêu cầu liên quan đến việc triển khai một phiên bản đi vào hoạt động.

Tiến trình này tiến hoá tự nhiên và được kết hợp lỏng lẻo với sự tiến hoá của thiết kế và kiến trúc thực tế. Cuối cùng, 100% tính theo dõi được là rất quan trọng, nhưng những hoạt động trung gian và những sự kiện quan trọng có ít liên quan đến độ ổn định và sự hoàn thiện hơn là chúng được sử dụng trong cách tiếp cận truyền thống đến phát triển phần mềm . Mỗi tiêu chuẩn ước lượng của một bước lặp được loại ra khi đã đạt đến mục tiêu; chúng là dạng tạo tác tạm thời. Một phiên bản tốt hơn luôn được tạo ra tại mỗi giai đoạn, vì vậy có nhiều nội dung được duy trì và bài học kinh nghiệm rút ra sau trong mỗi tiêu chuẩn tiến hoá tiếp theo. Mẫu tách lớp chi tiết và những tiêu chuẩn tiến hoá kế thừa nó giành mối quan tâm đầu tiên và cao nhất đến vấn đề phòng tránh rủi ro.

Nội dung
Nội dung tóm tắt đợt phát hành
Thể loại phát hành
Thông tin thêm
Những ràng buộc cụ thể hay là hạn chế
Kết quả đánh giá
Chứng minh đạt yêu cầu
Kế hoạch tiếp theo đối với không đạt yêu cầu
Giới thiệu phiên bản tiếp theo
Những vấn đề nổi bật
Khoản mục hành vi
Bài học kinh nghiệm

Hình 6-7 : *Phác thảo bản mô tả phát hành thông thường*

Mô tả việc phát hành

Tài liệu mô tả việc phát hành trình bày kết quả của mỗi lần phát hành, bao gồm cả hiệu năng so với mỗi tiêu chuẩn ước đoán tương ứng với đặc điểm chi tiết của lần phát hành đó. Ranh giới phát hành cũng cần kèm theo với tài liệu phát hành, trình bày tiêu chuẩn đánh giá cấu hình đó và đưa ra chứng minh (bằng cách trình diễn, thử nghiệm, kiểm tra, phân tích) rằng mỗi tiêu chuẩn đã được giải quyết theo cách chấp nhận được. Tài liệu này nên kèm theo cả bản tóm tắt số lượng các chỉ tiêu đánh giá chất lượng bằng những thuật ngữ liên quan thuần tuý (so sánh với phiên bản trước). Kết quả từ sự rút kinh nghiệm qua mỗi phiên bản cũng nên ghi ở đây, gồm có những vấn đề nổi bật, giới thiệu phương pháp và cải tiến sản phẩm, cân bằng các tiêu chuẩn đánh giá, bám sát quá trình diễn biến, và những thông tin tương tự. Hình 6-7 đưa ra những nét chính của một bản mô tả phát hành.

Đánh giá hiện trạng

Đánh giá hiện trạng đưa ra hình ảnh tức thời của thực tế dự án, gồm có phân tích rủi ro của giám đốc dự án phần mềm, chỉ số chất lượng, chỉ số điều hành. Mặc dù kỳ hạn có thể thay đổi, nhiệm vụ này bắt buộc phải được thực hiện. Mục tiêu cao nhất của một quy trình điều hành tốt là phải đảm bảo rằng mong muốn của tất cả các cổ đông (người đấu thầu, khách hàng, người sử dụng, nhà thầu phụ) được đồng bộ và nhất quán. Tài liệu đánh giá hiện trạng định kỳ đưa ra một cách thức hiệu quả để điều chỉnh mong muốn của mọi người khi trải qua vòng đời; để định địa chỉ, trao đổi thông tin, giải quyết vấn đề điều hành, vấn đề kỹ thuật, mức độ rủi ro, để nhìn lại lịch sử tiến triển dự án. Chúng là nhịp đập trái tim định kỳ để duy trì sự tập trung. Mục 9-3 thảo luận kỹ hơn về đề tài đánh giá thực trạng.

Thông thường một đánh giá hiện trạng gồm cả vấn đề xem xét về tài nguyên, đội ngũ nhân viên, thông tin tài chính (chi phí và thu nhập), 10 khả năng rủi ro cao nhất, tiến bộ kỹ thuật, các cột mốc chính của kế hoạch và kết quả, phạm vi của dự án và sản phẩm, chỉ mục hành vi, và đà tiến. Duy trì hệ thông tin mở với dữ liệu đích xuất phát trực tiếp từ sự tiến triển công việc và sự tiến hóa cấu hình sản phẩm là đòi hỏi bắt buộc đối với mọi dự án.

Môi trường

Một yếu tố quan trọng của cách tiếp cận hiện đại là đặt môi trường duy trì và phát triển như là một yếu tố hàng đầu của quy trình. Một môi trường phát triển mạnh mẽ được tích hợp phải trợ giúp tự động hoá tiến trình phát triển. Môi trường đó có thể có bộ phận điều khiển yêu cầu, mô hình trực quan, tự động hoá văn bản, công cụ lập trình hướng mục tiêu, tự động kiểm tra ngược, bộ quản lý các thay đổi được tích hợp và duy trì, theo dấu các sai sót hay là các điểm đặc trưng. Một điều thường thấy ở các dự án phần mềm thành công là phải thuê những nhân công giỏi và cung áp phương tiện tốt cho họ hoàn thành công việc. Tự động hoá trong tiến trình phát triển phần mềm mang lại chất lượng, khả năng dự toán chi phí và quản lý lịch trình, và toàn bộ quá trình sản xuất chỉ phải sử dụng một đội ngũ nhỏ hơn. Bằng cách cho phép người thiết kế di chuyển nhanh chóng giữa những mẫu phát

triển, và dễ dàng theo dõi các mẫu được cập nhật, bộ công cụ tích hợp đóng vai trò ngày quan trọng trong quá trình phát triển lặp và đi lên.

Triển khai

Một tài liệu triển khai có thể có rất nhiều dạng. Tuỳ thuộc vào từng dự án, nó có thể gồm mốt số tập tài liệu con để chuyển sản phẩm sang trạng thái có thể thi hành được. Trong nỗ lực thực hiện những bản hợp đồng lớn mà hệ thống được giao cho những tổ chức bảo trì riêng biệt, tạo tác triển khai có thể gồm sách hướng dẫn thao tác hệ thống máy tính, hướng dẫn cài đặt, kế hoạch và thủ tục giảm bớt (từ một hệ thống thừa kế), điều tra địa điểm,vv…Đối với những sản phẩm thương mại, tạo tác triển khai có thể có cả kế hoạch tiếp thị, bộ công cụ giới thiệu sản phẩm, và cả các khoá đào tạo.

△ Phiên bản không chính thức

▲ Ranh giới được điều khiển

Khởi đầu	Phát sinh		Xây dựng		Chuyển giao	
Lặp 1	Lặp 2	Lặp 3	Lặp 4	Lặp 5	Lặp 6	Lặp 7

Tập điều hành

1. Cấu trúc phân chia công việc
2. Tác vụ kinh doanh
3. Tháo gỡ đặc tả
4. Kế hoạch phát triển phần mềm
5. Mô tả việc phát hành
6. Đánh giá hiện trạng
7. Dữ liệu trật tự thay đổi phần mềm
8. Tài liệu triển khai
9. Môi trường

Tập yêu cầu

1. Tài liệu trực quan
2. Mô hình yêu cầu

Tập thiết kế

1. Mô hình thiết kế
2. Mô hình kiểm tra
3. Mô tả kiến trúc

Tập cài đặt

1. Ranh giới mã nguồn
2. File kèm theo khi biên dịch

3. Thành phần có thể thực thi

Tập triển khai

1. Ranh giới sản phẩm thực thi được tích hợp
2. File kèm theo khi thực thi
3. Tài liệu hưỡng dẫn người sử dụng

Hình 6-8 : *Trật tự tạo tác qua vòng đời điển hình*

Trật tự các tạo tác quản lý

Trong mỗi giai đoạn của vòng đời, mẫu mới được sinh ra và những mẫu đã tạo tác trước đó phải được cập nhật để phối hợp các bài học kinh nghiệm và để đi sâu và rộng hơn vào giải pháp. Một số mẫu được cập nhật ở những mốc chính, một số khác thì ở mỗi giai đoạn nhỏ.Hinh 6-8 minh họa một trật tự điển hình của các mẫu trải qua các giai đoạn của vòng đời.

6.3 .Tạo tác kỹ thuật

Phần lớn các tạo tác kỹ thuật thu được trong những ký hiệu kỹ thuật chặt chẽ như là UML, ngôn ngữ lập trình, hay là mã máy có thể thi hành được. Bởi vì quyển sách này đứng trên quan điểm quản lý, nên không chú trọng vào những mẫu này. Tuy nhiên, ba tạo tác kỹ thuật chính là để có cái nhìn tổng quan hơn, và chúng đáng được xem xét kỹ hơn.

Tài liệu quan sát

Tài liệu quan sát đưa ra một cái nhìn tổng thể về hiện trạng hệ thống đang được phát triển và cung cấp thông tin về thoả thuận giữa nhà đầu tư và nhóm phát triển. Tùy vào dự án có thể là cỡ lớn phát triển cho chuẩn quân sự (tài liệu có thể có đến 300 trang đặc tả hệ thống) hay vào loại nhỏ, sản phẩm thương mại tự đầu tư (tổng quan về nó chỉ khoảng 2 trang giấy), mọi dự án đều cần phải nắm bắt được mong muốn của các cổ đông. Tầm nhìn của dự án cũng phải thay đổi đực khi mà sự am hiểu tường tận về yêu cầu,về kiến trúc, kỹ thuật ngày càng tiến bộ. Một tài liệu quan sát tốt thường thay đổi từ từ. Hình 6-9 phác thảo những nét chung nhất của một tài liệu quan sát.

Mô tả tập chức năng
A. Thứ tự và quyền ưu tiên
Thuộc tính chất lượng và phạm vi
Ràng buộc yêu cầu
B. Giao tiếp ngoài
Phụ lục tiến hoá
Cách dùng trong từng trường hợp
Tự do yêu cầu (tiềm năng thay đổi)

Hình 6-9 : *Phác thảo tài liệu trực quan điển hình*

Tài liệu quan sát được soạn trên quan điểm của người sử dụng, tập trung vào những vấn đề nhạy cảm của hệ thống và mức độ chất lượng chất nhận được. Một tài liệu quan sát cần có ít nhất hai phụ lục. Phụ lục thứ nhất mô tả khái niệm thao tác qua các cách làm thông dụng (mô hình trực quan và các tạo tác riêng lẻ). Phụ lục thứ hai mô tả những thay đổi nguy hiểm xuất phát từ sự phát biểu trực quan, hướng dẫn sửa chữa những lỗi thiết kế.

Sự trình bày trực quan nên kèm theo cả sự mô tả những vấn đề đi kèm cũng như là những khía cạnh được cân nhắc nhưng không được gộp vào. Nó cần phải chỉ ra năng lực thi hành (khối lượng, thời gian đáp ứng, độ chính xác), sơ lược cách sử dụng và sự giao tiếp liên thao tác với những thực thể nằm ngoài ranh giới hệ thống, trong trường hợp có thể áp ụng được. Sự nhìn nhận chỉ nên định nghĩa cho mức thao tác đầu tiên; chiều hương tiến hoá thích hợp cũng nên vạch ra để có một khung cảnh cho đánh giá khả năng thích ứng thiết kế. Khái niệm thi hành được gồm cả việc chỉ ra cách sử dụng và đưa ra những tình huống sử dụng có thật và không có thật. Sự trình bày cách dùng cung cấp khung cảnh động để hiểu việc cải tiến phạm vi, đánh giá tính toàn vẹn của mô hình thiết kế, để xây dựng những thủ tục kiểm tra đạt yêu cầu. Tài liệu quan sát cung cấp cơ sở thoả thuận cho những yêu cầu nhìn thấy được đối với các cổ đông.

Mô tả kiến trúc

Việc mô tả kiến trúc cung cấp cách nhìn có hệ thống về kiến trúc phần mềm đang được phát triển. Nó được trích một phần lớn từ mô hình thiết kế và bao gồm cả quan điểm về tập thiết kế, cài đặt , triển khai đủ để hiểu khả năng thi hành được những yêu cầu sẽ đạt được bằng cách nào. Độ rộng của sự mô tả kiến trúc thay đổi theo từng dự án tuỳ thuộc vào nhiều yếu tố. Kiến trúc có thể được mô tả bằng một tập con của mô hình kiến trúc hay là sự trìu tượng hoá mô hình kiến trúc cùng với một số tài liệu phụ lục, hoặc là kết hợp cả hai cách. Một ví dụ hai cách mô tả, hãy xem xét kiến trúc của cuốn sách này:

- Thể loại dùng tập con có thể nhìn nhận như là bảng mục lục. Sự mô tả kiến trúc của cuốn sách này xuất phát từ chính bản thân nó.
- Cách làm theo kiểu trìu tượng hoá bằng cách dùng một bản ghi chú bên lề (ghi chú bên lề là tài liệu súc tích của thể loại sách truyền thống được dùng như là một hướng dẫn nghiên cứu tạo ra bởi các sinh viên đại học). Dạng này là một kiểu trìu tượng hoá được xây dựng riêng rẽ và kèm theo cả các vật chất phụ thêm mà không thu được trực tiếp trong khi xây dựng sản phẩm.

Cách tiếp cận mô tả ở mục 7-2 cho phép một mô tả kiến trúc được điều chỉnh để đáp ứng yêu cầu đặc trưng của mỗi dự án. Hình 6-10 đưa ra một nét phác thảo chung cho việc mô tả kiến trúc.

Sách hướng dẫn người sử dụng phần mềm

Sách hướng dẫn người sử dụng phần mềm cung cấp cho người sử dụng tài liệu tham khảo cần thiết để tiến hành phân phối phần mềm. Mặc dù nội dung thì biến đổi tuỳ theo từng lĩnh vực ứng dụng, tài liệu hướng dẫn người dùng nên có hướng dẫn thủ tục cài đặt, hướng dẫn cách dùng, mối ràng buộc thi hành, và mô tả giao diện người dùng tối thiểu. Với những sản

phẩm phần mềm có giao diện cho người sử dụng, tài liệu hướng dẫn này cần được bắt đầu thực hiện sớm trong vòng đời bởi vì đó là một cơ chế cần thiết để trao đổi thông tin duy trì tính ổn định của một tập các yêu cầu quan trọng. Tài liệu hướng dẫn sử dụng cần được viết bởi thành viên của nhóm thử nghiệm, người có khả năng hiểu rõ triển vọng người dùng hơn là nhóm phát triển. Nếu nhóm thử nghiệm chịu trách nhiệm làm sách hướng dẫn, nó có thể được tiến hành song song với quá trình phát triển và có thể tiến hóa sớm như là triển vọng hữu hình tương ứng của một tiêu chuẩn ước đoán. Nó cũng giúp đưa ra cơ sở cần thiết đối với kế hoạch kiểm tra và cách thử nghiệm, để xây dựng các bộ kiểm tra tự động.

Tổng quan kiến trúc
Mục đích
Ràng buộc
Đặc quyền
Quan điểm về kiến trúc
Quan điểm thiết kế
Quan điểm về phương pháp
Quan điểm về các bộ phận
Quan điểm về triển khai
Sự ảnh hưởng lẫn nhau về kiên trúc
Khái niệm thao tác trong viễn cảnh chính
Khái niệm thao tác trong viễn cảnh thứ yếu
Khái niệm thao tác trong tình huống bất thường
Hiệu năng kiến trúc
Nhân tố căn bản, sự cân bằng các yếu tố, và sự minh chứng

Hình 6-10. *Đề cương về một mô tả kiến trúc điển hình*

6.4 .Tạo tác trong thực tế

Cách tiếp cận theo cách truyền thống dựa vào tài liệu lãng phí một lượng lớn thời gian để phát triển, chỉnh sửa, định dạng, cân nhắc, cập nhật, và phân phối tài liệu. Vì sao ? Có một số lý do khiến cho tài liệu trở nên rất quan trọng đối với tiến trình. Thứ nhất, không có một phương pháp kỹ thuật chính xác hay ngôn ngữ nào để đặc tả yêu cầu hay bước thiết kế. Kết quả là, tài liệu trên giấy ở dạng đặc biệt và sự trình diễn hình ảnh là dạng chuẩn. Thứ hai, ngôn ngữ truyền thống để cài đặt và triển khai thường rất khó hiểu và không có cấu trúc. Để trình bày chi tiết cấu trúc phần mềm và những hành vi cho những nhà phê bình liên quan (người thử nghiệm, bảo trì, giám đốc), cần có một định dạng dễ hiểu hơn đối với con người. Điều quan trọng nhất là, những tiến triển phần mềm cần được đánh giá đúng. Cách trình bày bằng tài liệu rõ ràng là chân thực nhưng cũng có thể đưa ra thông tin sai lệch khi chứng minh sự đi lên.

Trên một số lĩnh vực, cách tiếp cận sử dụng tài liệu đã suy thoái sau 30 năm qua trở thành một cản trở chính đối với sự tiến bộ của quy trình. Chất lượng của tài liệu trở nên quan trọng hơn là chất lượng của thông tin kỹ thuật chứa trong chúng. Chất lượng phỏng đoán

qua quan điểm của con người về miêu tả trìu tượng là một phương pháp rất chủ quan. Rất nhiều nỗ lực tập trung vào đánh giá một khía cạnh riêng của vấn đề, trong khi đó thiếu sự tập trung cần thiết cho những vấn đề tổng quát ảnh hưởng trực tiếp đến chất lượng công trình, ví dụ như hiệu năng và khả năng thích ứng.

Chu trình sản sinh tài liệu, chu trình xem xét lại, chu trình cập nhật cũng thêm vào lịch trình một bức tranh hình thức hữu hình về những tiến bộ, qua đó đưa ra những phụ thuộc lịch trình và điểm đồng bộ hóa. Ví dụ, tình huống sau đây không phải là hiếm thấy ở các dự án phòng thủ lớn : Một tháng để chuẩn bị tài liệu thiết kế, phân phát tài liệu cho khách hàng để tham khảo, đợi một tháng nhận trả lời, một tháng nữa để đáp ứng lại lời nhận xét và phối hợp các thay đổi. Với rất nhiều chu trình trao đổi tài liệu kéo dài nhiều tháng, với việc lên lịch, điều hành và đồng bộ hóa, không có gì ngạc nhiên rằng những dự án như thế có thể kéo dài tới 5 năm. Những tài liệu dài và chi tiết, nói chung hiểu được là để xúc tiến tốt hơn, đưa ra những chi tiết kỹ thuật vội vàng và tăng số lượng việc phải loại bỏ hay là những công việc phải làm lại sau này trong vòng đời.

Một cách tiếp cận hiệu quả hơn là điều chỉnh lại những nỗ lực dẫn chứng bằng tư liệu để cải thiện độ chính xác và khả năng nắm bắt nguồn thông tin và cho phép xem thông tin nguồn bằng các công cụ lựa chọn và điều hướng thông minh. Cách tiếp cận này có thể loại ra nhiều vật thải loại và việc phải làm lại không hữu ích trong quy trình và cho phép mọi người liên quan trực tiếp tới chu trình phát triển mẫu trực tuyến tiếp tục có thể xem xét cân nhắc.

Quan điểm này liên quan tới những điểm nổi bật sau :

- Người nào đó muốn xem thông tin nhưng thể hiểu được ngôn ngữ của vật tạo tác (mẫu). Nhiều nhà phê bình liên quan trong những mẫu cụ thể sẽ phản đối việc bắt buộc phải học ngôn ngữ kỹ thuật của vật tạo tác. Không hiếm khi gặp một người (giám đốc phần mềm kỳ cựu, chuyên gia bảo hiểm chất lượng lâu năm, bộ phận kiểm toán từ tổ chức điều chỉnh môi giới) sẽ phản ứng như sau : "Tôi sẽ không học UML, nhưng tôi muốn hiểu phác thảo của phần mềm này, hãy đưa cho tôi một bản mô tả riêng rẽ như là biểu đồ tiến độ cùng với văn bản để tôi có thể hiểu được". Chúng ta có nên đáp ứng một yêu cầu tương tự của một người nào đó đòi hỏi xem bản thiết kế chi tiết kỹ thuật xây dựng một toà nhà.
- Một người muốn xem xét thông tin nhưng không thể sử dụng công cụ. Không phải mọi tổ chức phát triển đều được trang bị đầy đủ, rất hiếm các cổ đông có khả năng xem mẫu công trình trực tuyến. Kết quả là, các tổ chức bắt buộc phải trao đổi tài liệu giấy. Định dạng chuẩn (UML, bảng tính, Visual Basic, C++, và Ada 95), công cụ hình dung, và Web đang nhanh chóng mang lại tính khả thi về mặt kinh tế cho các cổ đông trao đổi xử lý thông tin bằng điện tử. Tiếp cận tới mẫu là một lĩnh vực mà quá trình phát triển tối ưu phần mềm có thể bị sai hỏng nếu học thuyết về chu trình đó không được chấp nhận bởi các cổ đông.
- Mẫu kỹ thuật mà con người có thể đọc được cần sử dụng những ký hiệu chính xác mà phải hoàn chỉnh, phù hợp, và được dùng theo cách tự đưa ra dẫn chứng bằng tài

liệu. Thuật ngữ thông dụng đúng đắn nên dùng đối với tất cả các từ định danh và miêu tả. Chữ viết tắt chỉ nên dùng khi chúng đã được chấp nhận rộng rãi ở trong cách dùng thành phần. Dù đang sử dụng ngôn ngữ hay công cụ nào, không có lý do gì để viết tắt hay là mã hóa mô hình và từ định danh trong chương trình nguồn. Tiết kiệm gõ phím bằng cách viết tắt có thể làm nhẹ công việc của người làm tạo tác, nhưng sẽ gây ra lỗi cho toàn bộ vòng đời. Không cho phép làm như thế sẽ đem lại cả năng suất và chất lượng. Phần mềm chỉ được viết một lần, nhưng nó được đọc nhiều lần. Vì vậy, tính năng đọc được nên được nhấn mạnh và việc dùng ngôn từ chuẩn xác cần phải đòi hỏi ở tất cả các mẫu công trình. Cách làm này đem lại sự trình bày dễ hiểu, định dạng có thể duyệt qua (xem lại không giấy), ký hiệu chính xác hơn, và giảm tỷ lệ lỗi.

- Dẫn chứng bằng tư liệu hữu ích có thể tự hạn chế nội dung : Đó là những dẫn chứng bằng tư liệu thường được dùng. Nói chung, xây dựng mô hình tự chứng minh bằng tài liệu kỹ thuật cho phép một tổ chức phát triển có quyền chỉ làm việc trên một loại ký hiệu kỹ thuật và tránh dùng tài liệu riêng biệt để mô tả tất cả các chi tiết của mô hình, thành phần, hay là thủ tục kiểm tra.. Nếu bạn tìm ra thông tin, tài liệu đặc thù đang trở nên quá dài mà lại không được dùng, hãy loại nó ra để hoàn thành mục đích đã định. Hãy cố gắng cải thiện khả năng tự chứng minh bằng tài liệu.
- Giấy thì hữu hình; mẫu điện tử thì dễ dàng thay đổi. Một lý do khiến các cổ đông thích dùng tài liệu bằng giấy hơn vì một khi chúng đã được phát hành, chúng là hữu hình, không thay đổi, cố định. Tạo tác trực tuyến và trên nền Web thì dễ dàng sửa đổi và được xem với nhiều thái độ hoài nghi vì đặc tính bay hơi cố hữu của chúng. Mặc dù tạo tác điện tử sẽ vượt qua sự hoài nghi của các nhà đầu tư, nhưng cần phải một thời gian nữa cả thế giới mới chuyển sang cách làm này. Những đảm bảo còn lại rằng công cụ và môi trường sẽ tiến hoá để hỗ trợ việc điều hành thay đổi, hoạt động kiểm toán, chữ ký điện tử, và những tiến bộ về phần mềm nhóm làm cho việc trao đổi điện tử sẽ thay thế cách dùng giấy.

Phải nhấn mạnh rằng thông tin trong tạo tác mới là điểm chính yếu, chứ không phải là giấy mà chúng được viết trên đó. Tài liệu ngắn gọn thường hữu ích hơn tài liệu dài. Phần mềm mới là sản phẩm chính; đưa ra tài liệu chỉ là để cung cấp thêm thông tin.

Chương 7

Mẫu kiến trúc phần mềm dựa trên mô hình

Các điểm chính :

- Một kiến trúc là thiết kế hệ thống phần mềm.
- Mục tiêu cuối cùng của bước thiết kế là đưa về mức kiến cơ sở.
- Một mức kiến trúc cơ sở không phải là một tài liệu trên giấy, nó là tập hợp các thông tin trong suốt tập hợp các quá trình công nghệ.
- Kiểu kiến trúc được mô tả bằng cách khai thác các thông tin cơ sở từ những kiểu thiết kế.

Kiến trúc phần mềm là bước thiết kế trung tâm. Các vấn đề của một hệ thống phần mềm phức tạp cũng như các vấn đề của việc thiết kế một toà nhà trọc trời. Tuy nhiên kiến trúc phần mềm có một số khía cạnh bổ sung rất phức tạp. Đối lập với kiến trúc của một toà nhà lớn, những thuộc tính kĩ thuật tới hạn và các điểm đặc trưng của một hệ thống phần mềm phổ thông không thể mô tả qua các định luật vật lý bền vững.Chúng không bị chi phối bởi bất kì hình thức thông thường của toán học. Vì vậy kiến trúc phần mềm không có nghuyên tắc bất di bất dịch nào. Có rất nhiều mẹo và các đường lối chỉ đạo mập mờ, nhưng cơ sở để đánh giá phụ thuộc chặt chẽ vào từng tình huống cụ thể. Không theo bất kì học thuyết nào, kiến trúc phần mềm phải dựa vào một số dạng của môi trường trong kiến trúc phần mềm đề ra. Đây là một trong các lý do chính của quá trình chuyển kiểu đến một quá trình tương tác, mà ở đó những hoạt động trong các quá trình trước làm nổi bật nên và xúc tiến sự phát triển của kiến trúc trong suốt quá trình lấy mẫu và thử.

Những chương trước chúng ta đã đưa ra rất nhiều kết luận về kiến trúc mà chưa đưa ra các định nghĩa của các thuật ngữ. Không có bất kì sự định nghĩa nào của kiến trúc được chấp nhận trong các quá trình công nghiệp. Chương này tóm tắt một số khía cạnh của kiến trúc để xây dựng một ngữ cảnh mà ở đó các tiền đề đầu tiên của quá trình kiến trúc có thể hiểu được.

Bởi vì các hệ thống phần mềm trước đây có tính năng kém hơn rất nhiều so với các hệ thống ngày nay, kiến trúc của chúng đơn giản hơn rất nhiều và chỉ yêu cầu sự trình bầy không theo qui định. Trên một máy tính đơn, hệ thống đơn chương trình, sự ánh xạ giữa các đối tượng thiết kế, các đối tượng được trình bày và các đối tượng triển khai thông thường rất tầm thường. Trong những hệ thống phần mềm phức tạp hiện nay chúng ta phải giải quyết với các máy tính đa chương trình đa người sử dụng, những mẫu ở xa và xem xét để giải thích thuận tiện của công nghệ hiện đại như là các thành phần thương mại, phương pháp lập trình hướng đối tượng, các hệ thống mở, hệ thống phân phối trong môi trường chủ khách và các ngôn ngữ hiên đại. Một mẫu là sự liên hệ các thành phần độc lập trừu tượng của một hệ thống. Sự xem xét như là một tập con của một mẫu, trừu tượng hoá góc nhìn thích ứng.

7.1. Kiến trúc: Từ góc nhìn về quản lý

Sản phẩm kĩ thuật khó thực hiện nhất của một dự án phần mềm là kiến trúc của chúng: cấu trúc bên trong, điều khiển và tương tác dữ liệu làm cho các thành phần của phần mềm có thể hoạt động như một hệ thống và người thiết kế phần mềm có thể thực hiện hiệu quả theo nhóm. Sự thiết lập một cách chính xác và nghiêm ngặt truyền thông giữa các đội là một vấn đề không bao giờ kết thúc của bất kì tổ chức nào. Từ khi các phương tiện truyền thông đại chúng bao gồm nhiều ngôn ngữ và sự biến đổi trình độ giữa các nhóm người, vấn đề truyền thông trở nên cực kì phức tạp và hầu như không thể giải quyết được. Nếu một đội muốn thành công thì thông tin giữa các đề tài ví dụ như trong kiến trúc phần mềm phải vừa chính xác và vừa rõ ràng.

Từ góc nhìn của quản lý có 3 khía cạnh khác nhau của kiến trúc:

1. Một "kiến trúc" (một khái niệm thiết kế mơ hồ) là thiết kế của một hệ thống phần mềm, đối lập lại với thiết kế một thành phần. ở đây bao gồm một danh sách hoàn chỉnh của các vật liệu. Một giải quyết hợp lý/các quyết định mua bán được giải quyết, và tất cả các lựa chọn được chi tiết hoá vì thế giá của các thành phần được chi tiết hoá vì thế giá của các thành phần đơn lẻ và giá phải trả cho sự thiết lập / lắp ráp có thể xác định một cách chắc chắn.
2. Một ranh giới kiến trúc cơ sở (một mẫu hữu hình) là một phần thông tin đi qua tập các quá trình công nghệ mẫu để thoả mãn tất cả các cổ đông mà thông tin về chức năng và chất lượng có thể đạt được qua các thông tin về tình trạng kinh doanh (giá cả, lợi nhuận, thời gian, công nghệ, nhân sự).
3. Một mô tả của kiến trúc (một bản trình bày có thể đọc được của một kiến trúc, nó là một trong những thành phần của ranh giới kiến trúc) là một tập con các thông tin đã được tổ chức rút ra từ tập các mẫu thiết kế. Nó bao gồm sự bổ sung của các kí hiệu (văn bản hoặc đồ hoạ) cần thiết để là cho các thông tin trong các mẫu được rõ ràng. Sự mô tả kiến trúc truyền thông làm sao cho các khái niệm mơ hồ được nhận ra trong các mẫu hữu hình.

Những định nghĩa trên là những lý thuyết cần thiết bởi vì kiến trúc được xác định khác nhau trên các vùng khác nhau của hệ thống. Đặc biệt số lượng các quan điểm và mức độ chi tiết có thể biến đổi rất rộng. Ví dụ kiến trúc của một bản tóm tắt thì rất đơn giản hơn kiến trúc của một bức tranh sống động, mặc dù sản phẩm có thể trình bày dưới các dạng khác nhau về sinh học. Kiến trúc của một tàu lượn thì rất đơn giản hơn kiến trúc của một máy bay phản lực mặc dù cả hai sản phẩm đều là máy bay. Tương tự kiến trúc cho một hệ thống giao thông trên không rất khác với kiến trúc của một phầm mềm phát triển nhỏ.

Tầm quan trọng của kiến trúc phần mềm và sự liên hợp gần gũi của nó với các quá trình phát triển phần mềm hiện đại có thể tóm tắt như sau:

Để đạt được kiến trúc phần mềm tiêu biểu cho mốc của dự án mà ở đó các quyết định mua/bán khó giải quyết có thể đã được giải quyết. Các sự kiện tiêu biểu cho vòng đời này tiêu biểu cho sự quá độ từ giai đoan công nghệ của một đề tài được mô tả đặc điểm bằng cách khám phá, phát minh và giải quyết một khối lượng lớn những điều không biết cho tới giai đoạn sản xuất được mô tả bằng việc quản lý một dự án phát triển khả thi.

Sự mô tả cấu trúc cung cấp một cơ sở cho sự cân bằng thương mại giữa không gian các ràng buộc và không gian các giải pháp sản suất các sản phẩm.

Kiến trúc và quá trình thâu tóm rất nhiều sự truyền tin quan trọng giữa các nhóm các tổ chức và các cổ đông.

- Kiến trúc nghèo nàn và các quá trình không chín muồi thường là các nguyên nhân dẫn đến sự thất bại của các dự án.
- Các quá trình chín muồi một bản rõ ràng các điều kiện cơ sở cần thiết và kiến trúc có thể giải thích được là những yêu cầu quan trong cho một kế hoạch có thể quản lý được.
- Sự phát triển kiến trúc và sự phát triển các quá trình là các bước khoa học tương ứng các vấn đề cần giải quyết tới các giải pháp mà không vi phạm các ràng buộc, chúng yêu cầu các sáng kiến của con người và không thể tiến hành một cách tự động.

7.2. Kiến trúc: Từ góc nhìn kĩ thuật

Mặc dù kiến trúc phần mềm đã được đề cập đến từ hơn một thập kỉ trước nhưng sự hội tụ của các thuật ngữ, các phương pháp vẫn còn thiếu. Vấn đề thảo luận sau đây mô tả tổng quan trên cơ sở của sự phát triển kiến trúc ở tập đoàn Raditional (và đặc biệt dựa trên các khái niệm của Philipe Kruchten về kiến trúc phần mềm [Kruchten, 1995].

Kiến trúc phần mềm bao gồm cấu trúc của hệ thống phần mềm (tập các phần tử và sự kết hợp của chúng vào trong các hệ thống con ngày càng lớn hơn), hành vi của chúng (sự hợp tác giữa các phần tử) và các mô hình hỗ trợ cho các phần tử này, sự hợp tác và kết hợp của chúng. Ngữ cảnh của cấu trúc kiến trúc phần mềm, hành vi và các mô hình phải bao gồm tính chức năng, thực hiện và phải hiểu được, sự trả giá kinh tế, các yêu cầu về công nghiệp và các vấn đề khác liên quan đến thẩm mĩ.

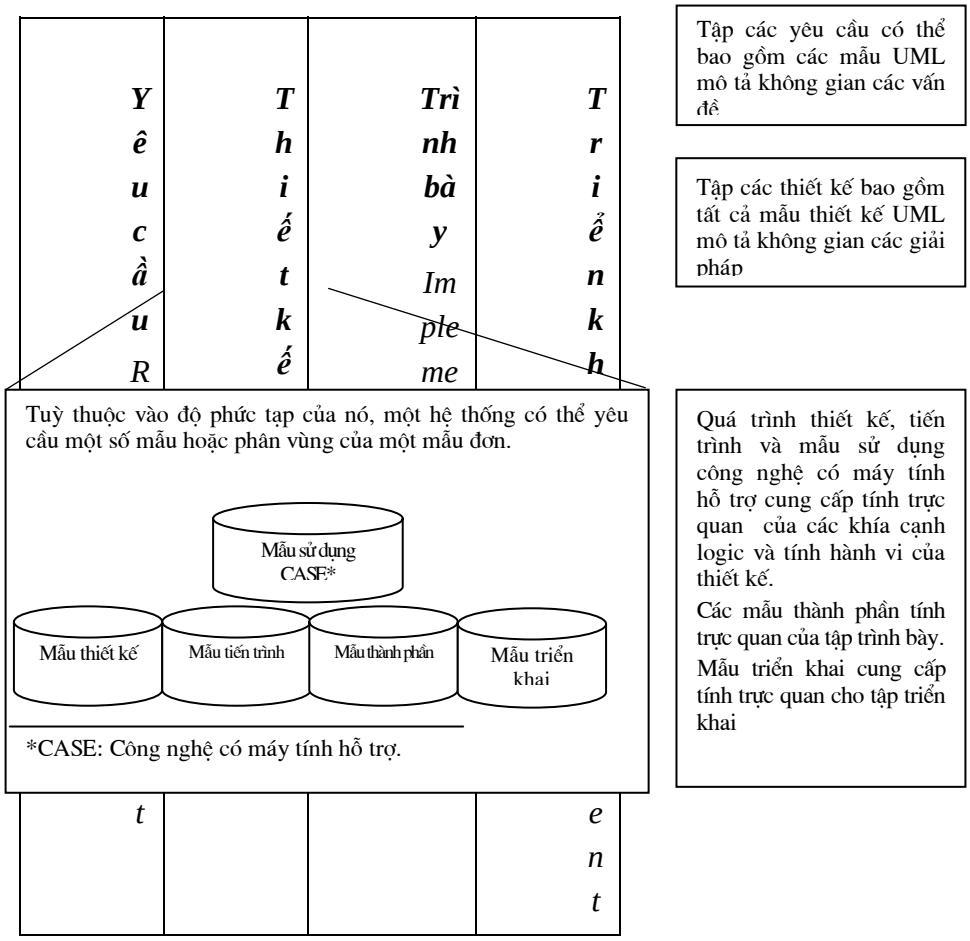
Một khung của kiến trúc được định nghĩa qua các thuật ngữ của việc nhìn nhận rằng đó là kiểu UML trong tập các thiết kế. Mẫu thiết kế bao gồm đầy đủ chiều rộng và chiều sâu của thông tin. Một hệ thống thực yêu cầu 4 sự nhìn nhận sau đây: thiết kế, tiến hành, các thành phần và triển khai. Mục đích của các điểm này như sau:

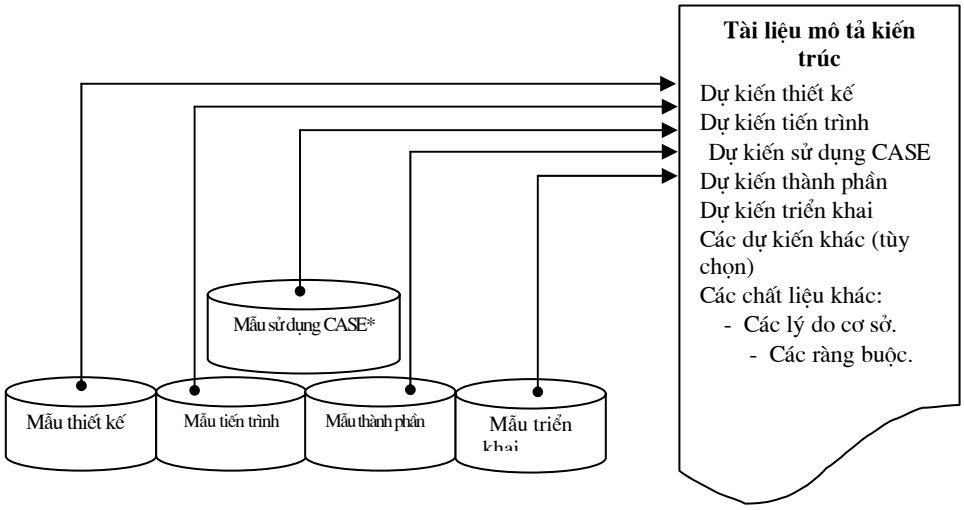
- Thiết kế: mô tả cấu trúc, tính kiến trúc và chức năng mẫu thiết kế.
- Tiến hành: mô tả sự kết hợp và móc nối quan hệ giữa các thiết kế, các thành phần và việc triển khai.
- Các thành phần: mô tả cấu trúc của tập trình bày.

- Triển khai: mô tả cấu trúc của tập triển khai.

Quan điểm thiết kế có lẽ là cần thiết trong tất cả các hệ thống, 3 quan điểm còn lại có thể bổ sung để giải quyết với các hệ thống phức tạp gần đây. Ví dụ bất cứ hệ thống nào được phân phối đều cần sự xem xét của quá trình triển khai. Hầu hết các hệ thống lớn cũng như các hệ thống bao gồm sự pha trộn giữa các thành phần tuỳ schọn và các thành phần thương mại cũng yêu cầu sự xem xét của các thành phần riêng biệt.

Hình 7.1 thống kê các mẫu của tập thiết kế bao gồm dự kiến kiến trúc và mô tả kiến trúc. Sự mô tả kiến trúc luôn mang tính điện tử nhưng nó luôn được lưu trữ do đó có thể in được như một văn bản cố kết. Mẫu công nghệ và mô tả kiến trúc được mô tả như một tập hợp của biểu đồ UML.





Hình 7: Kiến trúc, tổ chức và các dự kiến tóm tắt trong việc thiết kế mẫu

Các mẫu yêu cầu đánh dấu địa chỉ của các hành vi hệ thống qua người sử dụng,

phân tích, kiểm tra cuối cùng. Những dự kiến này được mẫu hoá sử dụng công nghệ phần mềm có máy tính hỗ trợ, sử dụng động thứ tự, sự cộng tác, đồ thị các bước và sơ đồ hoạt động.

- Dự kiến sử dụng công nghệ phần mềm có máy tính hỗ trợ (CASE) mô tả bằng cách nào mà một tới hạn của hệ thống (có ý nghĩa kiến trúc) sử dụng công nghệ CASE có thể nhận ra được bởi các phần tử của mẫu thiết kế, quá trình này được làm mẫu cố định sử dụng công dụng của biểu đồ CASE, và tính động của bất kì biểu đồ hành vi UML.

  Mẫu thiết kế địa chỉ hoá kiến trúc của hệ thống và thiết kế của các thành phần trong kiến trúc, bao gồm cấu trúc chức năng, cấu trúc phối hợp, cấu trúc trình bày và cấu trúc thực hiện của không gian các giải pháp được đề ra bởi người phát triển hệ thống. Các mô tả tĩnh được cung cấp cùng với sơ đồ có tính cấu trúc cùng với bất kì sơ đồ của sơ đồ UML (tương tác, thứ tự, đồ thị trạng thái, sơ đồ hoạt động).
- Dự kiến thiết kế mô tả ý nghĩa có tính kiến trúc các phần tử của mẫu thiết kế, địa chỉ hoá các cấu trúc cơ sở và chức năng của từng giải pháp. Nó được tạo mẫu tĩnh sử dụng biểu đồ lớp, biểu đồ đối tượng, và sử dụng động bất kì sơ đồ hành vi nào của UML.
- Dự kiến về tiến trình địa chỉ hoá sự tương tác thời gian chạy đưa ra bao gồm thực hiện kiến trúc trong mẫu triển khai, bao gồm cấu trúc mạng logic của mạng làm việc các phần mềm (vị trí tới hạn các tiến trình và mạch điều khiển), quá trình thông tin giữa các tiến trình và quản lý các bước. Dự kiến này được mô hình tĩnh sử dụng sơ đồ triển khai và sử dụng động bất kì sơ đồ hành vi nào của UML.

- Dự kiến về các thành phần mô tả ý nghĩa kiến trúc các phần tử của tập trình bày. Dự kiến này, một quan điểm trừu tượng của mẫu thiết kế địa chỉ hoá mã thực hiện phần mềm của hệ thống trên góc nhìn của người hợp nhất dự án và người phát triển, đặc biệt cùng với sự mong đợi được phát hành và quản lý cấu hình. Nó được tạo mẫu tĩnh sử dụng sơ đồ các thành phần, và sử dụng động bất kì sơ đồ hành vi nào của UML.
- Dự kiến triển khai dánh dấu phần thực hiện được của hệ thống, bao gồm vị trí của các quá trình logic đứng trên quan điểm phân phối (địa chỉ vật lý của phần mềm) tới các tài nguyên vật lý của mạng trển khai (địa chỉ vật lý của hệ thống). Nó được tạo mẫu tĩnh sử dụng sơ đồ triển khai, và sử dụng động bất kì sơ đồ hành vi nào của UML.

Sự mô tả kiến trúc thực hiện trên các dạng, các kiểu khác nhau, trong những tổ chức và miền làm việc khác nhau. Tại bất kì một thời điểm nào một kiến trúc yêu cầu một tập con của các mẫu trong tập các kĩ thuật. Sức chứa thực sự của mỗi thành phần của tập các kĩ thuật phụ thuộc từng tình huống cụ thể và có rất ít mẹo để mô tả một cách đối tượcg cái gì có tính kiến trúc và cái gì là không.

Một cách tổng quát, một ranh giới kiến trúc bao gồm:

- Các yêu cầu: Phê phán các trường hợp sử dụng, mục tiêu mức độ chất lượng của hệ thống và sự ưu tiên mối quan hệ giữa các tính năng và chất lượng
- Thiết kế: tên, thuộc tính,cấu trúc, hành vi, sự phân nhóm, mối quan hệ của lớp và các thành phần.
- Trình bày: Các thành phần nguồn của kho hàng và hoá đơn vật liệu (số lượng, tên, mục đích, giá cả) của tất cả các thành phần nguyên thuỷ.
- Triển khai: các thành phần thích hợp được thực hiện để chứng minh sự đánh giá trong các trường hợp sử dụng và các rủi ro gặp phải để đạt được chất lượng của hệ thống.

Mặc dù mô tả kĩ thuật chi tiết của kiến trúc không tập trung vào quản lý phần mềm, tinh thần cơ bản của phát triển kiến trúc là điều cốt yếu thứ nhất để thành công.Các điều rút ra từ cơ sở này (cái gì thuộc kiến trúc và cái gì là không) là sự thách thức đối với người quản lý dự án, đây là vấn đề cuối cùng của sự cân bằng ảnh hưởng nhất định đến sự thành công của dự án.

Một ranh giới của kiến trúc được xem như là sự cân bằng giữa tập con các thông tin chạy qua tất cả các tập hợp, trong khi miêu tả kiến trúc, được gói gọn trong tập các thiết kế và khoảng cách này là rất nhỏ nhưng là sự khác nhau quan trọng giữa cách tiếp cận truyền thống và quá trình phát triển tương tác hiện đại. Các cách tiếp cận truyền thống có thể coi ranh giới của kiến trúc ở sự mô tả kiến trúc (được xác định như một tài liệu không yêu cầu sự khắt khe của hệ thống các kí hiệu thiết kế), không có bất kì sự miêu tả nào trong tập các mẫu công nghệ để phê chuẩn tính hợp lệ của các mô tả. Trong môi trường tương tác, một ranh giới kiến trúc là sự thực hiện một bộ phận của mô tả kiến trúc mà đủ để cung cấp một bằng chứng rằng kiến trúc là phù hợp trong hoàn cảnh của ngững yêu cầu và kế hoạch.

Mô tả kiến trúc sẽ có một dải rộng các dạng khác nhau, từ đơn giản, tập con trực tiếp của biểu đồ UML đến tập phức tạp của các mẫu với sự biến đổi rất khác nhau của các quan điểm rất khác nhau để đạt được và phù hợp với các vấn đề liên quan của hệ thống phức tạp.Các dạmg trước có thể ứng dụng được cho đội xây dựng các công cụ phát triển cỡ nhỏ và có kĩ năng cao, dạng sau áp dụng cho các hệ thống ra lệnh và điều khiển có độ phân phối lớn, mật độ cao, thiệt hại do rủi ro khi thất bại lớn.

Các tập mẫu liên quan trong suốt vòng đời của dự án từ bước công nghệ (trọng tâm là các yêu cầu và thiết kế mẫu đó) cho tới bước sản xuất (khi trọng tâm dịch chuyển sang sự trình bày và triển khai mẫu).

Điểm chuyển tiếp từ bước công nghệ sang bước sản xuất tạo thành một bước mà ở đó dự án đã đạt được những cơ sở kiến trúc nhất định. Đứng trên góc nhìn về quản lý, bước này đạt được khi các cổ đông thích hợp đồng ý về cách nhìn(được hỗ trợ bởi tập các yêu cầu và các kiến trúc đã trình bày trong phần thiết kế các điều đạt được trong phần trình bày và triển khai) có thể đạt được với một giá biết rõ và theo đúng kế hoạch (như là sự hỗ trợ trong phần quản lý).

Sự minh chứng của bước này không chỉ yêu cầu phần tóm tắt và các tài liệu mà cả các mẫu khả thi minh hoạ cho các khả năng liên quan. Minh hoạ này cung cấp các phản hồi cụ thể dựa trên sự đúng đắn của các giải pháp.Càng nhiều các thành phần cơ sở được sử dụng bước này càng thành công dễ dàng. Càng nhiều các thành phần tuỳ chọn được sử dụng thì càng khó thành công và càng khó để ước lượng giá thành.

Chương 8

Luồng làm việc của tiến trình

(Workflows of the Process)

Chủ điểm (Key Points)

- Dãy hoạt động của tiến trình được tổ chức thành 7 luồng làm việc(workflows) chính : Quản lý , môi trường, yêu cầu , thiết kế , cài đặt, đánh giá và khai triển (Deployment) .
- Những pha này được thực hiện đồng thời, cùng nhiều mức công sức khác nhau và sự quan trọng của nó như là sự phát triển của dự án trong suốt quá trình xây dựng và phát triển
- Luồng làm việc quản lý (management workflows) liên quan đầu tiên đến 3 vấn đề: lên kế hoạch, điều khiển dự án và tổ chức.

Hầu hết việc mô tả quá trình sử dụng dãy những hoạt động như là định dạng thể hiện đầu tiên của chúng .Những mô tả định hướng quá trình liên tiếp là rất dễ hiểu, thể hiện , lập kế hoạch và quản lý. Trên quan điểm rằng tất cả những hoạt động vốn đã là một chuỗi liên tiếp. Tuy nhiên , chuỗi những hoạt động đơn giản này không dễ thực hiện trong những dự án phần mềm mà nó cần nhiều công sức nhóm .Những công sức này có thể bao gồm nhiều nhóm , sự tiến bộ trên những sản phẩm nhân tạo mà phải được đồng bộ , kiểm tra chéo, đồng nhất, kết hợp và tích hợp lại. Sự phân phối của các quá trình phần mềm và luồng làm việc (workflows) phụ thuộc của nó là nguồn gốc của việc phức tạp trong quản lý .

Một thiếu sút nhỏ trong quỏ trỡnh phần mềm cổ truyền là việc thể hiện macroprocess như chuỗi hoạt động liờn tiếp, tỏ phõn tớch yờu cầu đến thiết kế , viết mó, kiểm thử , tỏi phõn phối. Núi cỏch khỏc một dỏng thành cụng phải cài đặt quỏ trỡnh này, nhưng biờn giới giữa cỏc pha này khụng rừ ràng và được chấp nhận bởi những người cú liờn quan (accepted as by non-adversarial stakeholder) .Một khỏc một dỏng thật bởi điờn hỗnh là bỏ xa lũy trong việc cố gắng phõn biệt ranh giới giữa cỏc pha . Vớ dụ một nhúm dỏng cú thể đặt được việc vỏch ranh giới yờu cầu rừ ràng trước khi chuyển sang thiết kế hoặc cố gắng viết tài liệu thiết kế phỏt sinh đầy đủ trước khi chuyển sang viết mó . Kết quả là những cụng sức thừa này sẽ kộo dài trong cỏc phỏt sinh vẫn vật trong khi sự tiến triển trong việc giải quyết vấn đề quan trọng bị chậm lại thậm chớ bị ngưng lại.

Quá trình hiện đại tránh việc định rõ các pha trong chu trình phần mềm kể cả những phạm vi dễ nhận thấy nhất . Những pha – khởi đầu , phát sinh, xây dựng , và chuyển tiếp chỉ ra tình trạng của dự án hơn là chuỗi hoạt động trong quá trình thiết kế kiểu thác nước(waterfall). Mục tiêu là nhận thức rõ ràng sự liên tục của hoạt động trong tất cả các pha và tránh kết luận rằng có một chuỗi tiến triển từ yêu cầu đến thiết kế , viết mã , kiểm thử , phân phối.

8.1 Luồng làm việc củatiến trình phần mềm

(Sofware process workflows)

Chương trước đã giới thiệu life-cycle macroprocess và tập hợp cơ bản của những mẫu tạo tác. Macroprocess bao gồm các pha riêng biệt và việc lặp lại(iterations) chứ không phải là những hoạt động riêng biệt .Dãy những hoạt động liên tiếp xảy ra trong mỗi pha và lặp lại (iterations) .Mô tả quá trình mức tiếp theo là microprocess hoặc workflows mà chúng xây dựng nên những mẫu tạo tác . Thuật ngữ ***workflows*** được sử dụng để chỉ một chuỗi dính liền và hầu

hết những hoạt động liên tiếp .Workflows được ánh xạ tới mẫu tạo tác như mô tả ở chương 6 và tới nhóm dự án ở chương 11 . Có 7 lớp luồng làm việc chính(workflows):

1. Luồng quản lý(Management workflow): điều khiển quá trình và bảo đảm vượt qua tình trạng stakehoder
2. Luồng môi trường(Environment workflow): tự động hoá quá trình và mở ra môI trường duy trì
3. Luồng yêu cầu(Requirement workflow): phân tích không gian bàI toán và đưa ra mẫu yêu cầu (requiremet artifact)
4. Luồng thiết kế(Design workflow): đưa ra giảI pháp , kiến trúc , và mẫu thiết kế (design artifacts)
5. Luồng cài đặt (Implementation workflow): lập trình các thành phần , triển khai và cài đặt
6. Luồng đánh giá(Assessment workflow): định ra phương hướng trong tiến trình và chất lượng sản phẩm
7. Luồng triển khai(Deployment workflow): chuyển sản phẩm cuối cùng đến người sử dụng

Hình 8-1 minh hoạ mức liên quan của các mức công sức mong chờ trong các pha trong mỗi luồng. Nó thể hiện đặc trưng của sườn tiến trình hiện đại và minh hoạ một số nguyên tắc đã nói đến ở chương 4.

1. Bước xây dựng đầu tiên(Architecture-fist approach)

Phân tích yêu cầu , thiết kế , cài đặt,và chuỗi đánh giá được thực hiện trước các pha xây dựng , khi việc cài đặt đầy đủ là trọng tâm .Trọng tâm vòng đời sớm này trong cài đặt và kiểm thử kiến trúc phảI được phát triển trước và thử nghiệm tất cả các thành phần .

WORKFLOWS	ARTIFACTS	LIFE-CYCLE EMPHASIC
Quản lý	Công nghệ phần mềm thương mại Phát triển phần mềm Kế hoạch Định giá tình trạng Vison	Khởi đầu: Chuẩn bị công nghệ thương mại và vison Phát sinh: Phát triển kế hoạch Xây dựng : Giám sát và điều khiển phát triển Chuyển tiếp: Giám sát và đIều khiển phát triển

	Work BreakDown Cấu trúc	
Môi trường	Môi trường Thứ tự thay đổi phần mềm	Khởi đầu: Xác định môi trường phát triển và thay đổi cơ sở hạ tầng quản lý Phát sinh: Cài đặt môi trường phát triển, và thiết lập cơ sở dữ liệu quản lý biến đổi Xây dựng: Duy trì môI trường phát triển và cơ sở quản lý dữ liệu biến đổi
Yêu cầu	Tập yêu cầu Đặc tả phiên bản Vison	Khởi đầu: Định nghĩa các thao tác Phát sinh: Xác định đối tượng cấu trúc Xây dựng :Xác định đối tượng lặp(iterations) Chuyển tiếp: Lọc đối tượng phiên bản
Thiết kế	Tập thiết kế Mô tả cấu trúc	Khởi đầu: Phát biểu kháI niệm cấu trúc Phát sinh: thực hiện cơ sở kiến trúc Xây dựng : Thiết kế các thành phần Chuyển tiếp: lọc cấu trúc và các thành phần
Cài đặt	Tập càI đặt Tập triển khai	Khởi đầu : Cung cấp định dạng kiến trúc Phát sinh: thiết lập cơ sở kiến trúc Xây dựng : xây dựng toàn bộ các thành phần Chuyển tiếp: Duy trì các thành phần
Đánh giá	Đặc tả Phiên bản Mô tả Phiên bản Tuỳ chọn Tập triển khai	Khởi đầu : đánh giá kế hoạch , vison, định dạng Phát sinh: đánh giá kiến trúc Xây dựng:đánh giá Phiên bản chuyển tiếp Chuyển tiếp: đánh giá Phiên bản sản phẩm

Triển khai	Tập triển khai	Khởi đầu : phân tích nhóm người sử dụng Phát sinh: Xác định tuỳ chọn Xây dựng: chuẩn bị tàI liệu chuyển tiếp Chuyển tiếp: Chuyển sản phẩm đến người sử dụng

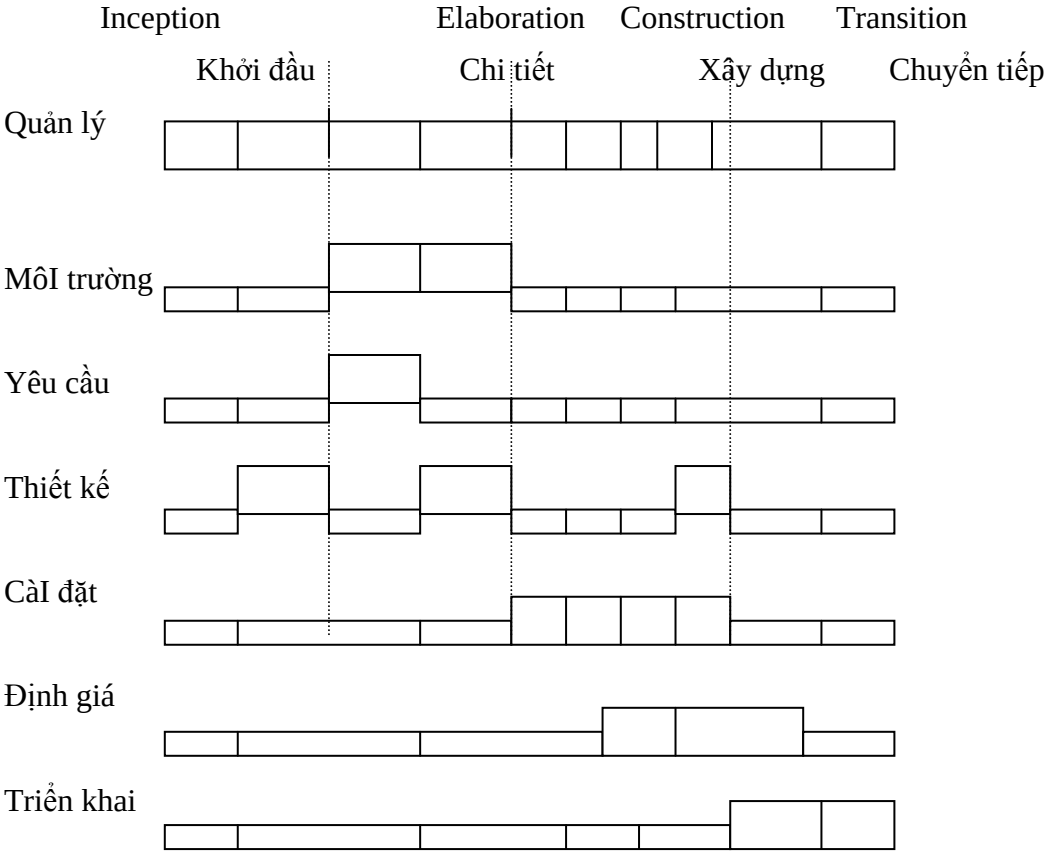


Fig 8-1

2. Tiến trình lặp(Iterative life-cycle process): Trong hình 8-1 mỗi pha miêu tả ít nhất hai giai đoạn lặp của mỗi luồng .Mặc định này được dùng để miêu tả chứ không phảI là nguyên tắc. Một vàI dự án có thể chỉ yêu cầu duy nhất 1 giai đoạn lặp trong mỗi pha , những dự án khác có thể yêu cầu vàI giai đoạn lặp . ĐIểm quan trọng là dãy hoạt động và mẫu của bất kỳ một luồng nào có thể yêu cầu nhiều hơn một để đạt được kết quả đầy đủ.

3. Công nghệ Round-Trip(Round-Trip engineering) : đưa dãy hoạt động môI trường lên luồng lớp thứ nhất là rất hạn chế . MôI trường là biểu hiện hữu hình của tiến trình của dự án , phương pháp , và ghi chú cho việc sản suất mẫu.
4. Demonstration-based approach : những hoạt động càI đặt và đánh giá được khởi tạo ngay sớm trong vòng đời phần mềm, mang lại sự quan trọng trong xây dựng tập hợp con có thể thực hiện được của kiến trúc mở.

Bảng 8-1 thể hiện sự phân phối của mẫu và tầm quan trọng của mỗi luồng mỗi pha của khởi đầu, phát sinh, xây dựng , chuyển tiếp.

Yêu cầu: phân tích kế hoạch cơ sở , kiến trúc cơ sở , yêu cầu tạo tác cơ sở để hoàn toàn chi tiết hoá sự sử dụng những trường hợp sẽ xảy ra vào lúc cuối của sự lặp và tiêu chuẩn ước lượng của chúng ; cập nhật tất cả những tạo tác yêu cầu để phản chiếu rằng thay đổi được cần phảI có bởi những kết quả của những hoạt động kĩ thuật của sự lặp này

Thiết kế: Mở ra kiến trúc cơ sở và mẫu thiết kế cơ sở được đặt mẫu để chi tiết hoá hoàn toàn mô hình thiết kếvà kiểm tra những thành phần mẫu cần thiết để chống lại tiêu chuẩn ước lượng được đề ra trong sự lặp; cập nhật tất cả những tạo tác thiết kế để phản chiếu rằng thay đổi được cần phảI có bởi những kết quả của những hoạt động kĩ thuật của sự lặp này

CàI đặt : phát triển và thu nhận bất cứ thành phần mới nào , tăng cường hoặc sửa đổi bất kỳ thành phần hiện hữu nào để ước lượng tiêu chuẩn cấp phát tới sự lặp , tích hợp và kiểm thử tất cả những thành phần mới cùng với những cơ sở đã có

Đánh giá : Ước lượng kết quả của sư lặp , bao gồm sự chiểu theo với cấp phát tiêu chuẩn ước lương và chất lượng của những cơ sở hiện thời này ; xác định bất kì công việc làm lại nào được yêu cầu và xác định liệu có phảI nó được thực hiện trước việc triển khai của phiên bản này hoặc được cấp phát tới phiên bản tiếp theo, đánh giá những kết quả để cảI thiện những cơ bản kế hoạch của sự lăp tiếp theo

Triển khai: chuyển phiên bản tới một tổ chức bên ngoàI (người dùng,người đấu thầu, đại lý) hoặc tới tổ chức bên trong

Cùng với bất kì chuỗi luồng phát triển phần mềm , các hoạt động diễn ra đồng thời .Ví dụ phân tích yêu cầu không được hoàn thành liên tục nó được trộn lẫn cùng với quản lý, thiết kế , càI đặt vv...

Sự lặp trong pha khởi đầu và chi tiết chú trọng ở quản lý , yêu cầu , và thiết kế. Sự lặp trong pha xây dựng chú trọng ở thiết kế, càI đặt và đánh giá. Sự lặp trong chuyển tiếp chú trọng trong đánh giá và triển khai .

Những miêu tả này là tương đối đơn giản . Trên thực tế nhiều chuỗi này và có lẽ cả sự lặp trở nên phức tạp hơn . Các thuật ngữ Iteration và Icrement có quan hệ với một vàI sự quan tâm trong thực tế. Một Iteration thể hiện tình trạng của toàn bộ kiến trúc và toàn bộ

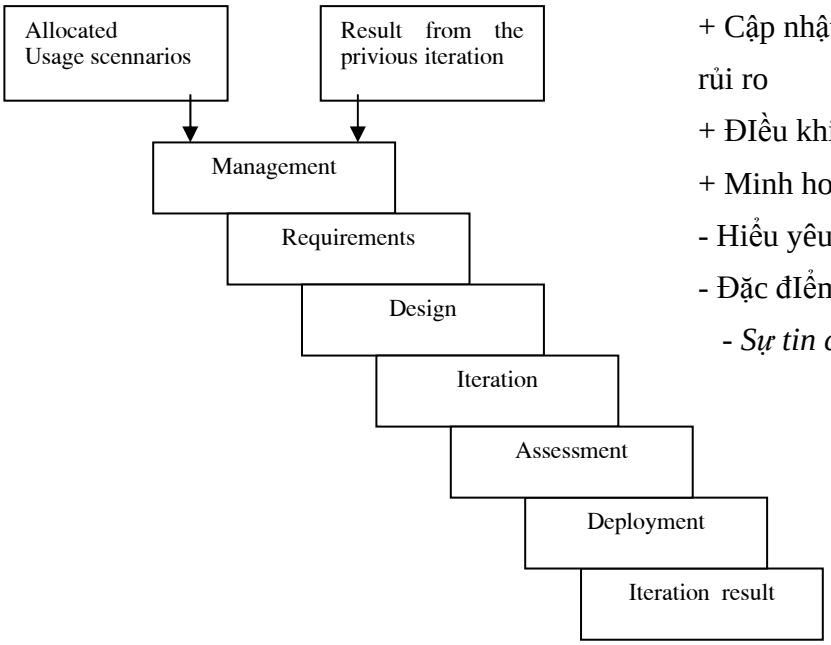
hệ thống phân phối .Một ỉncement thể hiện luồng công nghệ của yêu cầu , thiết kế càI đặt và đánh giá .

8.2 Luồng lặp (Iteration workflows)

Một Iteration bao gồm chuỗi rời rạc hoạt động trong nhiều vị trí phụ thục vào Iteration trong vòng phát triển phần mềm.Mỗi Iteration định nghĩa trong tập kịch bản sử dụng. Những thành phần cần càI đặt tất cả những kịch bản lựa chọn được phát triển và tích hợp cùng với kết quả của nhiều Iteration .Một luồng Iterarion riêng minh hoạ trên hình 8-2 bao gồm những chuỗi sau

Quản lý: Kế hoạch Iteration để chỉ ra nội dụng của phiên bản và phát triển kế hoạch chi tiết cho Iteration ; phân công gói việc ,nhiệm vụ tới nhóm phát triển

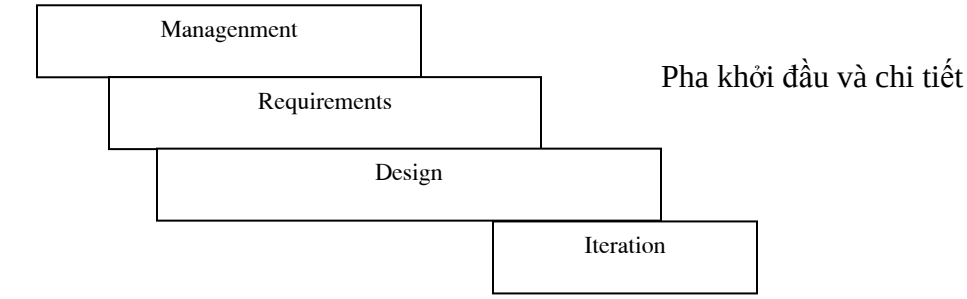
MôI trường : Mở ra phần mềm thay đổi thứ tự cơ sở dữ liệu dể phản chiếu những cơ sở mới và thay đổi cơ sở đã tồn tại cho tất cả sản phẩm , kiểm thử và môI các thành phần môI trường .



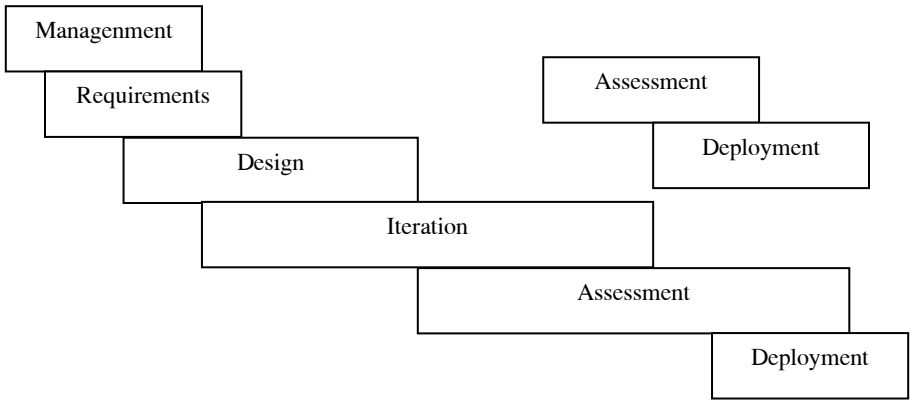
+ Cập nhật đánh giá
rủi ro
+ ĐIều khiển cơ sở mẫu
+ Minh hoạ kết quả
- Hiểu yêu cầu
- Đặc đIểm và thể hiện thiết kế
- *Sự tin cậy của kế hoạch*

Fig 8-2 Luồng của Iteration

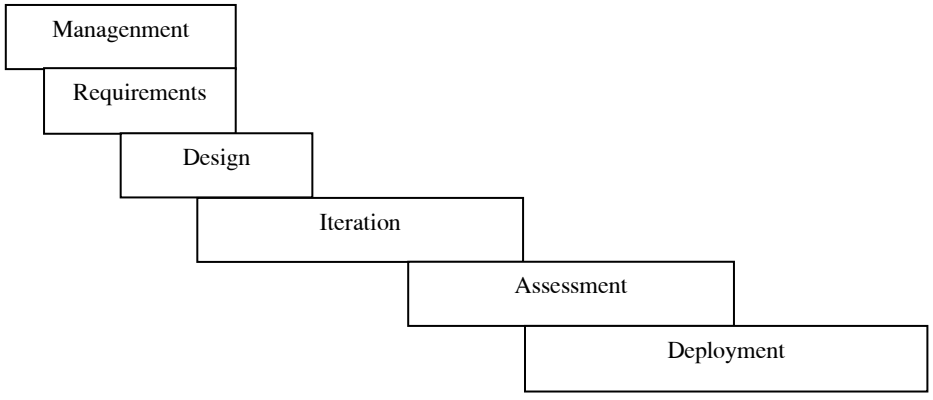
Công việc hiện tại trong tiến trình sẽ được tổ hợp cùng với Iteration trước tới Iteration sau hình 8-4 một ví dụ vòng phát triển đơn giản,chiến lược khác giữa Iterations và Increments



Pha khởi đầu và chi tiết

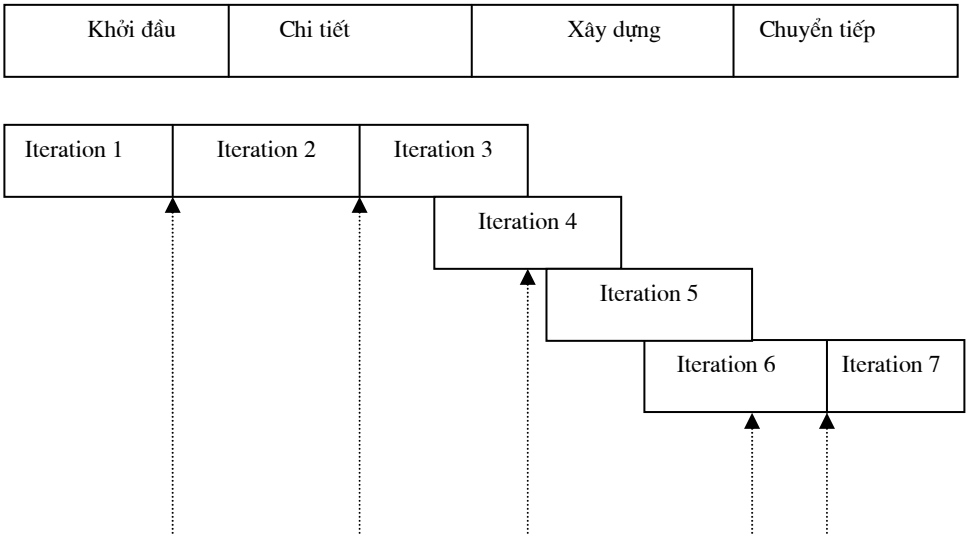


Pha xây dựng



Pha chuyển tiếp

Fig 8-3



Chương 9
Các điểm kiểm tra quá trình
(Checking point of the process)

Những cột mốc rõ ràng trong vòng đời phần mềm luôn quan trọng, nó là nơi những người có liên quan gặp nhau,và thảo luận về các tiến triển và đề ra các kế hoach. Những sự kiện này không chỉ để xác định dự án đã đi đến đâu mà còn nhằm đạt được các mục tiêu sau:

- Đồng bộ hoá các dự tính của những người có liên quan và những tiến bộ đạt được trên ba tiến độ: các yêu cầu, thiết kế, và kế hoạch.
- Đồng bộ hoá những phần việc có liên quan đến nhau thành một trạng thái cân bằng vững chắc.
- Xác định các rủi ro chính, các vấn đề quan trọng, và các điều kiện không chấp của dự nhận được.

Các điểm quan trọng

♦Có một dãy các điểm kiểm tra dùng để đồng bộ hoá các mong muốn của những người có liên quan trong vòng đời của dự án: các cột mốc chính, các cột mốc phụ và các đánh giá tình trạng.

♦Cột mốc chính quan trọng nhất là cột mốc chuyển từ pha chuẩn bị sang pha thực hiện

♦Thể thức và nội dung của các cột mốc phụ phụ thuộc rất lớn vào dự án và cách làm việc của tổ chức.

♦Các đánh giá tình trạng định kì chủ yếu nhằm tập trung sự chú ý không ngừng tới tình trạng án và các phần được ưu tiên.

- Đưa ra một đánh giá tổng thể cho toàn bộ vòng đời chứ không phải chỉ là tình trạng hiện của một tiến độ công việc nào đó hay của sản phẩm trung gian.

Các cột mốc phải có những dự tính rõ ràng và đưa ra được những kết quả cụ thể. Điều này không cản trở việc đàm phán lại các mục tiêu của cột mốc khi vấn đề cân bằng giữa các yếu tố: các yêu cầu, thiết kế, kế hoạch trong dự án đạt được những tiến bộ xa hơn.

Có ba kiểu xem xét được tiến hành trong quá trình thực hiện dự án:

1. Các cột mốc chính: Những sự kiện lớn như thế này được tổ chức vào cuối mỗi pha phát triển. Chúng cung cấp cái nhìn cụ thể về những vấn đề lớn, đồng bộ hoá các

viễn cảnh của việc quản lí và công nghệ, xác nhận rằng mục tiêu của pha đã đạt được.

2. Các cột mốc phụ: Những sự kiện này tập trung vào lần lặp, chúng diễn ra nhằm xem xét lại nội dung lần lặp một cách chi tiết và cho phép dự án được tiếp tục.
3. Các đánh giá về tình trạng dự án: Những sự kiện này lặp đi lặp lại định kì, chúng cung cấp cho việc quản lý cái nhìn cặn kẽ và thường xuyên về tình trạng dự án.

Mỗi pha trong bốn pha: khởi đầu, chuẩn bị, thực hiện, chuyển giao bao gồm một hay nhiều lần lặp và kết thúc với một cột mốc chính khi khả năng kĩ thuật đề ra là có tính khả thi. Mỗi lần lặp thể hiện một chu trình các hoạt động mà ở đó kết quả trung gian - cột mốc phụ được xác định rõ ràng - bao gồm hai phần: đặc điểm kĩ thuật được xác định (tiêu chuẩn đánh giá và kế hoạch) và sự mô tả được đưa ra (kết quả). Những cột mốc chính ở cuối mỗi pha thường mang tính nghi thức, những người có liên quan đồng ý với tiêu chuẩn đánh giá và những sự mô tả được đưa ra; các cột mốc phụ thì không mang nặng tính nghi thức, các kiểu công viêc như thế này được kiểm soát bởi đội phát triển.

Mức độ nghi thức và số cột mốc phụ thuộc vào một số yếu tố, chẳng hạn như quy mô, số người có liên quan, phạm vi công việc, rủi ro kĩ thuật, và sự nhạy cảm về giá cả cũng như sự lo lắng về kế hoạch. Hầu hết các dự án có bốn cột mốc chính. Chỉ trong các trường hợp đặc biệt thì số cột mốc mới nhiều hơn hay ít hơn (với một dự án quan trọng mang tính quốc gia và được nghiên cứu một cách kĩ lưỡng có lẽ sẽ cần nhiều hơn, với một thử nghiệm khoa học thì có lẽ cần ít hơn). Với một dự án đơn giản hơn, rất ít hoặc không cần phải có một cột mốc phụ nào để đạt được các kết quả trung gian, và các đánh giá về tình trạng dự án có thể là không thường xuyên (ví dụ: hàng quý.) Hình 9-1 minh hoạ một dãy các điểm kiểm tra dự án cụ thể cho một dự án tương đối lớn.

9.1 Các cột mốc chính

Các mô tả trong mục này bám sát cách tiếp cận “điểm neo vòng đời “ trong cuốn “Neo quy trình phần mềm” (Boehm, 1996). Bốn cột mốc chính xảy ra ở thời điểm chuyển tiếp giữa các pha trong vòng đời. Chúng có thể được dùng trong nhiều mô hình xử lý khác nhau, trong đó có mô hình thác nước truyền thống. Trong mô hình lặp, các cột mốc chính diễn ra nhằm đạt được sự thống nhất giữa những người liên quan về tình trạng hiện tại của dự án. Những người có liên quan có những mối quan tâm rất khác nhau:

Khởi đầu	Chuẩn bị	Thực hiện	Chuyển giao
Lần lặp 1	Lần lặp 2 Lần lặp 3	Lần lặp 4 Lần lặp5 Lần lặp 6	Lần lặp 7

Hình 9-1: Một dãy các điểm kiểm tra vòng đời cụ thể

- Khách hàng: các dự đoán về thời hạn và ngân quỹ, tính khả thi, các đánh giá rủi ro, việc hiểu các yêu cầu, sự tiến triển và tính tương thích của dòng sản phẩm.
- Người dùng: sự phù hợp với các yêu cầu và tương lai của tình huống sử dụng, khả năng nâng cấp dễ dàng, các thuộc tính chất lượng.
- Kiến trúc sư và kĩ sư hệ thống: tính tương thích của dòng sản phẩm, các yêu cầu luôn thay đổi, phân tích để đạt được sự cân bằng giữa các yếu tố khác nhau, tính trọn vẹn và tính phù hợp, sự cân bằng giữa các yếu tố: rủi ro, chất lượng và tiện dụng.
- Người phát triển: Chi tiết của các yêu cầu được cung cấp đầy đủ, sự mô tả về tình huống sử dụng trong tương lai, cơ cấu cho việc chọn lựa hay phát triển các thành phần, việc giải quyết các vấn đề rủi ro trong phát triển, tính tương thích của dòng sản phẩm, môi trường phát triển thuận lợi.
- Người bảo trì: Sự cung cấp đầy đủ các phần của sản phẩm đươc xác nhận bằng tài liệu , có thể hiểu được, tương thích với hệ thống đang tồn tại, môi trường bảo trì thuận lợi.
- Những người khác: có thể có những người liên quan khác như các đại lý, những nhà đấu thầu thẩm tra và công nhận độc lập, những nhà đầu tư, những nhà thầu phụ, những nhà thầu liên kết, đội ngũ bán hàng và tiếp thị.

Những cột mốc được nêu ra trong mục này có thể diễn ra như là cuộc gặp giữa những người quan tâm đến dự án hay thông qua việc xem xét các phần việc qua mạng máy tính. Có sự khác nhau đáng kể về nghi thức trong các sự kiện này, phụ thuộc vào vài nhân tố sẽ được xét tới trong chương 14. Thực chất, cột mốc chính đảm bảo việc hiểu các yêu cầu, hình thức của sản phẩm, chức năng, và chất lượng được phát triển một cách cân bằng và đảm bảo tính phù hợp giữa các thành phần khác nhau. Bảng 9-1 tóm tắt các thông tin về những cột mốc chính.

Cột mốc mục tiêu vòng đời

Cột mốc mục tiêu vòng đời diễn ra vào cuối pha khởi đầu. Nó giới thiệu với tất cả những người có liên quan về cách thức...(bị mờ), dự đoán giá cả và lịch làm việc, lợi nhuận và tích

luỹ. Tuyên bố về tầm nhìn và và những vấn đề giới hạn liên quan tới các khái niệm có thể dùng được đưa ra. Tài liệu về kiến trúc được phác thảo và kiến trúc của hình mẫu đầu tiên có tính khả thi sẽ cung cấp một bằng chứng trọn vẹn về tầm nhìn và kế hoạch phát triển phần mềm. Cột mốc mục tiêu vòng đời đầu tiên diễn ra thành công sẽ cho phép tất cả những người liên quan chuyển sang pha chuẩn bị.

Cột mốc kiến trúc vòng đời

Cột mốc kiến trúc vòng đời diễn ra vào cuối pha chuẩn bị. Mục đích chính là giới thiệu một kiến trúc khả thi tới tất cả người liên quan. Kế hoạch chi tiết hơn về cho pha thực hiện được giới thiệu để tìm sự ủng hộ. Các vấn đề giới hạn liên quan đến các yêu cầu và các khái niệm có thể được dùng được đưa ra. Cột mốc này cũng nhằm đạt được sự nhất trí về ranh giới của kiến trúc, ranh giới tầm nhìn, ranh giới kế hoạch phát triển phần mềm, và tiêu chuẩn đánh giá về cột mốc khả năng có thể được sử dụng lần đầu. Ranh giới kiến trúc bao gồm cả sự giới thiệu dưới dạng có thể đọc (tài liệu về kiến trúc) và cấu hình - một tập hợp (được kiểm soát) các thành phần của phần mềm trong các công việc mang tính công nghệ. Cột mốc kiến trúc vòng đời thành công sẽ cho phép những người có liên quan chuyển sang pha thực hiện.

Bảng 9-1. Các tình trạng của một dự án thông thường, các yêu cầu và các kết quả đạt đươc từ các cột mốc chính

Các cột mốc	**Các kế hoạch**	**Việc hiểu các vấn đề đặt ra (các yêu cầu)**	**Quá trình giải quyết (sản phẩm phần mềm)**
Cột mốc mục tiêu vòng đời	Xác định trách nhiệm của những người có liên quan. Kế	Ranh giới tầm nhìn, bao gồm phương hướng phát triển, các	Đưa ra ít nhất một kiến trúc khả thi. Việc cân bằng các yếu tố chế tạo, mua

	hoạch có độ trung thực thấp cho vòng đời. Kế hoạch có độ trung thực cao cho pha chuẩn bị.	thuộc tính chất lượng,và các quyền ưu tiên. Mô hình cho tình huống sử dụng	bán, tái sử dụng. Một thiết kế đầu tiên
Cột mốc kiến trúc vòng đời	Kế hoạch có độ trung thực cao cho pha thực hiện (dự thảo các thứ cần thiết, việc dùng lao động). Kế hoạch có độ trung thực thấp cho pha chuyển giao.	Tầm nhìn ổn định và mô hình cho tình huống sử dụng . Đưa ra các tiêu chuẩn đánh giá cho việc thực hiện, khả năng có thể dùng lần đầu. Phác thảo sổ tay người dùng	Tập hợp các thiết kế ổn định. Quyết định việc chế tạo, mua bán, tái sử dụng. Các thành phần giới hạn của mẫu đầu tiên.
Cột mốc khả năng dùng lần đầu	Kế hoạch có độ trung thực cao cho pha chuyển giao.	Tiêu chuẩn chấp nhận đươc cho sản phẩm được đưa ra. Sổ tay người dùng có thể đưa ra.	Tập hợp các bổ sung ổn định. Các đặc điểm giới hạn và khả năng chính. Hiểu biết một cách sâu sắc về chất lượng sản phẩm.
Cột mốc phát hành sản phẩm.	Kế hoạch cho thế hệ sản phẩm tiếp theo.	Sổ tay người dùng được quyết định dứt khoát.	Việc phát triển ổn định. Đầy đủ các đặc điểm. Chất lượng của sản phẩm.

Vì cột mốc chính quan trọng nhất là cột mốc chuyển từ pha chuẩn bị sang pha thực hiện. Nội dung cụ thể của việc chuẩn bị cho một pha được thể hiện chi tiết hơn ở đây. Xuất phát từ quan điểm hợp đồng và quản lý, cột mốc này nhằm đạt được trạng thái phát triển của phần mềm mà ở đó công tác nghiên cứu và phát triển phần mềm được chấm dứt còn việc sản xuất

sản phẩm được bắt đầu. Một dự án phát triển phần mềm sẵn sàng cho việc chuyển tiếp này có các đặc điểm sau:

- Giới hạn các trường hợp sử dụng được xác định, được đồng ý bởi những người có liên quan, và được hệ thống hoá thành tập hợp các đánh giá về viễn cảnh của việc phát triển kiến trúc.
- Một kiến trúc ổn định được vạch ra (đưa ra việc quản lý cấu hình) dưới dạng thức ngôn ngữ nguồn. Sự ổn định ở đây có nghĩa là các yếu tố chất lượng của kiến trúc (hiệu quả, chắc chắn, khả năng phát triển, khả năng thích ứng) được chứng minh là có thể vừa đáp ứng các tình huống sử dụng vừa giải quyết được tất cả các yêu cầu chính và thiết kế và dự tính các rủi ro (mặc dầu vấn đề rủi ro có thể không được giải quyết nhưng phương hướng để giải quyết chúng phải được xác định)
- Có một mô tả sơ lược về các rủi ro nhằm mọi người có thể hiểu được. Mặc dầu không nhất thiết phải giải quyết tất cả mọi rủi ro nhưng những người có liên quan nên có các hiểu biết căn bản về các rủi ro còn tồn tại mà chúng có thể gây ra những hậu quả nghiêm trọng và các phương án khắc phục cần phải được chuẩn bị đầy đủ.
- Kế hoạch phát triển cho pha thực hiện và pha chuyển chuyển giao được xác định với đủ độ chính xác cần thiết nhằm mục đích có thể dự đoán được kết quả việc thực hiện. Có thể dự đoán ở đây có nghĩa là tổ chức phát triển sẽ cam kết giữ nguyên giá cả với người dùng trong vòng ít hơn một năm.

I. Các đòi hỏi

A. Mô hình cho tình huống sử dụng

B. Tài liệu về tầm nhìn (văn bản, các trường hợp dùng)

C. Tiêu chuẩn đánh giá cho việc chuẩn bị (văn bản, các viễn cảnh)

II. Kiến trúc

A. Quan điểm thiết kế (các mô hình của đối tượng)

B. Quan điểm xử lý (nếu cần thiết, cách bố trí lúc chạy, cấu trúc mã có khả năng chạy được)

C. Quan điểm về các thành phần (việc bố trí các hệ thống con, xác định các thành phần được sản xuất / mua / tái sử dụng).

D. Quan điểm phát triển (mục tiêu của cách bố trí lúc chạy, mục tiêu cấu trúc mã lệnh có khả năng chạy được)

E. Quan điểm về tình huống sử dụng (cấu trúc cho các lần

thử nghiệm, các mong đợi về kết quả thử nghiệm)

1. Phác thảo sổ tay người dùng.

III. Các thư viện tài nguyên và có khả năng chạy được.

A. Các thành phần sản phẩm.

B. Các thành phần thử nghiệm.

C. Các thành phần môi trường và công cụ.

Hình 9-2. Các công việc mang tính công nghệ trong cột mốc kiến trúc vòng đời

Nội dung của cột mốc này sẽ thay đổi tuỳ theo lĩnh vực của dự án. Tốt nhất, các mục sau đây nên có:

- Giới thiệu và điểm qua tình trạng hiện tại của dự án.
- Cấu hình - tập hợp (được kiểm soát) các thông tin công nghệ, dưới dạng tài liệu điện tử hay tài liệu in ra giấy.
- Chứng minh tính khả thi.

Dữ liệu kĩ thuật được đưa ra trong hình 9-2 nên được xem xét trước cột mốc kiến trúc vòng đời. Hình 9-3 cung cấp một cách tóm tắt các công việc phải làm trong cột mốc này.

Chương trình giới thiệu

I. Phạm vi và mục tiêu

A. Thuyết minh lướt qua.

II. Đánh giá các yêu cầu

A. Tầm nhìn dự án và các trường hợp dùng.

B. Viễn cảnh chính và các chỉ tiêu đánh giá

III. Đánh giá về kiến trúc

A. Quá trình xúc tiến

1. Đánh giá kiến trúc vạch ra (xúc tiến việc cập nhật và vạch ranh giới cho việc đánh giá độ ổn định của kiến trúc tương lai, chia nhỏ, thực hiện lại).
2. Dự đoán ranh giới của các đánh giá phát triển (nhằm đánh giá các phát triển trong tương lai).
3. Dự đoán ranh giới các thử nghiệm (nhằm đánh giá sự phát triển trong tương lai của

đội thử nghiệm).

B. Chất lượng

1. Các đặc điểm kiến trúc (tóm tắt khả năng đã được chứng minh, và các tiêu chuẩn đánh giá)
2. Hiệu quả (tóm tắt khả năng đã khả thi, và các tiêu chuẩn đánh giá)
3. Công bố các rủi ro của kiến trúc và kế hoạch giải quyết.
4. Khả năng và việc cân bằng giữa việc sản xuất / mua / tái sử dụng.

IV. Đánh giá về kế hoạch của pha thực hiện

A. Nội dung lần lặp và xác định các trường hợp sử dụng.

B. Kế hoạch chi tiết cho lần lặp tiếp theo và các tiêu chuẩn đánh giá.

C. Hiệu quả về mặt lịch trình và giá cả của pha chuẩn bị.

D. Kế hoạch tài nguyên cho pha thực hiện và cơ sở của kế hoạch này.

E. Đánh giá rủi ro.

Lịch trình thuyết minh

I. Tiêu chuẩn đánh giá.

II. Tóm tắt tập hợp kiến trúc.

III. Tóm tắt môi trường diễn thuyết.

IV. Các thuyết minh được đưa ra theo lịch trình.

V. **Các kết quả tiêu chuẩn đánh giá và mục tiếp theo.**

Hình 9-3. Lịch trình tóm tắt cho cột mốc kiến trúc vòng đời

Cột mốc khả năng có thể sử dụng lần đầu

Cột mốc có thể bắt đầu sử dụng diễn ra vào cuối pha thực hiện. Mục đích nhằm đánh giá sự sẵn sàng của phần mềm cho việc chuyển giao tới khách hàng/ các địa điểm và cho phép bắt đầu các thử nghiệm. Vấn đề được đề cập tập trung vào các chỉ dẫn cài đặt, các mô tả về

phiên bản phần mềm, sổ tay người sử dụng và người vận hành, và khả năng tổ chức phát triển trợ giúp các vùng người dùng. Đánh giá chất lượng phần mềm nhằm xác định chất lượng có đủ để chuyển giao hay không. Việc chuẩn bị sẵn sàng môi trường thử nghiệm và phần mềm thử nghiệm được đề cập đến. Các thử nghiệm có thể làm tăng số lần lặp lên nhiều lần hoặc có thể được hoàn chỉnh trong pha chuyển giao. Việc bắt đầu pha chuyển giao không nhất thiết phải diễn ra sau khi hoàn thành pha thực hiện. Hai pha này đan xen vào nhau cho tới khi sản phẩm đầu tiên được đưa đến tay người sử dụng.

- **Cột mốc phát hành sản phẩm :**

Cột mốc phát hành sản phẩm diễn ra vào cuối pha chuyển giao. Mục đích nhằm đánh giá tình trạng của phần mềm khi hoàn thành và chuyển nó tới tổ chức trợ giúp. Các kết quả thử nghiệm được xem xét, và tất cả các vấn đề mở được chấp nhận. Những vấn đề này bao gồm các chỉ dẫn cài đặt, những mô tả về phiên bản phần mềm, sổ tay người sử dụng và người vận hành, sổ tay trợ giúp phần mềm và môi trường triển khai cài đặt ở những vùng trợ giúp. Các đánh giá chất lượng phần mềm nhằm xác định chất lượng có đủ để chuyển tới tổ chức trợ giúp hay không.

9.2 .Các cột mốc phụ

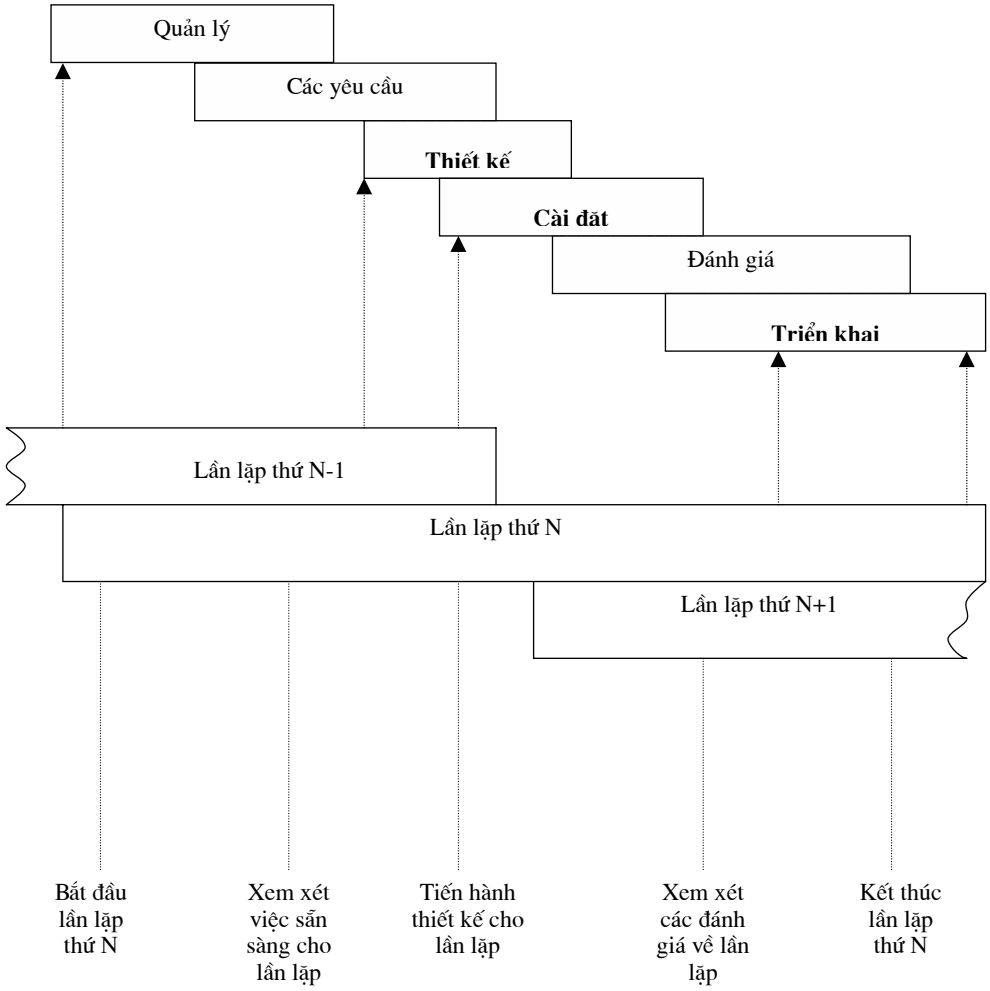
Số cột mốc riêng biệt được lặp đi lặp lại và không mang tính nghi thức này phụ thuộc vào khoảng thời gian của lần lặp. Với hầu hết các lần lặp kéo dài từ một đến sáu tháng, chỉ có hai cột mốc phụ: xem xét việc sẵn sàng cho lần lặp và xem xét việc đánh giá về lần lặp. Với các lần lặp dài hơn, thì sẽ có nhiều điểm trung gian hơn. Ví dụ, với những dự án có quá trình thử nghiệm mang tính nghi thức cao, phải được chứng kiến bởi những người có liên quan thì việc xem xét sự sẵn sàng cho quá trình thử nghiệm có thể diễn ra trước khi thử nghiệm diễn ra và được chấp nhận. Với quy mô lớn mà trước đây chưa từng có, cũng có thể dùng bước thiết kế trung gian như là công việc bắt buộc cho việc đánh giá và phổ biến toàn bộ dự án.

Các lần lặp có thể khác nhau. Các lần lặp khác nhau có mức độ ưu tiên rất khác nhau, phụ thuộc vào trạng thái của dự án trong vòng đời. Ban đầu là tập trung vào phân tích và thiết kế, các yếu tố được phát hiện trong thực tế, ước đoán về kinh nghiệm và rủi ro. Các lần lặp sau này tập trung nhiều hơn vào tính trọn vẹn, bao gồm khả năng sử dụng và việc quản lý luôn thay đổi. Các cột mốc của một lần lặp và những tiêu chuẩn đánh giá của nó cần tập trung vào các hoạt động công nghệ của dự án đã được xác định trong kế hoạch phát triển phần mềm, tình thế kinh doanh, và tầm nhìn.

- **Xem xét sự sẵn sàng cho lần lặp:** cột mốc không mang tính nghi thức này diễn ra đầu mỗi lần lặp để xem xét kế hoạch của lần lặp một cách chi tiết và tiêu chuẩn đánh giá cho lần lặp.

- **Xem xét đánh giá về lần lặp:** cột mốc không mang tính nghi thức này diễn ra vào cuối mỗi lần lặp để đánh giá mức độ đạt được mục tiêu của lần lặp và việc thoả mãn các tiêu chuẩn đánh giá của lần lặp, xem xét các kết quả của lần lặp, xem xét các kết quả thử nghiệm về chất lượng (Nếu là một phần của lần lặp), xác định số lượng công việc cần phải làm lại và xem xét tác động của các kết quả tới lần lặp tiếp theo.

Thể thức và nội dung của những cột mốc phụ này có phụ thuộc lớn vào dự án và cách làm việc của tổ chức. Hình 9-4 xác định các cột mốc khác nhau được xem xét khi dự án được lập kế hoạch.



Hình 9-4. Các cột mốc phụ thông thường trong vòng đời của một lần lặp

9.3 .Các đánh giá tình trạng định kì

Các rủi ro trong quản lý đòi hỏi sự chú ý không ngừng tới tất cả các hoạt động tác động lẫn nhau trong nỗ lực phát triển phần mềm. Các đánh giá tình trạng định kì là các xem xét về mặt quản lý diễn ra lặp đi lặp lại sau mỗi khoảng thời gian nào đó (hàng tháng, hàng quý) để xem xét quá trình phát triển và các chỉ tiêu chất lượng, đảm bảo sự chú ý không ngừng tới các độnglực của dự án và duy trì cơ chế giao tiếp mở giữa những người có liên quan. Mục đích tối cao của những đánh giá này là đảm bảo rằng các dự tính của những người có liên quan (nhà thầu, khách hàng, người dùng, nhà thầu phụ) được đồng bộ hoá và có tính vững chắc. Các đánh giá tình trạng định kì cung cấp cái nhìn lướt qua về dự án. Mặc dù chu kì có thể thay đổi,

nhưng các đánh giá diễn ra tuần hoàn này luôn tập trung vào lịch sử và tài liệu dự án. Các đánh giá cung cấp các nội dung sau:

- Một cơ cấu mở cho việc diễn thuyết, giao tiếp, giải quyết các vấn đề quản lý, vấn đề kĩ thuật và rủi ro của dự án
- Dữ liệu khách quan được đưa thẳng tới từ các công việc đang tiến hành và cấu hình sản phẩm đang phát triển.
- Cơ cấu cho quá trình phổ biến, phát triển, phương hướng cho chất lượng, các yếu tố thực tiễn, và việc thông báo các kinh nghiệm tới và từ tất cả những người có liên quan trong diễn đàn mở.

Các chủ đề lặp đi lặp lại từ các dự án không thành công có các đánh giá tình trạng là các hoạt động có chi phí quá cao, công việc gắn với việc tạo ra tình trạng tách rời với các công việc hằng ngày,và thường xuyên bị huỷ bỏ, bởi các vấn đề được ưu tiên hơn cần phải được giải quyết. Các chủ đề lặp đi lặp lại từ các dự án thành công là: các hoạt động có chi phí thấp, các tài liệu tồn tại như là những dữ liệu được quản lý hằng ngày, hiếm khi bị huỷ bỏ...(bị mờ)

Các đánh giá tình trạng định kì chủ yếu nhằm tập trung sự chú ý liên tục vào trạng thái phát triển của dự án và những động lực ưu tiên của nó. Chúng buộc người quản lý dự án góp nhặt và xem xét dữ liệu một cách định kì, bắt buộc việc...(bị mờ) và khuyến khích việc tuyên truyền các yếu tố thực tiễn tốt nhất tới và từ những người có liên quan, tiêu chuẩn hoá thể thức và các đánh giá được xem xét, một tổ chức cho phép so sánh dự án này với các dự án khác và tuyên truyền các yếu tố thực tiễn tốt nhất một cách hữu hiệu.

Nội dung tóm lược của các đánh giá tình trạng định kì nên gồm các phần như trong bảng 9-2. Nội dung mà người quản lí dự án nên đưa ra ngay từ đầu cho mỗi lần xem xét là việc đánh giá mười nguy cơ rủi ro hàng đầu, phần lớn là việc cập nhật lại các đánh giá trước đây. Với một quy tắc tốt, các biểu đồ đánh giá tình trạng sẽ dễ dàng được tạo ra sau một ngày nghiên cứu. Điều này có thể đạt được nếu dữ liệu tồn tại trong môi trường liên kết, chủ đề phát triển kĩ thuật sẽ được đề cập đến trong chương 13.

Bảng 9-2. *Nội dung tóm tắt của các đánh giá tình trạng*

Chủ đề	**Nội dung**
Nhân sự	Kế hoạch cung cấp và thực tế Sự mệt mỏi, các vấn đề khác
Chiều hướng tài chính	Kế hoạch chi tiêu và thực tế cho trước đây, hiện nay và các cột mốc. Dự đoán lợi tức

Mười rủi ro hàng đầu	Các vấn đề và kế hoạch giải quyết giới hạn Trình bày các phẩm chất (giá cả, thời gian, chất lượng).
Các tiến triển kĩ thuật	Các thời hạn vạch ra cho cấu hình của các cột mốc chính. Đánh giá và chỉ ra việc quản lý phần mềm Các khuynh hướng đang thay đổi ở hiện tại
Các kế hoạch cho các cột mốc chính và kết quả	Kế hoạch, danh mục, và các rủi ro cho cột mốc chính tiếp theo. Các kết quả về sự thành công hay thất bại cho tất cả các chỉ tiêu thừa nhận.
Tổng quan về sản phẩm	Tổng kích cỡ, sự phát triển, và sự lo lắng về các tiêu chuẩn được chấp nhận.

Chương 10

Lập kế hoạch tiến trình lặp

(Iterative process planning)

Điều hành dự án phần mềm là những nỗ lực liên kết nhiều lĩnh vực khác nhau. Nội dung phần III đã bàn đến những quy tắc chủ yếu cần thiết cho một quá trình điều hành hiệu quả bao gồm lập kế hoạch, tổ chức, tự động hoá và kiểm soát dự án. Những quy tắc này của quá trình điều hành dự án phần mềm không phải dễ dàng định nghĩa theo những thuật ngữ thông thường. Do đó, còn có một số quy tắc quan trọng khác cũng góp phần xây dựng nên một khuôn khổ hoạt động cho việc điều hành một dự án cụ thể.

Lập kế hoạch là điểm then chốt của việc điều hành. Vấn đề là phải xây dựng một kế hoạch có thể cân bằng các tài nguyên hiện có để cung cấp các điều kiện tối ưu cho tất cả những chủ dự án. Quy tắc tổ chức dự án có liên quan với vấn đề quản lý nhân sự – tổ chức họ thành từng nhóm và phân chia trách nhiệm cho các nhóm để đạt được hiệu quả cao. Tự động hoá quá trình thực hiện với một cách lưu trữ các sản phẩm hiện đại đã mang lại nền tảng cho việc sử dụng các công cụ khách quan. Các hoạt động kiểm soát dự án đóng vai trò trọng yếu. Chúng được sử dụng để xác định tình trạng của kế hoạch, chất lượng của sản phẩm và là điều cần thiết đối với các cấp quản lý nhằm đạt được mục tiêu của các chủ đầu tư dự án.

Các vấn đề chính:

- Các dự án có thể đã hoặc chưa xây dựng kế hoạch. Một lần nữa, việc cân đối giữa các mức dự tính chi tiết và việc thực sự bỏ vốn của các chủ đầu tư là rất quan trọng.
- Phân định cơ cấu các công việc chi tiết chính là việc thiết lập kế hoạch dự án một cách cụ thể. Cần thiết phải thâu tóm được các thay đổi và suy luận đối với mỗi một chi tiết tương ứng trong suốt chu kỳ tồn tại của dự án.
- Sử dụng các kỹ thuật phân tích tổng thể (theo cách phân tích dự án “từ trên xuống”) và các kỹ thuật phân tích cụ thể chi tiết (theo cách phân tích”từ dưới lên”) để ước tính chi phí và các nguồn ngân sách nhằm đạt được những kết quả mong đợi.

Giống như sự phát triển của phần mềm, việc lập kế hoạch dự án cần một quá trình lặp đi lặp lại. Cũng như phần mềm, kế hoạch mang tính trừu tượng cao, là một trong các sản phẩm trí tuệ cần được nghiên cứu chuyên sâu. Các kế hoạch đều trải qua một giai đoạn thiết kế, được phát triển, sau đó chúng sẽ đưa vào giai đoạn triển khai, khi mà các kế hoạch này đã có tính khả thi. Quá trình phát triển kế hoạch cho ta nhận thức sâu sắc hơn về bài toán và cách giải quyết. Sai sót trong việc xây dựng kế hoạch cũng giống như sai sót của sản phẩm, một khi

những sai sót này được giải quyết càng sớm thì ảnh hưởng xấu của chúng đến sự thành công của dự án càng nhỏ.

Xây dựng dự án toàn diện phụ thuộc vào nhiều tham số, mỗi tham số đều có ảnh hưởng lớn đến phương hướng của dự án. Tuy nhiên, mọi nhà quản lý dự án về phần mềm đều tìm kiếm các tin tức chung về việc lập kế hoạch như là một cái khung để từ đó bắt đầu xây dựng dự án. Chương này không phải là một kế hoạch, một quyển sách dạy lập kế hoạch, cũng không phải một công thức, nó chỉ là một mô hình đơn giản cho một vài trường hợp, có thể coi là điểm bắt đầu cho việc xây dựng kế hoạch.

10.1. Phân định cơ cấu các công việc chi tiết (work breakdown structures-WBS)

Việc phân định tốt các công việc chi tiết và sự đồng bộ hoá của nó với cả khuôn khổ quá trình là các nhân tố quyết định thành công của dự án phần mềm. Mặc dù các khái niệm và việc vận dụng phân định cơ cấu các công việc chi tiết này đã được chính thức hoá, chủ đề này bị ngăn ngừa một cách rộng rãi trong các tài liệu đã xuất bản. Điều này chủ yếu là do sự phát triển của một quá trình phân định các công việc phụ thuộc vào phong cách quản lý dự án, môi trường văn hoá của công ty, sự ưu đãi khách hàng, khả năng tài chính và một vài vấn đề khác khó có thể định nghĩa, các tham số kỹ thuật của dự án. *Các vấn đề kinh tế trong việc thiết kế phần mềm* [Boehm, 1981] bao hàm các kỹ thuật nền tảng trong việc phân định các công việc về phần mềm một cách chi tiết.

WBS là chỉ đơn giản một hệ thống phân cấp của các yếu tố phân tích kế hoạch dự án thành các nhiệm vụ tách rời. Một WBS cung cấp các thông tin sau đây:

+ Một phác hoạ tất cả các công việc quan trọng

+ Một công việc phân tích cụ thể cho việc phân phối các chức năng.

+ Một khung làm việc để lập lịch trình công việc, dự thảo ngân sách và theo dõi phí tổn trong quá trình làm việc.

Có nhiều tham số có thể điều khiển quá trình phân tích công việc thành các nhiệm vụ tách rời: các hệ thống phụ của sản phẩm, các thành phần, các chức năng, các khối cơ quan, các giai đoạn trong vòng đời sản phẩm... Hầu hết các hệ thống đều có một sự phân tích ở cấp đầu tiên một bởi hệ thống phụ trợ. Hệ thống phụ trợ sau đó được phân tích thành các thành phần nhỏ hơn, một trong số đó đặc trưng là phần mềm. Phần trao đổi này chủ yếu làm nổi bật các yếu tố của việc phân định cơ cấu các công việc chi tiết (các yếu tố WBS) về phần mềm, khi mà phần mềm là cả một dự án hay chỉ đơn giản là một thành phần của một hệ thống lớn.

10.1.1. Kết quả của WBS theo quy ước

Định hướng công việc chi tiết theo quy ước thường xuyên chịu tác động xấu từ ba thiếu sót cơ bản sau:

1. Chúng được xây dựng một cách hấp tấp ngay từ khi sản phẩm được thiết kế.

2. Chúng được phân tích quá sớm, lập kế hoạch quá sớm, dự thảo ngân sách hoặc là quá thừa hoặc là quá thiếu chi tiết.

3. Chúng là các chỉ tiêu kỹ thuật của dự án, và các sự so sánh chéo giữa các dự án thường là khó khăn hoặc không thể thực hiện.

Định hướng công việc chi tiết theo quy ước được xây dựng quá sớm ngay từ khi thiết kế sản phẩm. Sơ đồ 10.1 chỉ ra việc phân định cơ cấu công việc chi tiết đặc trưng theo các quy ước mà đã được phân định một cách sơ sài với các hệ thống phụ trợ của cơ cấu sản phẩm của nó, tiếp đến là sự phân tích sâu hơn tạo thành các phần cụ thể của mỗi một hệ thống. Khi mà cơ cấu này được chấp nhận và sau đó được phân phối đến những nhà quản lý đầy trọng trách cùng với ngân sách, thời hạn tiến hành, các yêu cầu mong đợi và một nền tảng thiết lập kế hoạch cụ thể được cho rằng là rất tốn kém để thay đổi. Một bản WBS tạo dựng cơ cấu cho việc thiết lập kế hoạch tài chính. Bởi vì chỉ có các cấu trúc phần mềm cần được thâu tóm các bộ phận có biểu hiện thay đổi cho nên cần phải thiết lập trước các cơ cấu. Việc đối chiếu 2 bản kế hoạch một cách chặt chẽ để ấn định một cơ cấu sản phẩm có thể tạo ra sự chậm trễ nếu như cả 2 đều đòi hỏi sự cẩn thận chắc chắn. Tuy nhiên, việc đối chiếu một cách nới lỏng lại được ưa thích hơn nếu cả 2 kế hoạch hoặc cơ cấu là dễ thay đổi.

Việc phân định cơ cấu công việc chi tiết theo quy ước được phân tích, lên kế hoạch, phân bổ ngân sách quá sớm cũng như quá sơ sài hoặc quá chi tiết. Các dự án phần mềm lớn có xu hướng được thiết lập kế hoạch quá kỹ còn các dự án nhỏ thì lại có xu hướng thiết lập kế hoạch quá sơ sài. Báo cáo WBS chỉ ra ở Sơ đồ 10.1 dường như là đơn giản quá mức cho hầu hết các hệ thống ở quy mô lớn, thường có 6 hoặc hơn 6 cấp độ các yếu tố WBS. Đội ngũ quản lý thiết lập hoàn toàn đối với mỗi cấp độ cụ thể chi tiết và tạo ra các giới hạn chi phí và thời hạn cho mỗi nhiệm vụ ở những cấp độ cụ thể như nhau. Mặt khác, hầu hết các phát triển có quy mô nhỏ hay mang tính chất cá nhân đều thiết lập các báo cáo WBS của họ với một cấp duy nhất, không chi tiết cụ thể. Đội ngũ quản lý lên kế hoạch và thực hiện dự án với những nhiệm vụ lớn và việc giải trình về chi phí và thời hạn. Cả 2 phương thức tiếp cận đều phải chắc chắn. Nói chung, WBS bao gồm ít nhất 2 hoặc 3 cấp độ thì mới có ý nghĩa. Đối với hệ thống có quy mô lớn, việc thêm một số cấp độ là cần thiết ở những giai đoạn cuối của chu kỳ tồn tại của dự án. Vấn đề cơ bản của việc thiết lập kế hoạch quá chi tiết khi mới bắt đầu chính là các chi tiết đó không hề có liên quan gì với mức độ chính xác của kế hoạch. Ví dụ, không thể đưa ra một cách chính xác trong vòng 1 tháng (khi mà kế hoạch mới đang bước đầu được xây dựng và trước khi cơ cấu và các thử nghiệm những chuỗi sự kiện dự tính được thiết lập) các chi tiết cụ thể các hoạt động thử nghiệm được lên kế hoạch 18 tháng sau đó.

Việc phân định cơ cấu công việc chi tiết theo quy ước là diễn giải chi tiết dự án cụ thể và việc so sánh các dự án với nhau thường lại gặp rất nhiều khó khăn và có thể không tiến hành được. Phần lớn các tổ chức cho phép các dự án riêng rẽ được xác định cơ cấu dự án riêng theo cách riêng của các nhà quản lý, đáp ứng nhu cầu của khách hàng hoặc các yêu cầu riêng khác của dự án. Không có một tiêu chuẩn nào, thiết lập WBS thực sự gặp khó khăn trong việc so

sánh các kế hoạch, số liệu tài chính, thời hạn, năng lực của tổ chức, các xu hướng chi tiêu, hiệu suất hay chất lượng của nhiều dự án khác nhau. Mỗi dự án tổ chức công việc một cách khác nhau và sử dụng các kỹ thuật tính toán khác nhau. Một vài câu hỏi ví dụ dưới đây là khó đối với bất cứ một chương trình phát triển được một tổ chức nào đó thực hiện, không thể trả lời bởi hầu hết các đội ngũ thực hiện dự án mà sử dụng việc phân định cơ cấu công việc chi tiết theo quy ước:

- Tỷ số giữa các hoạt động sản xuất (các yêu cầu, thiết kế, sự thực hiện, đánh giá, triển khai) là bao nhiêu để trang trải cho các hoạt động (quản lý, tạo môi trường)?
- Một tỷ lệ cố gắng chi tiêu hợp lý cho các hoạt động lặp lại là bao nhiêu?
- Một tỷ lệ chi tiêu cho các thiết bị phần mềm là bao nhiêu (các chi phí cho việc bảo vệ môi trường)?
- Một tỷ số giữa hoạt động thử nghiệm sản xuất với quá trình lặp (không trong giai đoạn sản xuất) là bao nhiêu?
- Chi phí cho sản phẩm thứ N (dựa trên cơ sở kế hoạch cho sản phẩm thứ N+1) là bao nhiêu?

Sơ đồ 10.1 *. Việc phân định cơ cấu công việc chi tiết theo quy ước, liệt kê theo mô hình phân cấp sản phẩm*

Quản lý

Các yêu cầu và thiết kế hệ thống

Hệ thống phụ trợ 1

-Thành phần 1-1

+ Các yêu cầu

+ Thiết kế

+ Mã chương trình

+ Thử nghiệm

+ Cung cấp tài liệu

... (Cấu trúc tương tự cho các thành phần khác)

- Thành phần 1-N

+ Các yêu cầu

+ Thiết kế

+ Mã chương trình

+ Thử nghiệm

+ Cung cấp tài liệu

... (Cấu trúc tương tự cho các hệ thống phụ trợ khác)

Hệ thống phụ trợ thứ M

- Thành phần M-1
    + Các yêu cầu
    + Thiết kế
  + Mã chương trình
  + Thử nghiệm
  + Cung cấp tài liệu

... (Cấu trúc tương tự cho các thành phần khác)

- Thành phần M-N
    + Các yêu cầu
    + Thiết kế
  + Mã chương trình
  + Thử nghiệm
  + Cung cấp tài liệu

Thực hiện vòng lặp và thử nghiệm

Lập kế hoạch thử nghiệm

Chuẩn bị các thủ tục thử nghiệm

Tiến hành thử nghiệm

Báo cáo kết quả thử nghiệm

Một số phạm vi quan trọng khác

Điều khiển cấu hình

Đảm bảo chất lượng

áp dụng hệ thống

10.1.2 Việc phân định cơ cấu công việc chi tiết hiện đại

Một báo cáo WBS (Work breakdown structure) hiện đại thiết lập nên các yếu tố kế hoạch cho khuôn khổ cả quá trình thực hiện hơn là chỉ riêng quá trình sản xuất. Cách tiếp cận này xử lý các thay đổi đã được dự đoán tốt hơn và cho phép mức độ chính xác của việc lên kế hoạch để suy luận theo cách dễ dàng. Việc đưa ra nền tảng cho WBS là để tạo ra các hệ thống phân cấp các yếu tố như sau:

- Các yếu tố WBS cấp 1 là các nhóm yếu tố chính (quản trị, thiết lập môi trường, các yêu cầu, thiết kế, sự thực hiện, đánh giá và triển khai). Những yếu tố này thường được giao cho một đội ngũ đơn lẻ (sẽ được đề cập đến ở chương 11) và tạo thành kỹ thuật phân tích của một dự án theo mục đích của việc thiết lập kế hoạch và so sánh với những dự án khác.

- Các yếu tố ở cấp 2 được xác định cho mỗi giai đoạn trong chu kỳ tồn tại của dự án (khởi đầu, nghiên cứu thử nghiệm, triển khai xây dựng và chuyển giao). Các yếu tố này đưa ra một mức độ chính xác của dự án cho phép suy luận một cách tất yếu hơn các mức độ nắm bắt về yêu cầu đòi hỏi, cơ cấu và các rủi ro chứa đựng bên trong.
- Các yếu tố cấp 3 được xác định tập trung vào các hoạt động sản xuất mẫu vật trong mỗi giai đoạn. Các yếu tố này có thể ở cấp thấp nhất trong hệ thống phân cấp các yếu tố WBS, tập hợp chi phí các sản phẩm rời rạc trong giai đoạn nhất định, hoặc chúng được phân tích một cách sâu hơn với nhiều hoạt động ở cấp độ nhỏ hơn, được tiến hành cùng lúc, sản xuất ra sản phẩm đơn lẻ.

Sự nhất quán được ngầm định trong WBS với cấu trúc của cả quá trình (bao gồm các giai đoạn, các nhóm yếu tố và các sản phẩm) được chỉ ra ở Sơ đồ 10.2 . Từ đó đề nghị một cấu trúc hay là một ví dụ về vấn đề làm như thế nào để các yếu tố được tham gia quá trình lặp trong dự án. Nó cung cấp một cấu trúc cho việc dự tính chi phí, thời hạn của mỗi một yếu tố, phân bổ chúng cho toàn bộ tổ chức thực hiện dự án và kiểm soát các khoản chi tiêu.

Sơ đồ 10.2 *Việc phân định các công việc chi tiết ngầm định*

A Quản lý

AA Quản lý ở giai đoạn khởi đầu

AAA Phát triển các tình huống kinh doanh

AAB Xác định chi tiết kết quả cho giai đoạn nghiên cứu thử nghiệm

AAC Đặt nền tảng WBS cho giai đoạn nghiên cứu thử nghiệm

AAD Thiết lập kế hoạch phát triển phần mềm

AAE Kiểm soát dự án ở giai đoạn khởi đầu và đánh giá hiện trạng

AB Quản lý ở giai đoạn nghiên cứu thử nghiệm

ABA Xác định chi tiết kết quả cho giai đoạn xây dựng triển khai

ABB Đặt nền tảng WBS cho giai đoạn xây dựng triển khai

ABC Kiểm soát dự án ở giai đoạn nghiên cứu thử nghiệm và đánh giá hiện trạng

AC Quản lý ở giai đoạn xây dựng triển khai

ACA Lên kế hoạch cho giai đoạn triển khai

ACB Đặt nền tảng WBS cho giai đoạn triển khai

ACC Kiểm soát dự án ở giai đoạn xây dựng triển khai và đánh giá hiện trạng

AD Quản lý ở giai đoạn chuyển giao

ADA Lên kế hoạch cho giai đoạn tiếp theo

ADB Kiểm soát dự án ở giai đoạn chuyển giao và đánh giá hiện trạng

B Tạo dựng môi trường

BA Xác định chi tiết môi trường giai đoạn khởi đầu

BB Đặt nền tảng môi trường cho giai đoạn nghiên cứu thử nghiệm

BBA Phát triển môi trường thiết bị kỹ thuật và môi trường hành chính

BBB Phát triển môi trường thống nhất và phục vị khách hàng

BBC Thiết lập hệ thống dữ liệu SCO

BC Duy trì môi trường trong giai đoạn xây dựng triển khai

BCA Phát triển môi trường thiết bị kỹ thuật và môi trường hành chính

BCB Duy trì dữ liệu SCO

BD Duy trì môi trường trong giai đoạn chuyển giao

BDA Phát triển môi trường duy trì và môi trường hành chính

BDB Duy trì dữ liệu SCO

BDC Duy trì môi trường đóng kiện và chuyển giao

C Những yêu cầu

CA Phát triển những yêu cầu trong giai đoạn khởi đầu

CAA Xác định chi tiết các quan điểm

CAB Sử dụng các mô hình tình huống

CB Đặt nền tảng yêu cầu cho giai đoạn nghiên cứu thử nghiệm

CBA Thiết lập nền tảng các quan điểm

CBB Sử dụng nền tảng các mô hình tình huống

CC Duy trì các yêu cầu cho giai đoạn xây dựng triển khai

CD Duy trì các yêu cầu cho giai đoạn chuyển giao

D Thiết kế

DAThiết lập cơ cấu mẫu vật cho giai đoạn khởi đầu

DB Thiết lập cơ cấu nền tảng cho giai đoạn nghiên cứu thử nghiệm

DBA Sử dụng mô hình thiết kế cấu trúc

DBB Lên kế hoạch và thực hiện các thiết kế

DBC Miêu tả cấu trúc phần mềm

DC Lập mô hình thiết kế cho giai đoạn xây dựng triển khai

DCA Duy trì mô hình cấu trúc thiết kế

DCB Lập mô hình thiết kế bộ phận

DD Duy trì thiết kế trong giai đoạn chuyển giao

E Thực hiện

EA Mẫu vật ở giai đoạn khởi đầu

EB Thực hiện từng phần ở giai đoạn nghiên cứu thử nghiệm

EBA Các vòng lặp giải mã bộ phận quan trọng

EC Thực hiện từng phần ở giai đoạn xây dựng triển khai

ECA Mã hiệu bộ phận kết quả ban đầu và những thử nghiệm đơn lẻ

ECB Mã hiệu bộ phận kết quả lần đưa ra thứ nhất và thử nghiệm đơn lẻ

ECC Mã hiệu bộ phận kết quả lần đưa ra thứ hai và thử nghiệm đơn lẻ

ECD Duy trì bộ phận

ED Duy trì bộ phận trong giai đoạn chuyển giao

F Đánh giá

FA Lên kế hoạch đánh giá giai đoạn khởi đầu

FB Đánh giá trong giai đoạn nghiên cứu thử nghiệm

FBA Mô hình kiểm nghiệm

FBB Thực hiện một chuỗi các kiểm nghiệm cấu trúc

FBC Đánh giá các biểu hiện và miêu tả kết quả

FC Đánh giá giai đoạn xây dựng triển khai

FCA Đánh giá kết quả ban đầu và miêu tả kết quả

FCB Đánh giá kết quả đưa ra lần thứ nhất và miêu tả kết quả

FCC Đánh giá kết quả đưa ra lần thứ hai và miêu tả kết quả

FD Đánh giá giai đoạn chuyển giao

FDA Đánh giá kết quả sản phẩm và miêu tả kết quả

G Triển khai

GA Lên kế hoạch triển khai giai đoạn khởi đầu

GB Lên kế hoạch triển khai giai đoạn nghiên cứu thử nghiệm

GC Thực hiện giai đoạn xây dựng triển khai

GCA Thực hiện dựa theo nhu cầu người sử dụng

GD Triển khai giai đoạn chuyển giao

GDA Chuyển giao sản phẩm tới người tiêu dùng

Cơ cấu được chỉ ra có xu hướng đơn thuần đi theo một điểm khởi đầu. Nó cần phải được điều chỉnh để xác định chi tiết dự án theo nhiều cách khác nhau.

- Quy mô: Các dự án có quy mô lớn sẽ có nhiều cấp và nhiều hệ thống phụ trợ hơn.

- Cơ cấu của tổ chức: Những dự án mà bao gồm nhiều thầu phụ và nhiều thực thể trong tổ chức có thể bao gồm những đòi hỏi bắt buộc yêu cầu những sự phân bổ WBS khác nhau.
- Mức độ đáp ứng yêu cầu từ phía khách hàng: Dựa trên các tính chất của dự án, các yêu cầu, thiết kế, các nhóm yếu tố thực hiện có thể có tầm quan trọng khác nhau. Một dự án kinh doanh trong giai đoạn thiết kế lại thường chủ yếu dựa trên các bộ phận hiện hành thì có nhiều đòi hỏi hơn đối với nhóm các yêu cầu, phần thiết kế vẫn còn khá sơ sài và phần thực hiện. Sự đáp ứng khách hàng một cách đầy đủ của một trong những kiểu phân bổ kỹ thuật có thể đòi hỏi phần thiết kế và thực hiện khá chi tiết để vượt qua các rủi ro liên quan tới khách hàng-bộ phận được xếp thứ hạng trên cùng.
- Nội dung kinh doanh: Các dự án có hợp đồng đòi hỏi nhiều hơn các yếu tố quản lý và đánh giá phức tạp. Các dự án sản xuất ra các sản phẩm thương mại phân phối cho nhiều khách hàng có thể yêu cầu nhiều hệ thống phụ trợ phức tạp cho việc triển khai thực hiện các yếu tố. Một sự ứng dụng khi được triển khai đơn lẻ có thể là yếu tố ít quan trọng (như là sự ứng dụng triển khai kinh doanh nội bộ) hoặc một yếu tố phức tạp (ví dụ như chuyển giao từ hệ thống kế thừa các nhiệm vụ quan trọng cùng với hoạt động đồng thời sang trạng thái ngưng trệ hoạt động của máy móc).
- Các kinh nghiệm đã có: Hầu như không có dự án nào bắt đầu mà không có một chút kinh nghiệm nào cả. Hầu hết chúng là một thế hệ mới nằm trong một hệ thống được kế thừa nhiều kinh nghiệm (với một báo cáo WBS được lập cẩn thận) hoặc nằm trong nội dung của các chuẩn mực hiện hành trong tổ chức (với các dự tính WBS được xác định trước). Việc đáp ứng các đòi hỏi bắt buộc là rất quan trọng nhằm đảm bảo rằng dự án mới được kế thừa những kinh nghiệm đã có và có được tiêu chuẩn cho hoạt động dự án.

Báo cáo WBS phân tích các đặc tính của dự án và kết hợp chúng với chu kỳ tồn tại, ngân sách, và vấn đề nhân sự. Trong quá trình hoạt động đánh giá dự án và quản lý phần mềm, kiểm toán nhiều năm qua, tôi đã nhận thấy rằng WBS là nguồn thông tin có giá trị nhất về việc thiết lập kế hoạch dự án. Trong khi kế hoạch phát triển phần mềm và các tình huống kinh doanh cung cấp nội dung để cân nhắc thì WBS và ngân sách có liên quan được phân bổ cùng với các nhân tố đã đưa ra các chỉ dẫn có ý nghĩa đầy đủ nhất cho phương pháp quản lý, các trường hợp ưu tiên và các vấn đề khác có liên quan.

Một thuộc tính quan trọng khác của một báo cáo WBS có chất lượng là mức độ chính xác của mỗi yếu tố khi lập kế hoạch có thích hợp với giai đoạn hiện hành trong vòng đời và trạng thái của dự án. Sơ đồ 10.3 sẽ chỉ ra điều này. Một trong những lý do chính cho việc thực hiện phương pháp WBS ngầm định mà tôi dưa ra là nó cho phép xây dựng các yếu tố bắt đầu từ việc lên kê hoạch cho khâu cuối (ngân sách ước tính qua vẫn chỉ là sư dự đoán suy luận sâu hơn là so với phân tích chi tiết) sau khi có những mạng lưới các hoạt động đã được lên kế hoạch trước

một cách đầy đủ (với ngân sách được xác định rõ ràng chắc chắn và sự đánh giá liên tục về các chi tiêu thực tế so với ước tính).

10.2.Các nguyên tắc lập kế hoạch

Các dự án phần mềm trải rộng lên các lĩnh vực ứng dụng khác nhau. Sẽ rất có ích nhưng cũng đầy rủi ro khi đưa ra một vài lời khuyên cho việc lập kế hoạch cụ thể tách rời khỏi môi trường dự án. Việc làm trên có ích vì hầu hết những người quản lý trước tiên đều tìm kiếm một kế hoạch sơ bộ để từ đó họ có thể xây dựng các chi tiết cụ thể của dự án. Họ hiểu rằng các nguyên tắc lập kế hoạch ban đầu thu hút sự giám định và kinh nghiệm của nhiều người khác. Do đó các nguyên tắc được xem như là những nền tảng đáng tin cậy của sự đánh giá và sự tin tưởng của các chủ đầu tư dự án.

Những lời khuyên cho việc lập kế hoạch tách biệt với dự án cũng chứa đầy rủi ro. Do những nguyên tắc này có thể được thông qua một cách mù quáng mà không được làm cho phù hợp với trường hợp dự án cụ thể. Mù quáng làm theo lời khuyên của người khác về việc lập kế hoạch tách biệt với dự án là một dấu hiệu chắc chắn là đội ngũ điều hành thiếu năng lực. Ngoài ra còn phải kể đến sự nguy hiểm nếu hiểu sai lời khuyên của người khác. Sự khác biệt của các thông số của dự án, của phạm vi công việc của dự án, của môi trường văn hoá, và các tiến trình dự án rất dễ tạo ra sai lầm có tác động nghiêm trọng không lường trước. Trong cuốn sách này, tôi đã cố gắng đưa ra một tình huống đầy đủ mà có khả năng tránh được những sai sót như vậy. Nhằm giảm các cuộc tranh luận về vấn đề này, Phụ lục D sẽ đưa ra chính một trường hợp nghiên cứu cụ thể của một dự án đã là hiện thực. Trường hợp này đưa ra một ví dụ hữu ích về một dự án mà có tới 90% là phù hợp với những nguyên tắc lập kế hoạch tách biệt với dự án ở đây. Nó cũng đưa ra những ví dụ và những lời giải thích hợp lý đối với một số sai lệch nhỏ so với những nguyên tắc này.

(1) Khởi đầu		**(2) Nghiên cứu thử nghiệm**	
Thành phần của WBS	**Độ chính xác**	**Thành phần của WBS**	**Độ chính xác**
Quản lý	***Cao***	***Quản lý***	***Cao***
Môi trường	Vừa phải	***Môi trường***	***Cao***
Các yêu cầu	***Cao***	***Các yêu cầu***	***Cao***
Thiết kế	Vừa phải	***Thiết kế***	***Cao***
Thực hiện	Thấp	Thực hiện	Vừa phải
Đánh giá	Thấp	Đánh giá	Vừa phải

Triển khai	Thấp	Triển khai	Thấp
Thành phần của WBS	**Độ chính xác**	**Thành phần của WBS**	**Độ chính xác**
Quản lý	***Cao***	***Quản lý***	***Cao***
Môi trường	Cao	***Môi trường***	***Cao***
Các yêu cầu	Thấp	Các yêu cầu	Thấp
Thiết kế	Thấp	Thiết kế	Vừa phải
Thực hiện	Vừa phải	***Thực hiện***	***Cao***
Đánh giá	***Cao***	***Đánh giá***	***Cao***
Triển khai	***Cao***	Triển khai	Vừa phải
(3) Chuyển giao		**(4) Triển khai xây dựng**	

Sơ đồ 10-3*. Tiến triển của việc lập kế hoạch đúng đắn trong WBS thông qua vòng đời sản phẩm*

Có hai nguyên tắc lập kế hoạch đơn giản nên được xem xét khi một kế hoạch dự án được khởi đầu hay đánh giá. Nguyên tắc thứ nhất, cụ thể nêu trong bảng 10-1, quy định một sự phân bổ chi phí mặc định giữa các thành phần của WBS cấp 1. Nguyên tắc thứ hai, cụ thể nêu trong bảng 10-2, quy định sự phân phối trách nhiệm và thời hạn trong các giai đoạn của vòng đời tồn tại của dự án. Đưa ra sự ước lượng ban đầu về tổng chi phí dự án và hai bảng trên, việc xây dựng một bảng mô tả sơ lược nhân sự, việc bố trí nguồn nhân lực thành các nhóm, xác định thời hạn lâu nhất cho phép của dự án, và báo cáo WBS ban đầu với ngân sách và thời hạn là tương đối dễ dàng. Phương pháp xây dựng kế hoạch “từ trên xuống” là một cách lập kế hoạch có ích mà sẽ dẫn đến cơ sở nền tảng cho các suy luận sâu hơn.

Làm sao để có được dữ liệu trong Bảng 10-1 và Bảng 10-2? Đáng tiếc, đó không phải là số liệu có từ một số các tình huống nghiên cứu được cung cấp đầy đủ dữ liệu cần thiết của một số dự án đã thành công, theo đó là một quá trình phát triển phần mềm hiện đại. Các số liệu này được lấy phần lớn là từ kinh nghiệm của tôi, bao gồm cả nỗ lực dự tính chi phí phần mềm trong suốt một thập kỷ qua với quy mô rộng bao gồm hàng loạt dự án phần mềm, các tổ chức, các quá trình và các kỹ thuật.

Bảng 10-1. Hoạch định ngân quỹ cho WBS

Thành phần WBS cấp một	Ngân quỹ ngầm định
Quản lý	10%
Môi trường	10%
Các yêu cầu	10%
Thiết kế	15%
Thực hiện	25%
Đánh giá	25%
Triển khai	5%
Tổng số	100%

Bảng 10.1 đưa ra những sự phân bổ chi tiêu ngân sách ngầm định đối với mỗi yếu tố WBS cấp 1. Trong khi những giá trị này chắc chắn tác động tới toàn bộ các dự án, cách phân bổ này đã cung cấp một tiêu chuẩn tốt cho việc đánh giá kế hoạch thông qua cách nắm bắt lý do cho việc sai khác so với những nguyên tắc này. Một điểm quan trọng ở đây là vấn đề phân bổ chi phí, chứ không phải là vấn đề phân bổ trách nhiệm. Để tránh sai sót thông số, cần xem xét 2 luận điểm dưới đây:

1. Chi phí của danh mục các công việc khác nhau phải hợp lý với các số liệu. Ví dụ như các yếu tố quản lý, các yêu cầu và thiết kế có xu hướng sử dụng nhiều người có thâm niên cao và trả lương cao hơn so với các yếu tố khác. Nếu yếu tố yêu cầu và thiết kế chiếm tới 25% ngân sách (thuê người với mức lương trung bình là 100$/giờ) thì tổng số chi này sẽ tương đương một nửa tổng số giờ nhân công của yếu tố đánh giá-chiếm 25% ngân sách nhưng chỉ chi có mức lương là 50$/giờ.
2. Chi phí của những tài sản phần cứng và phần mềm cho quá trình tự động hoá và phát triển đội ngũ là bao gồm cả chi phí cho yếu tố môi trường.

Bảng 10-2*. Phân phối ngầm định trách nhiệm và thời hạn cho mỗi giai đoạn*

Phạm vi	Giai đoạn khởi đầu	Giai đoạn nghiên cứu thử nghiệm	Giai đoạn triển khai xây dựng	Giai đoạn chuyển giao
Trách nhiệm	5%	20%	65%	10%
Thời hạn	10%	30%	50%	10%

Bảng 10.2 đưa ra những nguyên tắc cho việc phân bổ trách nhiệm và thời hạn trong tất cả các giai đoạn vòng đời của dự án. Mặc dù những con số này có thể có biến đổi nhiều, tuỳ theo những yêu cầu bắt buộc của việc ứng dụng, chúng vẫn đưa ra mức mong đợi trung bình trên toàn bộ phạm vi ứng dụng. Đạt được sự chắc chắn khi sử dụng những giá trị cụ thể không phải là điều quan trọng bằng việc hiểu được tại sao mà dự án của bạn lại có thể có những con số khác so với những con số trên.

10.3.Quá trình ước tính về chi phí và lịch trình của dự án

Những kế hoạch của dự án cần phải được xây dựng theo 2 hướng.

Thứ nhất là cách tiếp cận "từ trên xuống" và dự tính cho tương lai. Đầu tiên, dựa vào những lịch trình và chi tiêu tổng thể để nắm bắt những yêu cầu, đòi hỏi bắt buộc chung. Sau đó thì tiến hành phân tích những yếu tố này thành những khoản chi tiêu và ở các mốc thời điểm cụ thể hơn. Theo cách thức này, trình tự xây dựng kế hoạch sẽ diễn ra như sau:

1. Nhà quản lý dự án về phần mềm (hoặc ai khác) tiến hành xây dựng bảng mô tả đặc điểm về toàn bộ quy mô, cách thức, điều kiện, nhân lực và đặc tính cần thiết cho dự án.

2. Dựa vào mô hình ước tính chi phí phần mềm, xây dựng các ước tính tổng thể về kết quả cuối cùng và thời hạn thực hiện.

3. Nhà quản lý dự án phần mềm tách riêng phần dự tính về kết quả trong WBS ở mức cao nhất theo các nguyên tắc như ở bảng 10.1. Đồng thời, các ước tính về thời hạn cũng được tách riêng và phân tích thành các mốc thời gian chính. Kết quả được thể hiện trong bảng mô tả sơ lược nhân lực theo các chỉ dẫn như trong bảng 10.2. Kết thúc công việc trên, ta mới chỉ có kế hoạch ở tầm tổng thể. Các ước tính như vậy thường có xu hướng bỏ qua nhiều thông số chi tiết cụ thể của dự án.

4. Đến thời điểm này, dựa trên những báo cáo phân bổ chi phí ở mức cao nhất có thể, bảng mô tả sơ lược về nhân sự và các mốc thời hạn chính theo những quy định bắt buộc, những nhà quản lý dự án ở cấp thấp hơn có trách nhiệm phân tích từng yếu tố trong các báo cáo WBS ở mức độ cụ thể hơn.

Hướng thứ hai là cách tiếp cận "từ dưới lên" và dựa vào các số liệu đã có. Có nghĩa là công việc xây dựng kế hoạch sẽ bắt đầu với việc phân tích các chi phí và thời hạn chi tiết, sau đó tổng hợp lại thành chi phí cho các khoản mục tổng thể và các mốc thời hạn trung gian. Cách tiếp cận này có xu hướng chỉ rõ và ấn định những báo cáo WBS qua việc tổng hợp từ những mức thấp nhất lên. Theo cách tiếp cận này, tiền trình xây dựng kế hoạch diễn ra như sau:

1. Các yếu tố WBS ở cấp độ thấp nhất được xem xét kỹ lưỡng thành các công việc chi tiết, mà các chi phí và thời hạn của nó đã được ước tính bởi nhà quản lý phụ trách nghiên cứu về các yếu tố WBS. Các ước tính này có xu hướng kết hợp chặt chẽ các thông số cụ thể của dự án theo cách quá phóng đại.

2. Các số liệu ước tính được kết nối và tổng hợp lại thành chi phí các khoản mục, các mốc thời gian tổng thể hơn. Thành kiến suy nghĩ của mỗi một nhà phân tích cần phải thống nhất để có được một nền tảng lý luận vững chắc.

3. So sánh với các chi phí và thời hạn được ước tính theo cách "từ trên xuống". Đưa ra những sự khác nhau nói chung và điều chỉnh để có sự thống nhất giữa cách ước tính "từ trên xuống" và "từ dưới lên"

Lập các mốc thời hạn hay phân bổ chi phí bằng việc ước tính "từ trên xuống" có xu hướng phóng đại các khuynh hướng suy nghĩ khác nhau trong việc quản lý dự án và thường mang đến kết quả là các kế hoạch lạc quan qua mức. Còn theo cách ước tính "từ dưới lên" thì lại thường đề cao các thành kiến của những người thi hành công việc và kết quả là các kế hoạch lại bi quan qua mức. Sử dụng các kết quả theo cách tiếp cận này, chấp nhận và chọn ra các kết quả theo cách tiếp cận còn lại, quá trình lặp lại này là thực sự cần thiết. Để từ đó, rút ra một bản kế hoạch cuối cùng sau quá trình kết hợp các quan điểm. Quá trình này tạo ra được sự tham gia vào việc xây dựng kế hoạch của tất cả các cấp quản lý.

Hai cách tiếp cận xây dựng kế hoạch trên đây nên được cùng sử dụng một cách cân đối hợp lý cho suốt chu kỳ tồn tại của dự án. Trong giai đoạn thiết kế, khuynh hướng "từ trên xuống" sẽ mang tính chỉ đạo bởi vì thường là không có đủ sự hiểu biết chuyên sâu cũng như chắc chắn về trình tự các công việc chi tiết để tiến hành xây dựng kế hoạch theo cách "từ dưới lên" mà lại có thể tin tưởng được. Còn trong giai đoạn đi vào hoạt động, do đã có đủ kinh nghiệm trước đó và sự chính xác trong việc xây dựng kế hoạch nên cách tiếp cận "từ dưới lên" sẽ lại là chủ đạo. Có như vậy, cách phân tích "từ trên xuống" mới có được các thông số chi tiết về dự án (mà bản thân nó không thể có được), nhờ đó mà nó đã trở thành một kỹ thuật đánh giá phân tích toàn diện. Các con số ở bảng 10.4 minh hoạ sự đối trọng trong việc sử dụng 2 cách thức để xây dựng chu kỳ tồn tại của dự án.

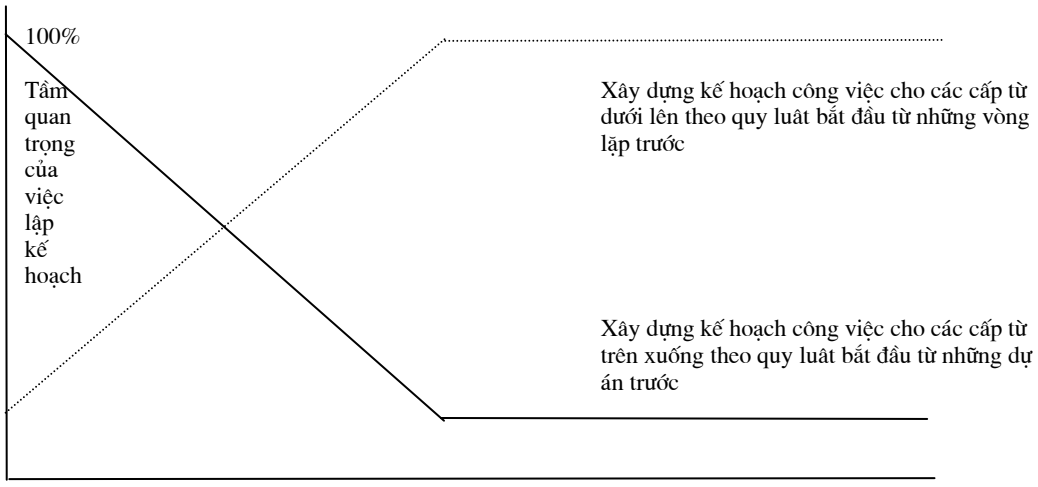
10.4.Quá trình xây dựng kế hoạch lặp, kéo dài vòng chu kỳ của dự án

Cho đến nay,vấn đề này mới chỉ xem xét việc thiết lập chi phí và phân bổ thời gian đối với các ứng dụng riêng biệt. Một vấn đề khác liên quan đến việc xây dựng kế hoạch là xác định rõ trình tự thực sự của những kết quả trung gian. Xây dựng nội dung và lịch trình của những mốc mục tiêu chính và các quá trình lặp lại trung gian có thể là hình thức hiển nhiên đối với kế hoạch quản lý toàn bộ mọi rủi ro. Đối với một kế hoạch được xây dựng, tính suy luận là rất cần thiết bởi vì luôn có những sự điều chỉnh trong việc xây dựng nội dung và thời hạn của dự án sao cho những sự phỏng đoán được rút ra sẽ trở thành những tình huống có khả năng nắm bắt tốt nhất.

Những đặc điểm chung quá trình tiến hành và những nguyên tắc chung về số lần lặp lại trong mỗi giai đoạn được thể hiện ở giai đoạn tiếp theo. Sự lặp lại ở đây có nghĩa là sự đồng bộ hoá hoàn toàn trong suốt dự án, với sự đánh giá một cách toàn diện, sắp đặt hợp lý về cơ sở của tổng thể dự án. Các quá trình lặp lại khác ở cấp thấp hơn, ví dụ như hàng tháng, hàng tuần, hay hàng ngày, được thực hiện theo một lộ trình gồm các mốc đồng bộ hoá các mức độ của dự án.

- Giai đoạn thử nghiệm: Các hoạt động thử nghiệm mẫu đầu tiên hợp thành các bộ phận nền tảng của một cấu trúc được đề ra và cung cấp một cơ chế hoạt động có tính khả thi cho việc soạn thảo kỹ lưỡng các tình huống sử dụng đột xuất của hệ thống. Cơ chế hoạt động này bao gồm cả những phần hiện hành, những bộ phận thương mại và các khách hàng thử nghiệm, đủ để thấy rõ về cơ cấu được đề ra và với các hiểu biết đầy đủ về các yêu cầu để có thể thiết lập một tình huống kinh doanh có thể có thực, một quan điểm, một kế hoạch phát triển phần mềm. ở một phạm vi lớn hơn, tăng khách hàng có thể đòi hỏi 2 sự lặp lại để đạt được một hình mẫu có thể chấp nhận được, nhưng phần lớn các dự án lại chỉ nên tiến hành 1 mà thôi.
- Giai đoạn nghiên cứu chi tiết: Các quá trình lặp lại ở đây nhằm đi đến một cơ trúc, bao gồm cả một khuôn khổ hoạt động đầy đủ và một hạ tầng cơ sở cho việc thực hiện. Ngay cả khi có sự hoàn thành một cơ cấu như vậy, một vài tình huống đột xuất vẫn cần được chỉ ra: (1) việc nạp dữ liệu ban đầu cho cơ cấu, (2) sự xen ngang vào chuỗi dữ liệu dẫn đến việc xử lý dữ liệu rơi vào tình huống xấu nhất được truyền qua hệ thống (ví dụ khi giao dịch lên tới tột đỉnh số lượng dữ liệu đưa vào hoặc chuỗi dữ liệu được truyền tải là tối đa), và (3) sự xen ngang chuỗi dữ liệu sẽ tạo nên sự kiểm soát kém nhất qua hệ thống (ví dụ như sự sắp đặt các tình huống sử dụng có dung sai bị hỏng. Hầu hết các dự án nên xây dựng trên 2 vòng lặp lại để có được một cơ cấu hợp lý. Những cơ cấu nào mà chưa từng có thì có thể đòi hỏi nhiều sự lặp lại hơn, trong khi các dự án được xây dựng dựa trên một cơ chế đã được thiết lập hoàn hảo thì có thể chỉ cần duy nhất một sự lặp lại mà thôi.
- Giai đoạn triển khai xây dựng: Phần lớn các dự án đòi hỏi ít nhất 2 vòng lặp xây dựng chính : đưa ra sản phẩm lần thứ nhất và đưa ra sản phẩm lần thứ hai. Lần thứ nhất đưa ra sẽ phải có được khả năng thực hiện trong tất cả mọi trường hợp sử dụng đột xuất. Nó thường chỉ đại diện khoảng 70% trên tổng sản lượng theo sự phát triển chiều rộng và hoạt động ở nhiều mức chất lượng (cả về hiệu suất và độ tin cậy) dưới mức mong đợi ở sản phẩm cuối cùng. Lần ra mắt thứ hai, như trở thành một đặc trưng tiêu biểu, cung cấp 95% tổng sản lượng tiềm năng và đạt một vài thuộc tính về chất lượng quan trọng. Dù là tiêu biểu, tuy nhiên, một vài đặc điểm nữa vẫn cần được hoàn thiện và sự nâng cao hơn nữa tính tinh xảo và hiệu suất cũng rất cần thiết đối với sản phẩm cuối cùng để được chấp nhận. Mặc dù hầu hết các dự án cần ít nhất 2 vòng lặp xây dựng, nhưng có rất nhiều lý do để tiến hành thêm 1 hoặc nhiều hơn nữa vòng lặp để xử lý được các rủi ro và để có được cái nhìn lạc quan về các chi phí bỏ ra là hiệu quả.
- Giai đoạn chuyển giao: Hầu hết các dự án đều sử dụng 1 vòng lặp duy nhất để chuyển từ sản phẩm đưa ra lần thứ hai đến sản phẩm cuối cùng. Một lần nữa, một số vòng lặp quy mô nhỏ, không chính thức có thể là cần thiết để giải quyết các khiếm khuyết, kết hợp chặt chẽ với các thông tin phản hồi từ lần đưa ra thứ hai và các cải

tiến trong quá trình tiến hành. Tuy vậy, bởi vì tổng chi phí liên quan đến việc chuyển giao ở quy mô lớn tới công chúng sử dụng nên hầu hết các dự án thường có kinh nghiệm là chỉ nên tiến hành với 1 vòng lặp từ khi lần đưa ra sản phẩm lần thứ hai đến khi đưa ra sản phẩm cuối cùng.



Quá trình thiết kế		Quá trình triển khai	
Khởi đầu	Nghiên cứu chi tiết	Triển khai xây dựng	Chuyển giao

Các vòng lặp tính khả thi

Các vòng lặp cấu trúc

Các vòng lặp thích hợp cho việc đưa ra sản phẩm

Đưa ra sản phẩm

Tầm quan trọng của việc lập kế hoạch giai đoạn thiết kế:

- Xây dựng tổng thể các sản phẩm cho quá trình triển khai
- Xây dựng chi tiết các sản phẩm cho quá trình thiết kế

Tầm quan trọng của việc xây dựng giai đoạn triển khai:

- Xây dựng chi tiết các sản phẩm cho quá trình triển khai
- Xây dựng tổng thể các sản phẩm cho quá trình thiết kế

• Thống nhất ý kiến các chủ đầu tư dự án • Điều chỉnh các nguyên tắc xây dựng kế hoạch các dự án độc lập theo cách "từ trên xuống" thành các nguyên tắc xây dựng kế hoạch dự án đặc trưng • Xác định và nghiên cứu về WBS	• Thống nhất ý kiến các chủ đầu tư dự án • Phân tích chi tiết sự khác biệt giữa chi phí thực tế và trong kế hoạch

Sơ đồ 10.4. *Cân đối thông qua chu kỳ tồn tại*

Nguyên tắc chung ở đây là: nói chung hầu hết các dự án tiến hành khoảng 4 đến 5 vòng lặp. Các dự án đặc biệt có thể có tới 6 vòng lặp và nên theo chỉ dẫn sơ lược sau đây:

- 1vòng lặp trong giai đoạn khởi đầu: xác định mẫu cấu trúc
- 2 vòng lặp trong giai đoạn nghiên cứu thử nghiệm: xác định mẫu và cơ sở cho cấu trúc
- 2 vòng lặp trong giai đoạn triển khai xây dựng: đưa ra sản phẩm lần thứ nhất và lần thứ hai
- 1 vòng lặp trong giai đoạn chuyển giao: đưa ra sản phẩm cuối cùng

Các dự án có nhiều kinh nghiệm đi trước với một cơ cấu đã được xác định, hay là các dự án có quy mô thật nhỏ thì nên bỏ bớt đi 1 vòng lặp nối giữa giai đoạn khởi đầuvà giai đoạn nghiên cứu thử nghiệm và cũng có thể sản xuất ra sản phẩm hoàn hảo chỉ với tổng chi phí cho 4 vòng lặp. Một dự án có quy mô rất lớn và không có kinh nghiệm, có nhiều chủ sở hữu có thể cần đòi hỏi thêm 1 vòng lặp trong giai đoạn khởi đầu và thêm 2 vòng lặp trong giai đoạn triển khai xây dựng, thành tổng là 9 vòng lặp. Chi phí quản lý cuối cùng rất xứng đáng với giá cho việc chắc chắn về các rủi ro cũng như thống nhất các chủ đầu tư.

10.5 Thực hiện kế hoạch

Kể cả việc xây dựng kế hoạch tốt có nhiều động lực thúc đẩy nhờ vào quá trình lặp, thực hiện nó còn lâu mới là dễ dàng.Trong khi thực hiện vòng lặp thứ N ở bất cứ một giai đoạn nào, nhà quản lý dự án về phần mềm cần phải giám sát, kiểm soát kế hoạch bắt đầu từ vòng lặp thứ (N-1) và chuẩn bị cho vòng lặp thứ (N+1). Nghệ thuật quản lý dự án tốt là làm sao để có được sự cân bằng các yếu tố một cách hiệu quả nhất ở vòng lặp hiện tại và vòng lặp kế tiếp, căn cứ trên những kết quả đối với mục tiêu đã đề ra ở vòng lặp hiện tại và vòng lặp trước. Nội dung này, dường như và thực tế là, có ở trong tất cả các giai đoạn hay là ở trong các dự án có sự phát triển quá trình lặp

được đặt lên hàng đầu. Nhưng nếu như kế hoạch được đầu tư tiền của vào nhiều thì quá trình sẽ trở nên dễ dàng hơn, nhất là đối với giai đoạn cần có độ chính xác cao.

Ngoài cấu trúc sai lệch và việc không nắm bắt được những đòi hỏi, xây dựng kế hoạch không chính xác (tiếp theo là việc quản lý kém) là một trong số những nguyên nhân dễ gặp phải nhất làm phá hỏng dự án. Ngược lại, thành công của mỗi dự án có thể có được nhờ việc xây dựng kế hoạch tốt. Quyển sách này nhấn mạnh tầm quan trọng của 3 vấn đề: xây dựng kế hoạch, yêu cầu đòi hỏi và cấu trúc. Những sản phẩm cuối cùng có liên quan tới những vấn đề này (một kế hoạch phát triển phần mềm, việc xác định rõ các yêu cầu, và văn bản miêu tả cấu trúc) không phải là vấn đề chính. Trong hầu hết các dự án thành công, chúng thường không quan trọng bởi vì chúng đã từng được tiến hành. Chúng rất hiếm khi được những nhà điều hành sử dụng hàng ngày, chúng không hấp dẫn đối với những người sử dụng cuối cùng, và những giấy tờ trình bầy chúng thì chỉ là đỉnh chóp của một tảng băng trôi đối với những công việc cụ thể nằm ở bên dưới tảng băng đó.

Trong khi các văn bản kế hoạch không được hữa dụng cho lắm như một văn bản cuối cùng, vai trò của việc xây dựng kế hoạch lại thực sự quan trọng đối với thành công của dự án. Nó cung cấp một khuôn khổ và áp đặt các chức năng cho việc ra quyết định, đảm bảo việc chi tiền đối với chủ sở hữu dự án và người điều hành, và đồng thời chuyển giao các mục tiêu chung, tổng thể thành các mục tiêu cụ thể. Kế hoạch của một dự án sẽ là một chỉ dẫn làm như thế nào để các yêu cầu đòi hỏi đối với dự án được thực hiện trong sản phẩm nằm trong giới hạn các quy định kinh doanh. Nó bắt buộc phải thực tế, phải hiện hành, phải là sản phẩm của cả tập thể, phải được sự thông qua của các chủ đầu tư dự án, và nhất thiết phải được sử dụng.

Các kế hoạch không phải là chỉ để dành cho các nhà quản lý. Quá trình và kết quả xây dựng kế hoạch càng được mở rộng, rõ ràng nhiều bao nhiêu, thì giữa các thành viên thực thi càng tăng thêm tính tự chủ bấy nhiêu. Những kế hoạch tốt, thông thoáng có thể tạo ra môi trường văn hoá trong công việc từ đó khuyến khích đội ngũ những người lao động.

Chương 11

Tổ chức và chịu trách nhiệm dự án

Những điểm trọng tâm

- Cơ cấu tổ chức định hình lên cơ cấu của các đội.
- Các tổ chức kinh doanh trong lĩnh vực phần mềm cần hỗ trợ các dự án bằng cơ sở hạ tầng cần thiết để sử dụng một tiến trình chung.
- Các tổ chức dự án cần phân công trách nhiệm rõ ràng cho các đội làm dự án để đảm bảo tổng thể cân bằng và quan tâm đến tất cả các bộ phận.
- Tổ chức này phải tiến hành trình tự với WBS và quan tâm tới chu kỳ sống

Lĩnh vực kinh doanh phần mềm và đội ngũ làm dự án có những sự quan tâm khác nhau. Lĩnh vực kinh doanh phần mềm bị tác động bởi lợi nhuận trong đầu tư, các nhà kinh doanh có đầu óc kinh doanh mới, sự đa dạng của thị trường và khả năng sinh lời. Đội ngũ làm dự án bị tác động bởi chi phí, kế hoạch và chất lượng

Ngành kinh doanh phần mềm ở cả hai kiểu tổ chức đều được thúc đẩy bởi sự phát triển của ngành nghề, sự hài lòng trong công việc và cơ hội để tạo nên một sự khác biệt. Chủ đề này được bàn luận kỹ trong *A Discipline for Software Engineering* [Humphrey, 1995].

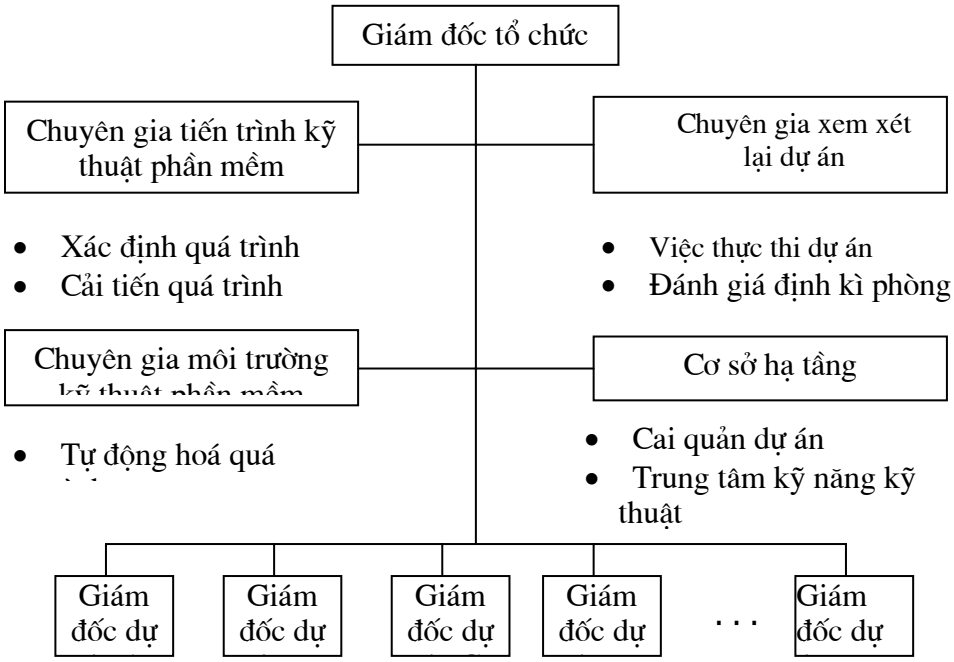
> *Trong thời gian trước đây, hầu hết những tư vấn về các cách tổ chức ở mức độ mà phần mềm được phát triển và lan rộng được tập trung vào dự án. Các dự án có những lợi ích riêng và sẽ không đầu tư cho bất kỳ ngành dịch vụ hoặc công nghệ nào không có ảnh hưởng trực tiếp tới chi phí, kế hoạch hay chất lượng của khả năng trình bày dự án. Chương này giới thiệu và mô tả các cách tổ chức một lĩnh vực kinh doanh và một dự án. Việc quy định hệ thống cấp bậc tổ chức rõ ràng là một trách nhiệm nguy hiểm trong điều kiện của từng người, từng tổ chức cụ thể. ở đây sẽ thảo luận về những vai trò chung, những mối liên hệ và các trách nhiệm. Đối với bất kỳ dự án hay lĩnh vực kinh doanh nào, những đề nghị đó chỉ là những điểm khởi động được cài đặt trước. Làm cho chúng thích hợp với lĩnh vực đó, với qui mô, văn hoá và các cá nhân trong từng hoàn cảnh cụ thể có thể đưa đến rất nhiều cách thực hiện khác nhau. Ví dụ, người ta có thể đánh giá cao việc tổ chức khác nhau một dự án hay đội ngũ như chia nhỏ hoặc hợp nhất những vai trò đã trình bày. Tuy nhiên, những đường lối chỉ đạo cách tổ chức đó gồm nhiều đề án lặp lại từ những dự án thành công và đưa ra một mô hình cho hầu hết các cách tổ chức.*

11.1.Tổ chức ngành kinh doanh

Hình 11.1 miêu tả các vai trò và trách nhiệm đối với công việc tổ chức một ngành kinh doanh được cài đặt trước. Cơ cấu này có thể thích hợp cho những hoàn cảnh cụ thể.

Những đặc trưng chính của việc tổ chức :

- Trách nhiệm xác định rõ và duy trì quy trình là rất cụ thể đối với một lĩnh vực kinh doanh mà ở đó tính đồng bộ của quá trình thực sự có ý nghĩa. Ví dụ, qui trình phát triển phần mềm kỹ thuật rất khác so với qui trình dùng để phát triển phần mềm ứng dụng văn phòng.
- Trách nhiệm tự động hoá quá trình là một nhiệm vụ tổ chức và cũng đóng vai trò quan trọng như nhiệm vụ xác định quá trình. Các dự án đạt được tính đồng bộ chủ yếu thông qua những công cụ hỗ trợ thông thường.
- Nhiệm vụ tổ chức có thể được thực hiện bởi một cá nhân đơn lẻ hoặc một số đội khác nhau phụ thuộc vào qui mô tổ chức. Một công ty sản xuất phần mềm có 20 nhân lực có thể chỉ cần 1 người để thực hiện toàn bộ những nhiệm vụ đó nhưng một công ty điện tử viễn thông có 10.000 nhân viên có thể cần hàng trăm người để đạt được một bộ máy tổ chức phần mềm hiệu quả



Hình 11.1 *Những vai trò định trước trong một tổ chức kinh doanh phần mềm*

Tổ chức quy trình kỹ thuật phần mềm (SEPA)

Chuyên gia tiến trình kỹ thuật phần mềm (SEPA- Software Engineering Process Authority) làm cho rễ ràng thay đổi thông tin và qui trình chỉ đạo cả hai cùng đến và xuất phát từ những người đang thực hành dự án. Vai trò này là trách nhiệm đối với tổng giám đốc tổ chức cho việc duy trì một đánh giá trước mắt về tiến trình của tổ chức một cách chín muồi và kế hoạch của nó

cho tiến trình cải tiến trong tương lai. SEPA phải trợ giúp sự khởi đầu và đánh giá định kì các tiến trình dự án. Chất xúc tác thu hút và phổ biến phần mềm thực hành tốt nhất có thể được hoàn thiện chỉ khi SEPA hiểu được cả cải tiến mong muốn và hoàn cảnh dự án. SEPA đóng một vai trò cần thiết trong bất kỳ tổ chức nào. Nó mang trách nhiệm và trách nhiệm giải trình đối với sự xác định tiến trình và sự bảo dưỡng của nó (sự sửa chữa, cải tiến, sự bổ xung công nghệ). SEPA có thể là một các nhân đơn lẻ, người tổng giám đốc, hay thậm chí là một nhóm các đại diện. SEPA phải thực sự là một chuyên gia, đủ khả năng và mạnh mẽ, không phải là một vị trí nhân viên được trả lại sự yếu kém bởi bộ máy quan liêu không hiệu quả.

Tổ chức kiểm tra dự án (PRA)

Chuyên gia xem xét lại dự án (PRA- Project Review Authority) là cá nhân đơn lẻ chịu trách nhiệm đối với việc bảo đảm rằng một dự án phần mềm dịch với toàn bộ tổ chức và các điều khoản, các thực hành, và các tiêu chuẩn phần mềm đơn vị thương mại. Một người quản lý dự án phần mềm có trách nhiệm đối với việc tập hợp các yêu cầu của một hợp đồng hay một vài tiêu chuẩn dịch dự án khác, và cũng là trách nhiệm đối với PRA. PRA xem xét lại cả hiệu năng của dự án tới các giao ước bằng hợp đồng và các điều khoản giao ước cách thức tổ chức của dự án. Khách hàng điều hành các yêu cầu hợp đồng, các mốc hợp đồng, các chuyển giao hợp đồng, hàng tháng quản lý xem xét, tiến độ, chất lượng, chi phí, lịch trình, và rủi ro. PRA xem xét các điều ràng buộc khách hàng tốt như sự dính chặt với các điều khoản cách thức tổ chức, các chuyển giao cách thức tổ chức, hiệu năng tài chính, và những rủi ro và những sự hoàn thiện khác.

Chuyên gia môi trường kỹ thuật phần mềm (SEEA)

Chuyên gia môi trường kỹ thuật phần mềm (SEEA- Software Engineering Environment Authority) đối với việc tự động hoá đối với tiến trình của tổ chức, việc bảo dưỡng các môi trường tiêu chuẩn của tổ chức, việc đào tạo các dự án để sử dụng môi trường, và bảo dưỡng các tài sản tổ chức có thể tái sử dụng rộng rãi. SEEA đóng vai trò cần thiết để đạt được một kết quả có ý nghĩa trong đầu tư đối với một tiến trình chung. Các công cụ, các kỹ thuật, và việc đào tạo có thể được truyền lại một cách có hiệu quả qua nhiều dự án nếu chỉ một vài người trong tổ chức (SEEA) có trách nhiệm đối với việc hỗ trợ và việc điều hành một môi trường tiêu chuẩn. Trong nhiều trường hợp, môi trường có thể được tăng lên, được tuỳ biến, hoặc được sửa chữa, nhưng sự tồn tại của 80% giải pháp ngầm định đối với mỗi dự án là tiêu chuẩn để đạt được sự thể chế hoá tiến trình của dự án và một ROI (kết quả đầu tư) tốt trong sự vốn công cụ đầu tư.

Cơ sở hạ tầng

Cơ sở hạ tầng của một tổ chức cung cấp hỗ trợ về nguồn nhân lực, nghiên cứu và phát triển dự án một cách độc lập và vốn tài sản kỹ thuật phần mềm. Cơ sở hạ tầng cho bất kỳ một công việc kinh doanh trong lĩnh vực phần mềm nhất định nào có thể bao gồm từ những thứ nhỏ nhặt đến những bộ máy hành chính được thiết lập cực kỳ vững chắc. Những bộ phận cấu thành cơ bản của cơ sở hạ tầng cho công tác tổ chức như sau:

- Tổ chức dự án: hệ thống tính thời gian; các hợp đồng, định giá, các điều khoản và điều kiện, sự gia nhập vào hệ thống thông tin doanh nghiệp
- Các trung tâm kỹ năng kỹ thuật: chứa và bảo quản các công cụ thông thường, hỗ trợ khi có đề nghị, nghiên cứu và phát triển độc lập
- Phát triển nghiệp vụ: tiến hành đào tạo thêm trong nội bộ, tuyển chọn nhân sự, lưu trữ cơ sở dữ liệu kỹ năng nhân sự, thư viện văn học, tài sản, xuất bản sách kỹ thuật

Một trung tâm dịch vụ có tổ chức sẽ thúc đẩy một môi trường chuẩn mực được tài trợ từ lĩnh vực kinh doanh đó và được duy trì như một tài sản vốn cho các dự án trong khuôn khổ tổ chức. SEEA là một nhóm cộng tác với SEPA. SEPA chịu trách nhiệm xác định và cải tiến quá trình, còn SEEA có trách nhiệm tự động hoá quá trình

Một điều rất quan trọng là các nhà tổ chức quản lý phải xem xét các môI trường phát triển phần mềm hoàn toàn giống như các môi trường phát triển phần cứng, chính xác là như trang thiết bị vốn. Có sự trở ngại trong phương pháp tiếp cận này trong hầu hết các tổ chức non nớt hoặc có quy mô nhỏ mà ở đó bao gồm tạo và phát triển một quy trình cụ thể và đó là chi phí dự án trực tiếp. Đối với hầu hết các tổ chức phần mềm vững chắc, quá trình này, trang bị công cụ là các tài sản tổ chức, hoàn toàn giống như trong các hệ thống điều lệ kỹ thuật khác. Theo đúng nghĩa, chúng được cấp vốn từ các nguồn vốn. Các mô hình tài chính có thể gồm cả sự tập trung vào các chi phí chung, chi phí hành chính hoặc các chi phí thực hiện dự án. Trong ngành công nghiệp phần mềm hiện nay, được đặc trưng bởi các hệ thống kế toán chuyên sâu, trang bị công cụ cho các dự án được cấp vốn và các phương pháp đăng ký sử dụng phần mềm tương đối ít tổ chức chuyển sang đầu tư vốn cho môi trường phần mềm của họ. Những tổ chức đó sẽ trở thành các nhà phát triển phần mềm với quy mô lớn, vững vàng, những nhà phát triển mà đã đạt được sự xác định quy trình ổn định và đã thiết lập được quan hệ bạn hàng lâu dài với các nhà cung cấp thiết bị phần mềm.

11.2.Tổ chức dự án

Hình 11.2 chỉ ra một tổ chức dự án ngầm định và sơ đồ nhiệm vụ phân bậc dự án và các trách nhiệm. Cấu trúc này có thể phù hợp với quy mô và hoàn cảnh của mỗi tổ chức dự án cụ thể.

Những đặc trưng chính của một tổ chức cơ bản :


Quản lý phần mềm
Mẫu(artifact)
Tình trạng kinh doanh
Kế hoạch phát triển phần mềm
Đánh giá vị thế
Các hoạt động
Tác động của khách hàng, tác động của PRA
Lên kế hoạch quản lý quy trình
Quản lý rủi ro
Xác định quy trình phần mềm
Cải tiến quy trình
Kỹ thuật hệ thống
Mẫu (artifact)
Trình bày quan điểm
Lập các yêu cầu
Các hoạt động
Gợi các yêu cầu
Cụ thể hoá các yêu cầu
Sử dụng mô hình hoá
Quản trị (Administration)
Mẫu (artifact)
Cấu trúc phân nhỏ công việc
Các hoạt động
Dự đoán và báo cáo tài chính
Định rõ và quản lý WBS
Kiến trúc phần mềm
Mẫu (artifact)
Mô tả thiết kế
Đưa ra các chi tiết kỹ thuật
Bộ thiết kế
Các hoạt động
Trình bày kế hoạch
Phân tích, thiết kế
Tạo mẫu kiến trúc
Cung cấp tài liệu kiến trúc
Trình bày phối hợp
Phát triển phần mềm
Mẫu
Tập thiết kế
Tập cài đặt
Tập yêu cầu
Tập triển khai
Các hoạt động
Thiết kế thành phần
Cài đặt thành phần
Kiểm thử thành phần
Bảo trì thành phần
Đánh giá phần mềm
Mẫu
Tập triển khai
Cơ sở dữ liệu SCO
Người dùng làm thủ công
Đưa ra sự miêu tả
Môi trường
Các tài liệu triển khai
Các hoạt động
Đưa ra sự đánh giá
Kiểm thử nhánh/kịch bản sử dụng
Kiểm thử kịch bản phát triển
Quản lý thay đổi
Chuyển tới người dùng
Quản lý hệ thống
Cấu hình môi trường
Bảo trì môi trường
Sản xuất các công cụ

Hình 11-2. *Tổ chức và trách nhiệm một dự án cơ bản*

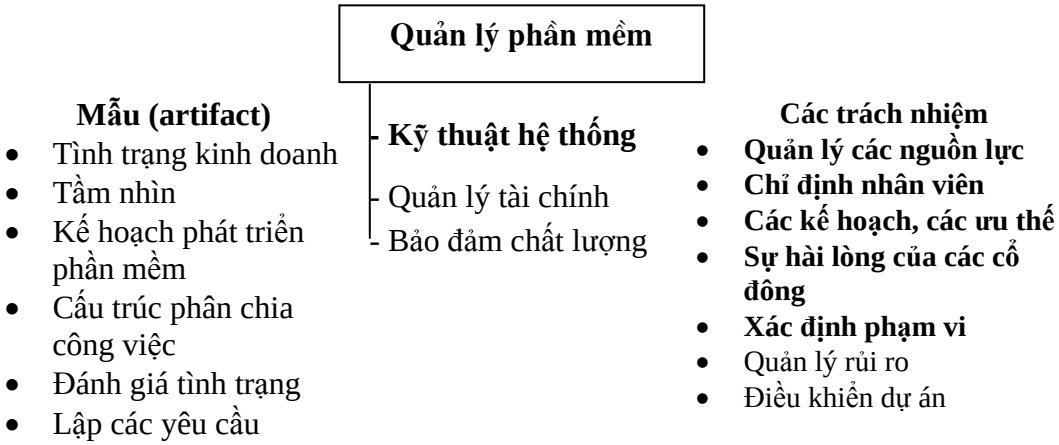
- Đội quản lý dự án là một thành viên tích cực, chịu trách nhiệm cả sản xuất và quản lý. Quản lý dự án không phải là một môn thể thao hấp dẫn cao độ.
- Đội kiến trúc chịu trách nhiệm về các mẫu thực và phối hợp các bộ phận, không chỉ đối với các chức năng của nhân viên.
- Đội phát triển sẽ thực hiện kiến tạo và duy trì các bộ phận. Đội đánh giá tách biệt với đội phát triển. Cấu trúc này thúc đẩy một triển vọng chất lượng độc lập và hướng một đội vào các hoạt động kiểm tra, đánh giá sản xuất và hài hoà với sự phát triển hiện thời.
- Chất lượng là công việc của mọi người , nó được phối hợp với toàn bộ mọi hoạt động và các khâu kiểm tra. Mỗi đội chịu trách nhiệm về một triển vọng chất lượng khác nhau

Đội quản lý phần mềm

Hầu hết các dự án đều được quản lý chặt chẽ. Các chương trình, chi phí , tính hoạt động và sự chờ đợi chất lượng liên quan chặt chẽ lẫn nhau và đòi hỏi có sự thoả thuận liên tục giữa nhiều cổ đông, những người có mục đích khác nhau. Đội quản lý phần mềm có trách nhiệm đem lại các điều kiện thắng lợi đến cho tất cả các cổ đông. ở điểm này, giám đốc dự án phần mềm phải lo lắng hàng ngày tới *sự cân bằng*. Hình 11.3 chỉ ra trạng thái các hoạt động của đội quản lý phần mềm qua vòng đời dự án.

Đội quản lý phần mềm có trách nhiệm lên kế hoạch phấn đấu và trách nhiệm đem lại một kế hoạch và làm cho kế hoạch đó thích nghi với những thay đổi trong sự hiểu biết về các yêu cầu và thiết kế. Theo hướng kết thúc này, đội quản lý giữ quyền sở hữu quản lý nguồn nhân lực và phạm vi dự án và thiết lập các vị trí ưu tiên điều hành trong suốt vòng đời dự án. ở mức độ lý thuyết, những hoạt động đó tương đương với việc quản lý tất cả các mong đợi của cổ đông trong suốt vòng đời dự án.

Đội quản lý phần mềm sở hữu toàn bộ các khía cạnh chất lượng. Đặc biệt nó có trách nhiệm vươn tới và duy trì sự cân bằng giữa các khía cạnh đó để giải pháp tổng thể thích hợp với toàn bộ cổ đông và là tốt nhất cho số đông nhất có thể trong số họ.



Trọng tâm vòng đời

Khởi đầu	Chi tiết	Xây dựng	Chuyển tiếp
Kế hoạch chi tiết từng giai đoạn Công thức đội **Khởi đầu hợp đồng** Chi phí thiết kế	Kế hoạch xây dựng từng giai đoạn Tuyển mộ đầy đủ nhân viên Cách giải quyết rủi ro Tiêu chuẩn chấp nhận sản phẩm Chi phí xây dựng	Kế hoạch chuyển đổi giai đoạn Tối ưu kế hoạch xây dựng **Quản lý rủi ro**	Lợi ích của khách hàng **Kết thúc hợp đồng** **Hỗ trợ bán hàng** Kế hoạch cho giai đoạn sau

Hình 11-3. *Các hoạt động của đội kiến trúc phần mềm.*

Đội thiết kế phần mềm

Đội thiết kế phần mềm chịu trách nhiệm đối với thiết kế, trách nhiệm này gồm kỹ thuật rất cần thiết để định rõ hoá đơn hoàn hảo về các nguyên liệu cho phần mềm và các kỹ thuật cần thiết để tạo nên sự cân đối quan trọng để tất cả các bộ phận truyền thống được trình bày chi tiết trên quy mô rộng lớn, mà chi phí xây dựng/lắp ráp có thể dự báo được là rất cao. Hình 11.4 chỉ ra trọng điểm hoạt động của đội kiến trúc phần mềm trong suốt vòng đời dự án.

Đối với bất kỳ dự án nào, kỹ năng kiến trúc phần mềm là rất quan trọng. Nó tạo ra một cơ cấu tổ chức để làm cho việc liên lạc giữa các đội thuận tiện hơn và để đạt được chất lượng mang tính hệ thống rộng khắp và phục vụ cho việc thực thi các ứng dụng. Với một đội kiến trúc tốt, một đội phát triển trung bình cũng có thể thành công. Nếu đội kiến trúc yếu kém thì một đội phát triển giàu kinh nghiệm gồm các lập trình viên xuất xắc cũng sẽ có thể không thành công.

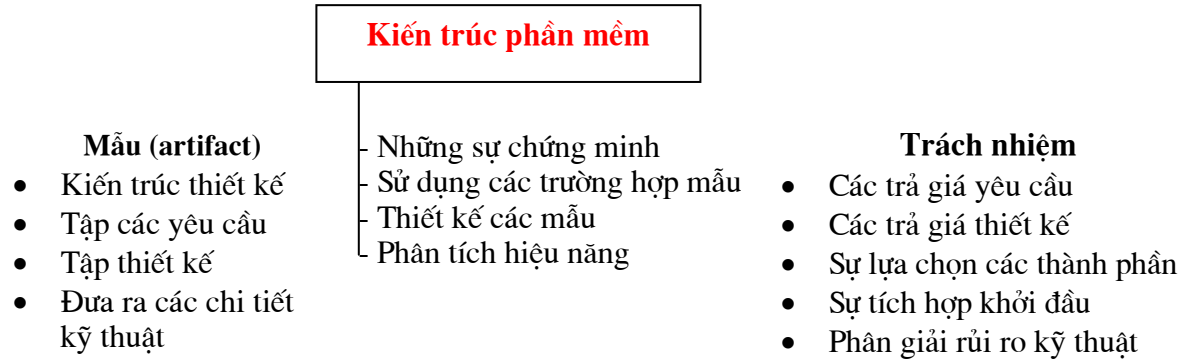
ở hầu hết các dự án, giai đoạn khởi đầu và các giai đoạn phát sinh sẽ bị chi phối bởi hai đội riêng biệt: đội quản lý phần mềm và đội kiến trúc phần mềm (sự khác biệt có thể rất mờ nhạt, nó phụ thuộc vào quy mô). Đội phát triển phần mềm và đội đánh giá phần mềm sẽ có xu hướng làm nhiệm vụ hỗ trợ trong khi chuẩn bị cho giai đoạn sản xuất đầy đủ. Vào thời gian bắt đầu việc xây dựng, kiến trúc chuyển sang chế độ duy trì và cần được hỗ trợ ở mức tối thiểu để đảm bảo rằng có sự kế thừa liên tục các kỹ thuật.

Để thành công, đội thiết kế phải có chuyên môn khá rộng, bao gồm:

- Có kinh nghiệm (trong lĩnh vực này) để có cách nhìn chấp nhận được về mặt thiết kế (những thành phần thiết kế quan trọng trong mẫu thiết kế) và cách nhìn về thực tế sử dụng (các thành phần thiết kế của một mô hình sử dụng thực tế).

- Có kinh nghiệm về công nghệ phần mềm để có cách nhìn thoả đáng về các quy trình (sự đồng thời, kiểm soát chuỗi các liên hệ giữa thiết kế, các bộ phận và các mô hình triển khai), cái nhìn bộ phận (cấu trúc thực hiện), cái nhìn triển khai (cấu trúc triển khai).

Đội kiến trúc chịu trách nhiệm về chất lượng của hệ thống, bao gồm các thuộc tính: có chất lượng, sự hoàn thành, khả năng duy trì. Những thuộc tính này trải rộng trên nhiều thành phần và cho biết các thành phần đó phối hợp với nhau như thế nào để để đưa ra một giải pháp hiệu quả. Theo đó, đội kiến trúc quyết định sẽ giải quyết hầu hết các vấn đề về thiết kế bộ phận như thế nào.



Trọng tâm vòng đời

Sự khởi đầu	**Chi tiết**	**Xây dựng**	**Quá độ**
Bản mẫu kiến trúc **Trả giá tạo/ mua** Xác định kế hoạch chính Xác định tiêu chuẩn đánh giá thiết kế	Kiến trúc đường cơ sở **Thể hiện kế hoạch** **Chính** Đường cơ sở trả giá tạo/ mua	Bảo dưỡng kiến trúc Giải quyết vấn đề nhiều thành phần Điều chỉnh hiệu năng Cải tiến chất lượng	Bảo dưỡng kiến trúc Giải quyết vấn đề nhiều thành phần Điều chỉnh hiệu năng Cải tiến chất lượng

Hình 11-4 . *Các hoạt động của đội kiến trúc phần mềm*

Đội phát triển phần mềm

Hình 11-5 chỉ ra trọng tâm các hoạt động của đội phát triển phần mềm trong suốt vòng đời dự án.

Phát triển phần mềm

Các đội thành phần

Mẫu (artifact)

- Tập thiết kế
- Tập cài đặt
- Tập triển khai

Các trách nhiệm

- Thành phầnn thiết kế
- Thành phần cài đặt
- Kiểm tra từng thành phần riêng rẽ
- Thành phần bảo dưỡng
- Thành phần cung cấp tài liệu

Trọng tâm vòng đời

Khởi đầu	Chi tiết	Xây dựng	Quá độ
Hỗ trợ việc tạo mẫu Trả giá tạo /mua	Thiết kế các bộ phận thiết yếu Cài đặt và kiểm tra các bộ phận chủ yếu	Thiết kế bộ phận Thực thi bộ phận Kiểm tra từng bộ phận riêng rẽ Thành phần bảo dưỡng	Bộ phận duy trì Bộ phận cung cấp tài liệu

Hình 11-5. *Các hoạt động của đội phát triển phần mềm*

Đội phát triển phần mềm là một nhóm có đặc thù ứng dụng nhất. Nhìn chung, đội này gồm một số các đội phụ để dành cho nhóm các thành phần đòi hỏi một kỹ năng thông thường. Việc xếp đặt các kỹ năng cơ bản gồm:

- Thành phần thương mại: Các chuyên gia có kiến thức sâu về thành phần thương mại tập trung vào việc thiết kế hệ thống
- Cơ sở dữ liệu: Các nhà chuyên môn có kinh nghiệm trong lĩnh vực tổ chức, lưu trữ và điều chỉnh dữ liệu
- Giao diện đồ họa người sử dụng: Các nhà chuyên môn có kinh nghiệm trong việc trình bày cách tổ chức và trình bày dữ liệu và tác động tất yếu của người sử dụng đến việc hỗ trợ đầu vào, đầu ra con người và nhu cầu điều khiển
- Hệ điều hành và mạng: Các chuyên gia giàu kinh nghiệm trong thực hiện nhiều phần mềm có mục tiêu hướng vào mạng các tài nguyên phần cứng, gồm tất cả các vấn đề điều hành cơ bản cùng với sự khởi động đồng bộ hoá, chia sẻ tài nguyên, quản lý tên, cấu hình lại, kết thúc và liên lạc giữa các đối tượng
- Lĩnh vực ứng dụng: Các chuyên gia giàu kinh nghiệm trong các thuật toán, các quy trình ứng dụng hay các quy tắc làm việc cụ thể cho hệ thống

Đội phát triển phần mềm chịu trách nhiệm về chất lượng của các thành phần cá biệt gồm tất cả các công việc phát triển, kiểm tra, duy trì các thành phần. Việc kiểm tra các thành phần có thể được xây dựng giống như phần mềm có khả năng lặp lại và tự báo cáo bằng tài liệu

mà được xem xét như các thành phần mã nguồn hoạt động khác để nó có thể được duy trì một cách tự nhiên và sẵn sàng cho cuộc kiểm tra trở lại một cách tự động. Đội phát triển sẽ quyết định bất kỳ vấn đề thiết kế hay thực hiện nào ảnh hưởng tới một thành phần đơn lẻ sẽ được giải quyết như thế nào.

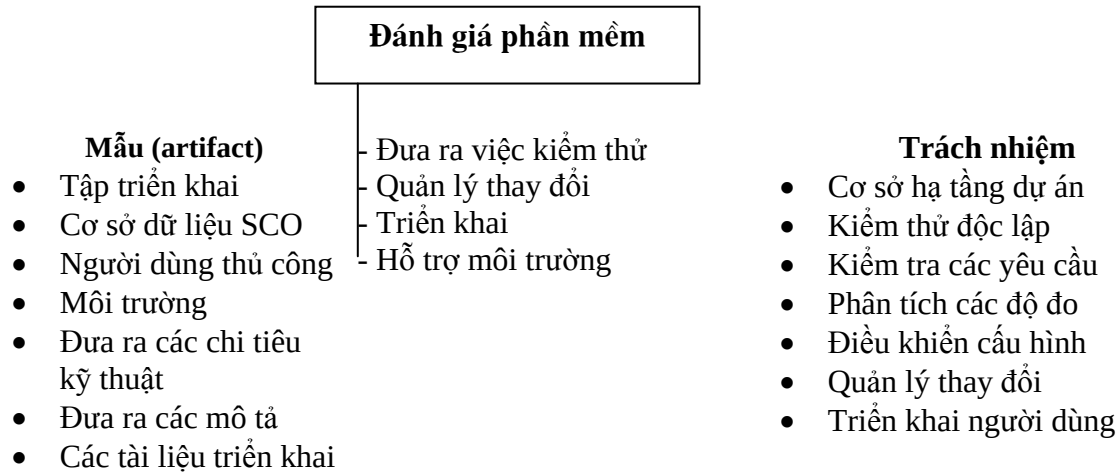
Đội đánh giá phần mềm

Hình 11-6 chỉ ra trọng tâm hoạt động của đội đánh giá phần mềm trong suốt vòng đời dự án.

Có hai lý do để lập một đội đánh giá phần mềm độc lập. Thứ nhất là phải đảm bảo một triển vọng chất lượng độc lập. Phương cách thường gây tranh cãi nay có những ưu điểm (như đảm bảo rằng những ảnh hưởng xấu đến quyền sở hữu của người phát triển không gây xấu tới việc đánh giá chất lượng) và khuyết điểm (ví dụ như trong chừng mực nào đó, trút bỏ dần hết quyền sở hữu của đội phát triển đối với chất lượng). Một lý do quan trọng hơn là sử dụng đội đánh giá phần mềm để khai thác đồng thời giữa các hoạt động. Các kế hoạch có thể đẩy nhanh bởi việc phát triển phần mềm và chuẩn bị cho việc kiểm tra song song với các hoạt động phát triển. Thay đổi sự quản lý, kế hoạch kiểm tra và kiểm tra kế hoạch phát triển có thể thực thi song song với việc thiết kế và phát triển.

> *Một quy trình hiện đại nên sử dụng việc thử nghiệm trên khả năng cơ bản hay định hướng cách sử dụng (nó có thể trải rộng trên nhiều thành phần) được tổ chức như một việc xây dựng liên tục và được cơ khí hoá thông qua hai mẫu (artifact).*

1. Đưa ra chi tiết kỹ thuật (kế hoạch và tiêu chuẩn cho mỗi lần đưa ra chi tiết kỹ thuật).
2. Đưa ra bản mô tả (những kết quả)



Trọng tâm vòng đời

Khởi đầu	Chi tiết	Xây dựng	Quá độ
Kế hoạch cơ sở hạ tầng Mẫu kịch bản chính	Vạch danh giới cơ sở hạ tầng Kiểm tra thiết kế đưa ra Quản lý thay đổi Người dùng thủ công ban đầu	Nâng cấp cơ sở hạ tầng Đưa ra việc kiểm tra Quản lý thay đổi Danh giới người sử dụng thủ công Kiểm tra các yêu cầu	Duy trì cơ sở hạ tầng Đưa ra vạch danh giới Quản lý thay đổi **Triển khai đến người sử dụng** **Kiểm tra các yêu cầu**

Hình 11-6. *Các hoạt động của đội đánh giá phần mềm*

Mỗi việc phát hành có thể gồm một số (có thể chưa đầy đủ) thành phần bởi vì sự phối hợp sẽ tiến hành liên tục. Các chỉ tiêu đánh giá sẽ cung cấp tài liệu, điều mà khách hàng có thể mong đợi ở một giai đoạn quan trọng, bản mô tả được đưa ra sẽ chứng minh kết quả thử nghiệm. Sự lặp lại cuối cùng này nhìn chung sẽ tương tự việc thử nghiệm được chấp nhận và gồm các mức độ chi tiết tương tự với các mức độ chi tiết trong kế hoạch thử nghiệm phần mềm, các thủ tục và các báo cáo. Những mẫu (artifact) đó phát triển từ những phiên bản khá ngắn gọn và tóm tắt ở sự lặp đi lặp lại ban đầu, thành các tài liệu chi tiết và tỉ mỉ hơn, có sự thảo luận tính chất bổ xung và các chi tiết hoàn thành trong những phiên bản sau. Ngay cả với các thử nghiệm cách sử dụng, các thành phần kiểm tra sẽ được phát triển ở khuôn mẫu giống như sự phát triển của các trường hợp thử nghiệm thành phần. Ví dụ, một dự án thì nên tạo ra những dự đoán biến cố thay đổi trong thực nghiệm, cái mà là các chương trình phần mềm trong quyền sở hữu của chúng hơn là việc phát triển các tài liệu thủ tục thử nghiệm. Những kịch bản khó tránh khỏi phải thay đổi cách quản lý giống như phần mềm khác và luôn đảm bảo cập nhật với việc kiểm tra tự động hoá. Một số việc kiểm tra bộ phận có thể nâng cao chỉ tiêu đánh giá mà kết quả được ghi trong bản mô tả chi tiết. Nhiều bộ phận có thể chỉ được kiểm tra qua loa bởi đội phát triển mà kết quả chỉ được đưa ra trong phạm vi phần mềm kiểm tra được xây dựng bởi nhà phát triển. Việc kiểm tra chính thức đối với nhiều bộ phận sẽ được xếp vào các tiêu chuẩn đánh giá ở mức cao hơn (thường là các dự đoán theo hướng có khả năng) và các bản mô tả tương ứng. Tất cả các bộ phận không được tạo ra bình đẳng: Một số bộ phận phải được kiểm tra chính thức để thẩm tra lại các yêu cầu trong khi một số bộ phận khác tốt nhất là được kiểm tra trong khuôn khổ kiểm tra có khả năng. Việc nhận xét nay phải được chuyển sang cho đội đánh giá.

Đội đánh giá chịu trách nhiệm đối với chất lượng ban đầu liên quan đến các yêu cầu và mong đợi của khách hàng. Bởi vậy, đội đánh giá có trách nhiệm giải quyết bất kỳ vấn đề chất lượng nào ảnh hưởng đến kỳ vọng của khách hàng dù những kỳ vọng đó có được ghi trong các yêu cầu hay không.

11.3.Tiến triển của các tổ chức

Tổ chức dự án trình bày cơ cấu của đội và cần phải phát triển đúng với kế hoạch dự án đã đề ra trong cơ cấu làm việc. Hình 11-7 minh hoạ sự thay đổi trọng tâm của đội trong vòng đời dự án với khoảng 50% số nhân viên làm một trong các hoạt động trong mỗi giai đoạn

Sự khác biệt của các hoạt động được nhấn mạnh ở mỗi giai đoạn như sau:

- Đội khởi động: một tổ chức được tập trung vào việc lên kế hoạch được sự giúp đỡ từ các đội khác để đảm bảo rằng các kế hoạch đó bộc lộ một sự thống nhất các triển vọng.
- Đội trình bày chi tiết: một tổ chức tập trung vào việc thiết kế ở đó hiệu lực của dự án không thay đổi đối với đội thiết kế phần mềm và được hỗ trợ bởi các đội phát triển phần mềm và đội đánh giá phần mềm vì nó cần thiết để đạt được một cơ cấu ổn định
- Đội xây dựng: một tổ chức khá cân bằng mà trong đó hầu hết các hoạt động tập trung vào đội phát triển phần mềm và đội đánh giá phần mềm
- Đội chuyển tiếp: một tổ chức tập trung vào các khách hàng ở đó việc phản ánh cách sử dụng dẫn đến các hoạt động phát triển

Trình bày chi tiết về các đội phụ, các trách nhiệm và công việc trọn gói là khá quan trọng, nhưng chỉ tới khi chi tiết việc lập kế hoạch trong WBS đã ổn định. Xác định mọi chi tiết về cấu trúc các đội ở mức thấp hơn có thể dẫn tới việc giảm hiệu quả nghiêm trọng

Hình 11-7. *Sự tiến triển của đội làm dự án phần mềm qua vòng đời*

Chương 12

Tự động hoá quá trình

(Process automation)

Phần lớn các tổ chức phát triển phần mềm luôn phải tập trung vào việc phát triển những quá trình hoàn chỉnh để tăng khả năng dự đoán của việc quản lý phần mềm và hiệu quả của dòng phần mềm sản phẩm doanh nghiệp của họ (bao gồm chất lượng sản phẩm, thời gian tung ra thị trường, thu hồi vốn đầu tư, và năng suất). Trong khi sự định nghĩa và mô tả bề ngoài của quá trình là cần thiết thì một mức chuẩn của tự động hoá quá trình cũng là một yêu cầu cho những dự án phát triển phần mềm hiện đại để đạt được lợi nhuận.

Chủ điểm

- *Môi trường phải là lớp tạo tác đầu tiên của quá trình*
- *Tự động hoá quá trình và quản lý sự thay đổi là tới hạn với một quá trình lặp lại. Nếu sự thay đổi là quá đắt thì tổ chức phát triển sẽ không chấp nhận*
- *Kỹ thuật khứ hồi và môi trường hoà nhập thúc đẩy tự do thay đổi và sự phát triển hiệu quả của các tạo tác kỹ thuật*
- *Tự động hoá độ đo có vai trò quan trọng để kiểm soát dự án một cách hiệu quả*
- *Những người liên quan cần truy nhập các tài nguyên môi trường để cải tiến sự lặp lại với đội ngũ phát triển và tăng thêm chất lượng cho quá trình*

Sự tự động hoá cần phát triển phụ thuộc vào mức độ của sự công việc. Giống như việc quy trình xây dựng thay đổi phụ thuộc vào việc bạn đang xây một ngôi nhà búp bê, một ngôi nhà cho gia đình hay một ngôi nhà cao tầng, một quá trình trình phần mềm(software process) thay đổi qua một dải từ thao tác bảng tính đơn người(single-person spreadsheet) cho đến những thao tác lớn, cấu tạo bội, những ứng dụng thất bại không có lợi nhuận. Các kỹ thuật, việc huấn luyện, phạm vi thời gian, tiêu chuẩn chấp nhận được và các mức độ khác nhau của sự tự động hoá có ý nghĩa khác biệt tại các điểm cuối đối lập của dải

Hầu hết các tổ chức phần mềm phải đối diện với nhiệm vụ hoà nhập môi trường của họ với cơ sở hạ tầng cho sự phát triển phần mềm. Quy trình này sẽ dẫn đến sự lựa chọn nhiều hay ít những công cụ không tương thích có những thông tin khác biệt, được cung cấp bởi những đại lý khác nhau, làm việc trên những nền khác nhau, sử dụng các biệt ngữ khác nhau và dựa trên những giả định tiến trình khác nhau. Việc hoà nhập một cơ sở hạ tầng như vậy đã chỉ ra rằng còn nhiều việc phải xem xét chứ không như chúng ta tưởng.

Tự động hóa quá trình phát triển và thiết lập một cơ sở hạ tầng(infrastructure) cung cấp cho các luồng công việc (workflow) dự án là những hoạt động quan trọng của giai đoạn kỹ

thuật của vòng đời(life cycle). Những hoạt động này bao gồm tuyển chọn công cụ, toolsmithing phong tục và tự động hoá quá trình cần thiết thực hiện ngược với kế hoạch phát triển với hiệu quả chấp nhận được. Phát triển một cách tự nhiên môi trường phát triển vào trong môi trường bảo trì cũng là thiết yếu với bất kỳ dự án phát triển phần mềm.

Để làm phức tạp các vấn đề hơn nữa, hiếm khi tìm một người giữ tiền đặt cọc(hay là người có liên quan – stakeholder) người mà đối xử môi trường như một mẫu lớp đầu tiên cần thiết để tiếp tục sự duy trì của sản phẩm. Môi trường cung cấp sự tự động hoá quá trình là một mẫu hữu hình mà nó là tới hạn của giá vòng đời của hệ thống đang được phát triển. Mức đỉnh WBS dã được giới thiệu ở chương 10 tổ chức môi trường như là luồng công việc lớp đầu tiên.

Phần 3.2 giới thiệu ba mức của quá trình. Mỗi một mức đòi hỏi một cấp độ cụ thể của tự động hoá quá trình cho quá trình tương ứng để có thể được thực hiện một cách hệu quả:

1. Siêu quá trình(metaprocess): Các chính sách của một tổ chức, các thủ tục, việc thực hành để quản lý một đường phần mềm xuyên suốt của doanh nghiệp. Sự tự động hoá cung cấp cho mức độ này được gọi là cơ sở hạ tầng. Một cơ sở hạ tầng là một bảng liệt kê các công cụ thích hợp, các mẫu tạo tác(artifact), đường lối chỉ đạo vi tiến trình(microprocess guideline), các đường lối chỉ đạo tiến trình vĩ mô, và thư viện về những ví dụ tiền lệ của các kế hoạch dự án trước đó cùng với kết quả tương ứng.
2. Quá trình vĩ mô(macroprocess): Các chính sách của một dự án, các thủ tục, và việc thực hành để sản xuất một sản phẩm phần mềm hoàn chỉnh cùng với giá cả, lịch trình và những ràng buộc chất lượng. Sự tự động hoá cung cấp cho quá trình của dự án được gọi là môi trường(environment). Một môi trường là một tập hợp đặc biệt các công cụ để sản xuất một tập đặc biệt của các tạo tác như được chỉ đạo bởi một kế hoạch dự án đặc biệt.
3. Quá trình vi mô(microprocess): Một số chính sách của đội dự án, các thủ tục, và việc thực hành để đạt được một tạo tác của quá trình phần mềm. Sự tự động hoá cung cấp để phát sinh một tạo tác được gọi chung là công cụ. Một công cụ điển hình bao gồm quản lý các yêu cầu, mẫu trực quan, các trình dịch, các trình soạn thảo, các trình bắt lỗi, quản lý sự thay đổi, tự động hoá các độ đo, tự động hoá tài liệu, tự động hoá thử nghiệm, định giá cả, và tự động hoá luồng công việc.

Trong khi sự tập trung chính của tự động hoá quá trình là luồng công việc của môi trường mức độ dự án, hoàn cảnh cơ sở hạ tầng của tổ chức cấp trên của dự án và các khối xây dựng công cụ là điều kiện tiên quyết quan trọng.

12.1.Các công cụ

Những khối xây dựng sự tự động hoá

Có rất nhiều công cụ sẵn sàng cho tự động hoá quá trình phát triển phần mềm. Phần này sẽ đưa ra một cái nhình tổng quan của môi trường lõi cần thiết hỗ trợ cho khung làm

việc(framework). Nó giới thiệu một số công cụ quan trọng mà có xu hướng sử dụng rộng rãi qua các kế hoạch phần mềm và có sự tương quan với cơ cấu của tiến trình.(Nhiều công cụ khác và sự giúp đỡ tự động cho tiến trình không được tính đến). Hầu như những công cụ phát triển nhân phần mềm đều tương ứng với một luồng công việc của quy trình, điều này được minh hoạ trong hình 12- 1. Mỗi một luồng công việc của quy trình cần có một sự hỗ trợ tự động hoá riêng bit. Trong một số trường hợp, cần thiết phải phát sinh một hiện tượng giả; nó cần thiết cho kế toán đơn giản. Một vài chỉ trích liên quan tới mỗi luồng công việc sẽ được thảo luận sau đây.

Dòng công việc	Các công cụ môi trường và tự động hoá quy trình
Quản lý	Tự động hoá luồng công việc, tự động hoá độ đo
Môi trường	Quản lý sự thay đổi, tự động hoá tài liệu
Các yêu cầu	Quản lý các yêu cầu
Thiết kế	Kiểu trực quan
Thực hiện	Soạn thảo - Biên dịch – Bắt lỗi
Định giá	Tự động kiểm tra, tự hiệu chỉnh thiếu sót
Triển khai	Tự hiệu chỉnh thiếu sót
Tiến trình	Chính sách tổ chức

Vòng đời	Khởi đầu	Chi tiết	Xây dựng	Triển khai

Hình 12-1: Sự tự động hoá điển hình và các thành phần công cụ cung cấp cho các luồng công việc của quá trình

Quản lý(management)

Có rất nhiều cơ hội cho sự tự động hoá việc lập kế hoạch dự án và kiểm soát các hoạt động của sự quản lý luồng công việc. Công cụ ước lượng giá cả phần mềm và công cụ WBS rất hữu ích để phát sinh việc lập kế hoạch giả tạo tác. Tương phản với sự quản lý một dự án, công cụ quản lý luồng công việc và một bảng kiểm soát dự án phần mềm có thể duy trì theo kiểu trực tuyến của việc định giá tình trạng là thuận tiện. Sự hỗ trợ tự động hoá này có thể hoàn thiện sự hiểu biết sâu sắc của việc tập hợp độ đo(metric) và khái niệm thuật lại được thảo luận trong chương 13. Toàn bộ khái niệm của bảng kiểm soát dự án phần mềm được thảo luận trong phần 13.6

Môi trường(environment)

Quản lý cấu hình và kiểm soát kiểu là cần thiết trong một quá trình phát triển lặp lại hiện đại. Rất nhiều cách nhìn nhận độ đo được giới thiệu trong chương 13 là phụ thuộc vào việc đo sự thay đổi trong ranh giới tạo tác phần mềm. Phần 12.2 thảo luận một số sự thay đổi trong tự động hoá quản lý phải được cung cấp bởi môi trường.

Các yêu cầu(requirement)

Cách tiếp cận thông thường là phân tích các yêu cầu hệ thống thành những yêu cầu hệ thống con, những yêu cầu hệ thống con thành những yêu cầu thành phần và những yêu cầu của thành phần thành những yêu cầu đơn vị. Sự giải quyết bình đẳng của tất cả các yêu cầu đã mất dần những giờ kỹ thuật từ việc điều khiển các yêu cầu, sau đó lãng phí thời gian trên công việc giấy tờ kết hợp với khả năng vạch ra chi tiết sau này sẽ chắc chắn được loại bỏ giống như việc điều khiển các yêu cầu và sự hiểu biết thiết kế kế tiếp đã tiến triển.

Trong một quy trình hiện đại, những yêu cầu hệ thống giành được trong sự trình bày hình ảnh(vision statement). Những mức thấp của sự yêu cầu được điều khiển bởi quá trình - tổ chức bởi sự lặp lại nhiều hơn là bởi thành phần mức thấp - trong mẫu của tiêu chuẩn định giá. Những tiêu chuẩn này được thu hút bởi một tập hợp cách sử dụng và mục đích biểu diễn thực chất khác. Sự trình bày hình ảnh này giành được hợp đồng giữa nhóm thiết kế và người mua. Những thông tin này có thể được phát triển nhưng thay đổi chậm, trong suốt vòng đời, và có thể được biểu diễn dưới hình thức mà người sử dụng có thể hiểu được. Tiêu chuẩn đánh giá có thể được lấy trong tạo tác đặc tả, cái mà được chuyển tiếp nhanh chóng của những đối tượng với một sự lặp lại cho trước. Tiêu chuẩn đánh giá này có thể lấy được từ sự trình bày hình ảnh cũng như từ những nguồn khác, thí dụ phân tích sản xuất/bán, những rủi ro quản lý gặp phải, nghiên cứu kiến trúc, làm quá trình thực hiện chậm lại, giới hạn chất lượng và thậm chí là cả việc may rủi.

Mô hình lặp đi lặp lại cho phép khách hàng và chuyên viên thiết kế làm việc một cách xác thực, phát triển những phiên bản của hệ thống. Trên thực tế, những yêu cầu có thể và phải phát triển cùng với một cấu trúc và sự phát triển một lớp các ứng dụng.

Bằng cách này, khách hàng và chuyên viên thiết kế có một điểm chung, sự hiểu biết các thuộc tính của đối tượng và sự cân bằng giá/lịch trình/thi công kết hợp với những yêu cầu đó. Hơn nữa là tập trung vào sự nhất quán, sự hoàn chỉnh và khả năng theo dõi những chỉ tiêu kỹ thuật đòi hỏi chưa tốt, dự án cần tập trung vào việc hoàn thành những chỉ tiêu kỹ thuật đúng đắn của hình ảnh quá trình và để phát triển những chỉ tiêu kỹ thuật mức thấp cho đến những bộ tiêu chuẩn đánh kế tiếp ngược lại sự lặp lại thiết kế thu tạo ra.

Sự phân nhánh của cách tiếp cận này trên sự hỗ trợ của môi trường cho việc quản lý cách yêu cầu bao gồm hai phần:

1. *Cách tiếp cận những yêu cầu được giới thiệu phụ thuộc vào cả sự biểu diễn nguyên mẫu và biểu diễn mẫu cơ sở. Kết quả là, môi trường nên cung cấp kỹ thuật tự động hoá tài liệu tổng hợp và mẫu trực quan để đạt được chỉ tiêu kỹ thuật nguyên mẫu và sử dụng mẫu trạng thái. Điều này là cần thiết để quản lý và theo dõi sự thay đổi cho mỗi dạng mẫu và biểu diễn chúng trong khuôn mẫu con người có thể đọc được, dù ở dạng điện tử hay là trên giấy.*
2. Khả năng sao lại giữa những yêu cầu và những tạo tác khác cần được tự động hoá. Tuy nhiên sự mở rộng của khả năng sao lại giữa các bộ là một chủ đề của một cuộc tranh cãi lâu dài. ý kiến của tôi là những tập các hiện tượng giả của những yêu cầu cần một khả năng sao chép xác định cho những hiện tượng giả thử nghiệm, bởi vì nhóm ước định phải có trách nhiệm chỉ ra rằng mức độ phù hợp với những yêu cầu của sản phẩm. Tuy nhiên, tôi không thấy một lý do thuyết phục nào để theo đuổi mối quan hệ khả năng sao chép mạnh giữa tập tạo tác của những yêu cầu và những tạo tác kỹ thuật khác. Vấn đề này mở ra sự mô tả, như biểu diễn trong tập của những yêu cầu, và lời giải mở ra một sự mô tả, như biểu diễn trong tập tạo tác kỹ thuật khác, thường xuyên có khả năng sao chép là rất khó khăn để biểu diễn. Điều này đặc biệt đúng trong những kiến trúc của thành phần cơ bản mà trong đó có phần lớn là các thành phần thương mại. Nếu quá trình đòi hỏi khả năng sao chép khắt khe giữa những yêu cầu và bản thiết kế, kiến trúc có thể được phát triển bằng cách tối ưu khả năng sao chép yêu cầu hơn là toàn bộ bản thiết kế. Hiệu quả của việc làm này thậm chí còn hơn cả dự định nếu những công cụ đã từng được sử dụng để tự động hoá tiến trình này.

Thiết kế(design)

Những công cụ để hỗ trợ cho những yêu cầu, thiết kế, thực hiện, và luồng công việc của sự ước định(assessment) thường được sử dụng cùng nhau. Trên thực tế, càng ít chia nhỏ chúng càng tốt. Sự hỗ trợ chính đòi hỏi cho luồng công việc thiết kế là mẫu trực quan, cái được sử dụng để đạt được mẫu thiết kế, biểu diễn chúng dưới khuôn mẫu con người có thể đọc được, và chuyển chúng thành mã nguồn. Một tiến trình kiến trúc đầu tiên và minh hoạ cơ sở có thể thực hiện bởi sự tồn tại những thành phần kiến trúc và sản phẩm trung gian.

***Thực hiện*(Implementation)**

Luồng công việc thực hiện phần lớn dựa trên môi trường lập trình(soạn thảo, biên dịch, bắt lỗi, liên kết, thời gian chạy) nhưng cũng phải bao gồm sự kết hợp thực sự với những công cụ quản lý thay đổi, những công cụ mẫu trực quan, và những công cụ tự động kiểm tra để hỗ trợ quy trình sản xuất.

Đánh giá và triển khai(assessment and deployment)

Luồng công việc đánh giá đòi hỏi tất cả các công cụ đã được thảo luận cũng như những khả năng thêm vào để hỗ trợ tự động hóa kiểm tra và quản lý thử nghiệm. Để tăng sự tự do thay đổi, sự nghiên cứu và tài liệu sản xuất phải được tự động hoàn toàn. Việc kiểm tra sai sót là một công cụ quan trọng khác hỗ trợ cho sự đánh giá: Nó cung cấp công cụ quản lý thay đổi cần thiết tự động hoá độ đo và kiểm soát giới hạn phiên bản. Nó cũng cần thiết để hỗ trợ luồng công việc triển khai trong suốt vòng đời.

12.2.Môi trường dự án

Những tạo tác môi trường dự án phát triển qua ba trạng thái tách biệt: môi trường mẫu thử, môi trường sự phát triển và môi trường duy trì.

1. *Môi trường mẫu thử bao(prototyping environment) gồm một thí điểm kiến trúc cho các kiến trúc dự án mẫu thử để đánh giá cân bằng trong suốt thời gian bắt đầu và những pha phát sinh của vòng đời. Cấu hình không chính thức này của những công cụ có khả năng hỗ trợ cho những hoạt động sau đây:*

- Những thoả hiệp thực hiện và những phân tích rủi ro kỹ thuật
- Những thoả hiệp sản xuất/bán và nghiên cứu tính khả thi cho những sản phẩm thương mại
- Những thoả hiệp định lại cấu hình chấp nhận sai sót/động lực
- Sự phân tích những rủi ro kết hợp với sự chuyển tiếp tới việc thực hiện đúng kích thước
- Sự phát triển của những đợt kiểm tra, và những công cụ phù hợp với việc phân tích những yêu cầu.

2. *Môi trường phát triển(development environment) nên gồm một bộ của những công cụ phát triển cần thiết để hỗ trợ cho những luồng công việc khác nhau và hỗ trợ cho việc tiến tới sự mở rộng tối đa có thể.*
3. *Môi trườn duy trì(maintennance development) nên xảy ra đồng thời với một phiên bản hoàn chỉnh của môi trường sự phát triển. Trong một số trường hợp, môi trường duy trì có thể là một tập con của mội trường phát triển sinh ra như là một trong những sản phẩm cuối cùng của dự án.*

Việc chuyển tiếp tới một quá trình phần mềm hoàn chỉnh đưa ra những thách thức và cơ hội cho việc kiểm tra quản lý của những hoạt động hiện tại và cho sự đánh giá của sự tiến bộ

xác thực và chất lượng. Sự trải qua dự án của thế giới thực đã chỉ ra rằng một môi trường hoà hợp cao là cần thiết để làm dễ dàng và phát huy hiệu lực kiểm soát quản lý của tiến trình. Để đạt được mục đích cuối cùng, có bốn môi trường quan trọng như là tiêu chuẩn cho phạm vi quản lý và sự thành công của một tiến trình phát triển lặp lại:

1. *Những công cụ phải được hoà hợp để duy trì sự chắc chắn và khả năng sao chép(traceability).Kỹ thuật trọn vòng là khái niệm sử dụng để mô tả c yêu cầu cốt yếu này cho môi trường để hỗ trợ sự phát triển lặp lại*
2. *Quản lý sự thay đổi phải được tự động hoá và có hiệu lực cho những sự lặp lại đa quản lý và có thể tự do thay đổi. Sự thay đổi là rất cơ bản của phát triển lặp lại.*
3. *Cơ sở hạ tầng của tổ chức được môi trường dự án nhận từ những cơ sở thông thường của những quá trình và những công cụ. Một cơ sở hạ tầng thông thường sẽ khuyến khích sự chắc chắn trong dự án, sử dụng lại quá trình huấn luyện, sử dụng lại những bài đã học, và những sự cải tiến mang tính chiến lược với siêu tiến trình của tổ chức.*
4. *Mở rộng tự động hoá hỗ trợ cho các môi trường của những người liên quan có thể hỗ trợ nhiều hơn để giảm bớt giấy tờ trao đổi thông tin và hiệu xem xét quả hơn các tạo tác kỹ thuật*

12.2.1 Kỹ thuật trọn vòng(round-trip engineering)

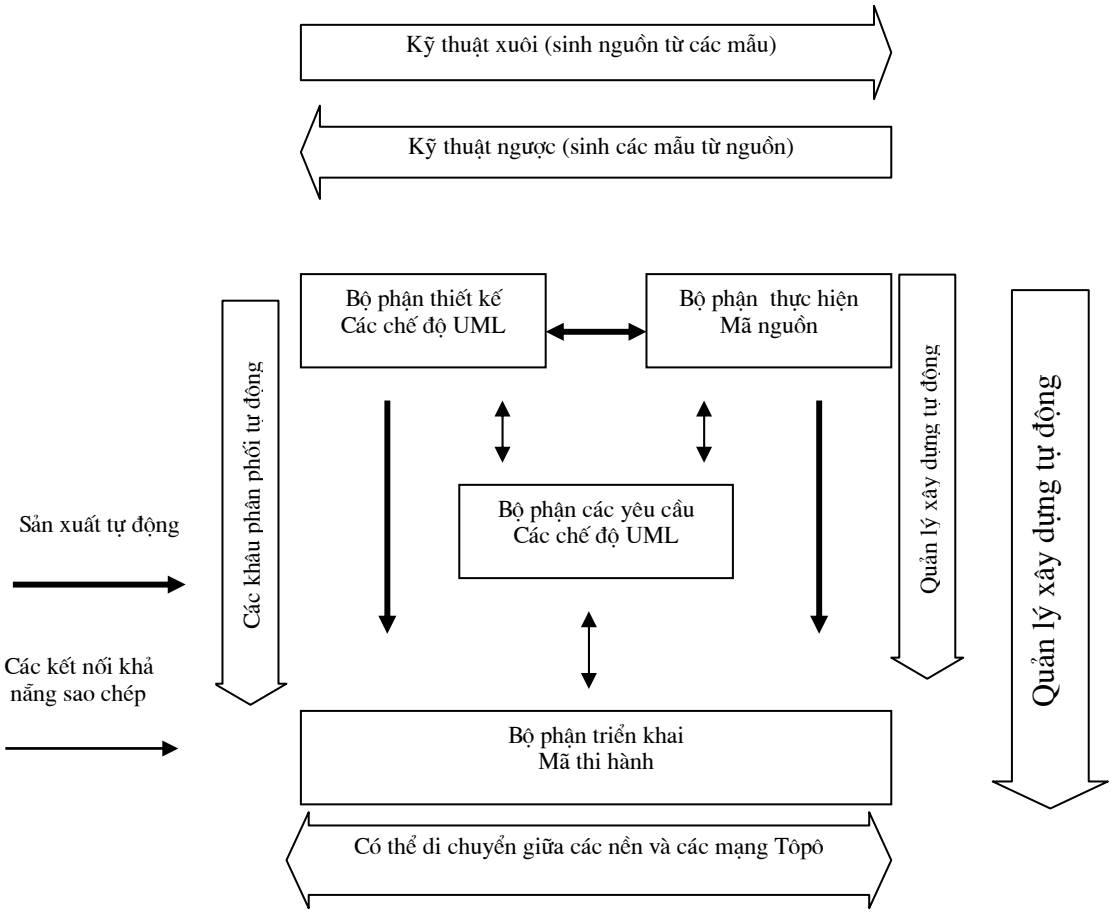
Khi công nghiệp phần mềm chuyển sang lưu trữ thông tin dưới những dạng khác nhau đối với những hiện tượng giả kỹ thuật, hỗ trợ tự động hoá được cần đến nhiều hơn để bảo đảm sự chuyển đổi kiểu dữ liệu một cách có hiệu quả và không bị lỗi từ một hiện tượng giả này đến một hiện tượng giả khác. Kỹ thuật trọn vòng là hỗ trợ môi trường cần thiết để duy trì ổn định giữa những hiện tượng giả kỹ thuật.

Hình 12-2 miêu tả một số chuyển đổi quan trọng giữa các dạng chứa thông tin. Việc thông dịch tự động của mẫu thiết kế sang mã nguồn (cả kỹ thuật tiến tới và đảo ngược) được thiết lập thật sự tốt. Việc chuyển tiếp tự động của mẫu thiết kế sang mẫu tiến trình (sự phân phối) cũng trở nên dễ dàng nhờ các công nghệ vis dụ như ActiveX và Common Object Request Broker Architecture(CORBA)

Trình biên dịch và liên kết đã chịu trách nhiệm tự động chuyển mã nguồn sang mã thi hành. Giống như sự bắt đầu của công trình xây dựng sử dụng các thành phần, nền, ngôn ngữ hỗn tạp, sự phức tạp của toà nhà, kiểm soát và duy trì thanh rầm lớn của các thành phần đưa ra nhiều nhiều nhu cầu mới cho nhà quản lý xây dựng trong tự động hoá và kiểm soát hình dạng. Tuy nhiên, môi trường ngày nay không tự động hỗ trợ để có thể mở rộng tối đa. Một ví dụ, trường hợp kiểm tra tự động tạo nên từ việc sử dụng sự mô tả kịch bản và các trường hợp vẫn chưa có được bất kỳ hỗ trợ nào trừ những ví dụ nhận được, ví như kịch bản kiểm tra đơn vị.

Lý do chính cho kỹ thuật trọn vòng cho phép tự do thay đổi nguồn dữ liệu kiến trúc phần mềm. Việc kiểm soát cấu hình này của tất cả hiện tượng giả kỹ thuật là cốt yếu để duy trì một

sự biểu diễn thích hợp và không lỗi của sản phẩm phát triển. Tuy nhiên không cần thiết có hai hướng chuyển đổi trong mọi trường hợp. Ví dụ, mặc dù chúng ta có thể xây dựng những trường hợp kiểm tra cho các kịch bản đã được định nghĩa cho một tập các đối tượng logic có trước, chúng ta không thể có "kỹ thuật đảo ngược": xây dựng những đối tượng đó chỉ từ những trường hợp kiểm tra. Một cách tương tự, kỹ thuật đảo ngược của mã nguồn xây dựng có tính kế thừa yếu thành mẫu thiết kế hướng đối tượng có thể sản xuất ngược lại.



Hình 12-2 Kỹ thuật trọn vòng

Việc chuyển đổi từ một nguồn dữ liệu này sang một nguồn khác có thể là không hoàn chỉnh 100%. Một ví dụ, việc chuyển đổi những mẫu thiết kế sang mã nguồn C++ có thể cung cấp vẻ bên ngoài về cấu trúc và khai báo của cách biểu diễn mã nguồn. Các thành phần mã vẫn cần phải được bổ sung đầy đủ những gì cụ thể của các phương thức và thuộc tính đối tượng.

12.2.2. Quản lý sự thay đổi(change management)

Quản lý thay đổi là việc lập kế hoạch giống như giới hạn cho những tiến trình lặp lại. Theo dõi thay đổi trong những hiện tượng giả kỹ thuật là cốt yếu để hiểu được xu hướng phát triển công nghệ thật sự và xu hướng chất lượng tiến tới đưa ra một sản phẩm cuối cùng được

thừa nhận hay phát hành tạm thời. Trong những quy trình quản lý phần mềm thông thường, những kỹ thuật quản lý cấu hình giới hạn cho những hiện tượng giả kỹ thuật là hoạt động sau này chiếm ưu thế vòng đời. Trong một quy trình quản lý hiện đại, trong đó những yêu cầu và các bộ tạo tác thực hiệnsớm đạt được những yêu cầu khắt khe trong vòng đời và phát triển qua nhiều thế hệ - quản lý thay đổi đã trở nên cơ bản đối với tất cả các giai đoạn và hầu như tất cả các hoạt động.

Những thứ tự của sự thay đổi phần mềm*(software change order)*

Những đơn vị nguyên tử của công việc phần mềm được phép phát sinh, thay đổi hoặc bỏ đi các thành phần trong một giới hạn cấu hình được gọi là thứ tự thay đổi phần mềm (SCO). Những thứ tự thay đổi phần mềm là một cơ chế cho việc phân hoạch, cấp phát, lập lịch công việc phần mềm ngược lại một giới hạn phần mềm đã được thiết lập và để đánh giá sự tiến bộ và đánh giá chất lượng. Một ví dụ của SCO đưa ra trên hình 12-3 là một điểm bắt đầu tốt cho việc mô tả một tập các thay đổi cơ bản. Nó chỉ ra các yêu cầu chi tiết để hoàn thành tính nghiêm ngặt của các độ đo và của sự quản lý thay đổi ,cần thiết cho một quy trình phần mềm hiện đại. Bằng cách vào dữ liệu tự động và khả năng duy trì những bản ghi thay đổi một cách trực tuyến, việc bộ máy quản lý thay đổi kết hợp với các độ đo báo cáo những hoạt động cũng có thể được tự động hoá.

Mức độ mà tại đó một SCO được viết vẫn luôn luôn là một vấn đề. Một sự thay đổi riêng biệt là gì? Có phải là sự thay đổi một đơn vị chương trình hay thay đổi một thành phần, một file, một hệ thống con? Có phải là một tính chất mới, một sự khắc phục khuyết điểm, hay một sự nâng cấp thực hiện? Trong hầu hết các dự án, đơn vị nguyên tử của SCO có xu hướng dễ dàng được chấp nhận. Một cách tổng quát, một SCO nên được viết ngược lại một thành phần đơn do đó nó dễ dàng được cấp phát cho một cá thể đơn. Nếu một quyết định yêu cầu hai người trong hai đội khác nhau thì hai SCO riêng biệt nên được viết.

Những trường cơ bản của SCO là tiêu đề, phần mô tả, các ma trận, những quyết định, sự đánh giá và sự sắp xếp.

- Tiêu đề*(title). Tiêu đề được đề xuất bởi người sáng tạo và được hoàn thành nhờ vào sự chấp nhận của bảng kiểm soát cấu hình (CCB)Trường này nên bao gồm một sự tham khảo một bản báo cáo vấn đề phần mềm bên ngoài nếu sự thay đổi được đưa ra bởi người ngoài (ví dụ như người sử dụng)*
- Phần mô tả*(description). Phần mô tả các vấn đề bao gồm tên của người sáng tạo, ngày phát minh, bộ chỉ định SCO được gán CCB, và bộ các chỉ định phiên bản liên quan của phần mềm hỗ trợ. Phần miêu tả những vấn đề nguyên bản nên cung cấp chi tiết nhất có thể những trích dẫn mã gắn vào, in màn hình hiển thị, thông báo lỗi và bất kỳ dữ liệu nào khác có thể giúp tách vấn đề hoặc miêu tả việc thay đổi cần thiết.*

Tiêu đề:_____

Phần mô tả Tên:_____ Ngày:_____ Dự án: _____
Độ đo Phân loại:____(0/1: lỗi, 2: cải tiến, 3: đặc trưng mới ,4: vấn đề khác)
Giải quyết Người phân tích:_____ Thành phần phần mềm:_____
Đánh giá Cách thức:_____(xem xét, phân tích, chứng minh ,kiểm tra) Người kiểm tra:_____ Các nền:_____Ngày:_____
Cách bố trí Trạng thái:_____Phát hành:_____ Ưu tiên_____ Công nhận:_____ Ngày:_____ Kết thúc : _____ Ngày:_____

Hình 12-3 Các thành phần cơ bản của một thứ tự thay đổi phần mềm

- Độ đo*(metric). Những độ đo đã tập hợp cho mỗi SCO là quan trọng cho việc lập kế hoạch, lập lịch trình, và cho việc đánh giá cải tiến chất lượng. Những hạng mục thay đổi là loại 0(lỗi giới hạn), loại 1(lỗi), loại 2(nâng cao), loại 3(đặc tính mới) và loại 4(những cái khác), như sẽ được miêu tả sau trong phần này. Nhờ vào sự chấp nhận của SCO, những đánh giá ban đầu được tạo thành bởi một số đứt đoạn(breakage) và yêu cầu nỗ lực để giả quyết vấn đề. Mục đứt đoạn xác định số lượng thay đổi và mục làm lại(rework) xác định độ phức tạp của sự thay đổi. Mục phân tích xác định số giờ cần thiết để hiểu những thay đổi yêu cầu (phát minh lại, cách ly, và phát hiện lỗi vấn đề nếu thay đổi là loại 0 hoặc 1). Mục thi hành xác định số giờ cần thiết để thiết kế và thi hành quyết định. Mục thử xác định số giờ sử đụng để thử nghiệm quyết định, và mục tài liệu xác định tất cả sự nỗ lực bỏ ra để cập nhật những hiện tượng giả khác ví dụ như sách hướng dẫn người sử dụng hoặc phần mô tả phiên bản.Sự đứt đoạn xác định phạm vi của thay đổi và có thể được định nghĩa trong những đơn vị của SLOC, những điểm hàm, các file, các thành phần hay các lớp. Trong trường hợp của SLOC, một chương trình so sánh file nguồn xác định những khác biệt có thể cung cấp một sự đánh giá của sự đứt đoạn. Nói chung, sự chính xác của số đứt đoạn là tương đối không quan trọng. Những thay đổi giữa 0 và 100 dòng có thể lấy chính xác bằng 10, những thay đổi giữa 100 và 1000 lấy là 100 và cứ như vậy.*
- Quyết định*(resolution). Trường này bao gồm tên của những người có trách nhiệm trong việc thi hiện những thay đổi, những thành phần thay đổi, những ma trận thực sự và một phần mô tả của sự thay đổi. Mặc dù sự đúng đắn mức của các thành phần với một dự án theo dõi những liên quan thay đổi có thể được mô tả bên ngoài, nói chung, mức thấp nhất củatham chiếu thành phần nên được giữ tương đối tại một mức của việc cung cấp cho một cá thể. Một ví dụ, một "thành" phần mà được cung cấp cho một đội không là một tham chiếu đủ chi tiết.*
- Sự định giá*(assessment). Trường này miêu tả kỹ thuật đánh giá như sự kiểm tra, phấn tích, chứng minh hoặc thử. Nơi có thể được ứng dụng cũng nên tham chiếu tất cả các trường hợp thử nghiệm tồn tại và những trường hợp thử nghiệm mới được thực hiện, và cũng nên xác định tất cả các cấu hình thử nghiệm khác nhau, ví dụ như các nền, cấu trúc liên kết và các trình biên dịch.*
- Sự xếp đặt(deposition). SCO được gán với một trong những trạng thái sau đây bởi CCB
  - *Được đề nghị : viết ra, xem xét lại CCB chưa giả quyết*
  - *Được chấp nhận: CCB được chấp nhận cho cách giải quyết*
  - *Bị bác bỏ: đóng với những lý do căn bản ví dụ như không một vấn đề, bản sao, sự thay đổi quá hạn, giả quyết bởi SCO khác*
  - *Được lưu trữ: được chấp thuận nhưng bị hoãn lại cho đến khi có một sự đưa ra sau*

- *Đang tiến hành: đã được phân công và đang được giải quyết bởi một tổ chức phát triển.*
- *Đang đánh giá: đã được giải quyết bởi một tổ chức phát triển; đang được đánh giá bởi một tổ chức thử nghiệm*
- *Đóng: được giải quyết hoàn chỉnh với sự hợp tác của tất cả bộ phận CCB*

Một bộ phận chỉ định quyền ưu tiên và nhả ra cũng có thể được phân công bởi CCB để hướng dẫn .. và cách tổ chức của các hoạt động phát triển hiện tại.

Ranh giới cấu hình*(configuration baseline)*

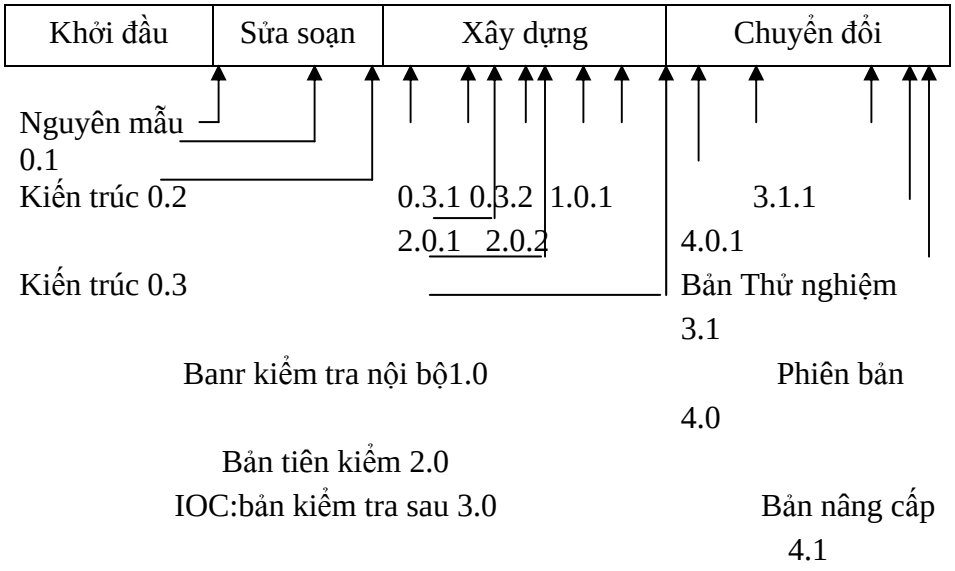
Một ranh giới cấu hình là một bộ tên của các thaành phần phần mềm và những tài liệu hỗ trợ là chủ thể để quản lý sự thay đổi và được nâng cấp, duy trì, thử nghiệm, như là một đơn vị. Với những hệ thống quản lý cấu hình phức tạp, có nhiều tiêu chuẩn phạm vi dặc biệt và dự án đặc biệt.

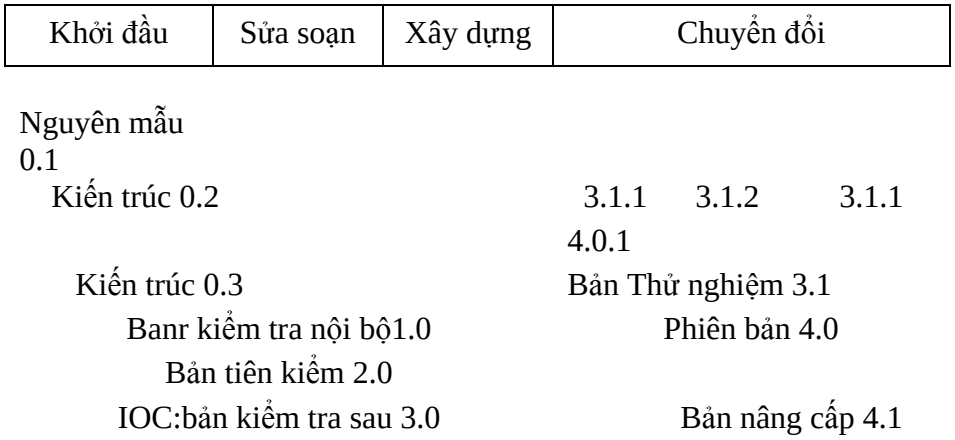
Nói chung có hai lớp của ranh giới: phiên bản sản phẩm ngoài và phiên bản kiểm tra trong. Một ranh giới cấu hình là một tập tên của các thành phần được coi như là một đơn vị. Nó được quản lý một cách chính thức bởi vì nó sự trao đổi được đóng gói giữa các nhóm. Một ví dụ, tổ chức phát triển có thể đưa ra một ranh giới cấu hình cho tổ chức kiểm tra hoặc thậm chí là cho bản thân tổ chức đó. Một dự án có thể đưa ra một ranh giới cấu hình cho người sử dụng để kiểm tra.

Nói chung, ba mức của ranh giới đưa ra là yêu cầu cho mọi hệ thống: lớn, nhỏ và tạm thời. Mỗi mức tương ứng với một từ định danh ví dụ như N, M, X, trong đó N là lớn, M là nhỏ và X là từ định danh cho mức tạm thời. Một phiên bản chính biểu diễn một thế hệ mới của sản phẩm hay dự án, trong khi một phiên bảm phụ biểu diễn cùng một sản phẩm cơ sở nhưng với những đặc điểm, hiệu suất, chất lượng được đề cao. Phiên bản chính và phụ được dùng như các phiên bản sản phẩm bên ngoài mà liên tục và được hỗ trợ trong một thời kỳ. Một phiên bản trong lúc chờ đọi tương ứng với một cấu hình tiến triển được dùng tạm thời. Vòng đời của nó càng ngắn càng tốt. Hình 12-4 chỉ ra một ví dụ của một số tên phiên bản cho hai trường hợp khác nhau.

Một khi phần mềm được thiết lập trong ranh giới kiểm soát, tất cả sự thay đổi được theo dõi. Sự khác biệt chỉ là nguyên nhân của một sự thay đổi. Những loại thay đổi giống như sau:

- *Loại 0: thiếu khả năng tới hạn, đó là những thiếu sót gần như luôn luôn được sửa chữa trước bất ky một phiên bản ngoài .Nói chung, loại của các thay đổi được biểu diễn showstopper có một sự ảnh hưởng lên khả năng sử dụng của phần mềm trong những trường hợp sử dụng tới hạn của nó.*
- *Loại 1: Một con bọ hay một thiếu sót không làm giảm khả năng sử dụng của hệ thống hoặc có thể bị vòng. Những lỗi như vậy có thể gây ra phiền toái trong những trường hợp sử dụng tới hạn hoặc những lỗi nghiêm trọng trong những trường hợp sử dụng phụ là những trường hợp có khả năng xuất hiện thấp.*





Hình 12-4 Những thí dụ về lịch sử công bố đối với các dự án cụ thể và các sản phẩm phần mềm

- *Loại 2: Một sự thay đổi mà là một cách làm nổi bật hơn là một hỏng hóc. Mục đích của nó là cải thiện hiệu suất, khả năng thử nghiệm, khả năng sử dụng hoặc những mặt khác của chất lượng mà là một điển hình của kỹ thuật tốt.*
- *Loại 3: Một sự thay đổi cần có để cập nhật những yêu cầu. Đây có thể là những khả năng và tính chất mới nằm ngoài phạm vi của phiên bản và giao dịch hiện tại*
- *Loại 4: Một thay đổi chưa được dàn xếp giữa các hạng mục. Những ví dụ chỉ gồm có tài liệu hoặc những phiên bản nâng cấp cho các thành phần thương mại.*

Bảng 12-1 đưa ra những ví dụ của những sự thay đổi này trong phạm vi của hai của hai dự án khác nhau: phạm vi lớn là hệ thống kiểm soát giao thông hàng không tin cậu và phạm vi nhỏ là công cụ phát triển phần mềm.

Bảng kiểm soát cấu hình*(Confirguration control board)*

Một CCB là một đội cán bộ mà chức năng như là có quyền quyết định chấp nhận những ranh giới cấu hình. Một CCB thường bao gồm giám đốc phần mềm, giám đóc thiết kế phần mềm, giám đốc phát triển phần mềm, giám đốc định giá phần mềm và những người khác (khách hàng, giám đốc dự án phần mềm, kỹ sư hệ thống, người sử dụng) đó là những người được kết hợp để sự duy trì một hệ thống phân phối kiểm phần mềm kiểm soát.Trong khi các CCB hoạt động qua trao đổi bảng, sự phân phối trực tuyến, hiệ tại, và sự chấp thuận của hoạt động CCB có thể tạo ra khả năng đánh giá dưới rất nhiều thực trạng dự án.

Loại thay đổi	*Dự án kiểm soát giao thông hàng không*	*Công cụ mẫu trực quan đóng gói*
Loại 0	*Kiểm soát dữ liệu về quá hạn và lỡ chuyến bay*	*Sự mất mát của dữ liệu người sử dụng*
Loại 1	*Hiển thị thời gian đáp ứng nhiều hơn yêu cầu 0.5 giây*	*Mở rộng trình duyệt nhưng không phá vỡ các điểm vào hiển thị*
Loại 2	*Thêm trường thông tin bên trong cho máy thời gian đáp ứng*	*Sử dụng màu sắc để phân biệt sự nâng cấp từ những phiên bản trước đó của những mẫu trực quan*
Loại 3	*Tăng khả năng quản lý giao thông hàng không từ 1200 đến 2400 chuyến bay mô phỏng*	*Cổng cho những nền mới như WinNT*
Loại 4	*Nâng cấp từ Oracle 7 lên 8 để cải thiện hiệu suất yêu cầu*	*Ngoại lệ tăng khi giao diện MSExcel 5.0 bởi vì quản lý tài nguyên Windown*

Khái niệm hoạt động của một tiến trình phát triển lặp lại phải bao gồm quản lý sự thay đổi chặt chẽ và mang tính toàn diện của những ranh giới phần mềm mở ra. Tiến trình cơ bản cho việc kiểm soát sự phát triển phần mềm và duy trì những hoạt động được mô tả thông qua một dãy các trạng thái được duyệt bởi một SCO. Những từ (trong ngoặc) tạo thành các trạng thái của một SCO chuyển tiếp qua một quá trình.

- *[Được đề nghị(propose)]. Một sự thay đổi được đề nghị được phác thảo và CCB chấp nhận. Sự thay đổi được đề nghị phải bao gồm một phần miêu tả kỹ thuật của vấn đề và đánh giá hiệu quả của quyết định.*

- *[Được chấp thuận, được hoàn thành hay bị loại bỏ]. CCB phân một bộ chỉ định duy nhất và chấp thuận, hoàn thành hay loại bỏ mỗi sự thay đổi được đề nghị. Sự chấp thuận bao gồm sự thay đổi của quyết định trong phiên bản tương lai; và sự loại bỏ với những sự thay đổi được đề nghị khác hoặc nằm ngoài phạm vi. CCB kiểm tra tất cả các trường SCO cho phù hợp và chính xác trước khi chấp nhận SCO, sau đó phân SCO cho một người có trách nhiệm trong sự tổ chức phát triển cho quyết định.*
- *[Trong tiến trình]. Người có trách nhiệm phân tích, thực hiện và kiểm tra một giải pháp phù hợp với SCO. Nhiệm vụ này bao gồm cập nhật tài liệu, đưa ra chú thích, và ma trận SCO thật sự, và chấp nhận những SCO mới, nếu cần thiết. Thông quan hoàn thành một giải pháp hoàn chỉnh, người có trách nhiệm hoàn thành bộ phận quyết định của SCO và đưa cho đội thử nghiệm độc lập để định giá.*
- *[Trong định giá]. Đội thử nghiệm độc lập đánh giá nếu SCO được giải quyết hoàn toàn. Khi đội thử nghiệm độc lập nghĩ rằng sự thay đổi được giải quyết ổn thoả, SCO được đưa tới cho CCB để sắp xếp và đóng gói cuối cùng.*
- *[Đóng] Khi tổ chức phát triển, tổ chức thử nghiệm độc lập và CCB tán thành rằng SCO đã được giải quyết, nó sẽ chuyển sang tình trạng đóng.*

12.2.3 Cơ sở hạ tầng

Từ viễn cảnh tự động hoá cho quá trình, cơ sở hạ tầng của tổ chức cung cấp những tài sản cố định, bao gồm hai mầu chính: một co chế để đạt được những tiêu chuẩn cho quá trình phát triển phần mềm dự án, và một môi trường để đạt được bản kê khai các công cụ. Những công cụ này là những khối xây dựng sự tự động hoá mà từ đó môi trường dự án có thể được định dạng một cách hiệu quả và kinh tế.

Chính sách tổ chức*(organization policy)*

Chính sácch tổ chức thường được đóng gói như là một quyến sách tra cứu để định nghĩa vòng đời, những nguyên thuỷ quát trình (những cột mốc chính, những mầu trung gian, những kho chứa kỹ nghệ, những độ đo, vai trò và trách nhiệm). Sách tra cứu cung cấp một khung chính để trả lời những câu hỏi sau:

- *Cái gì được làm? (những hoạt động và những mẫu)*
- *Khi nào nó được làm? (ánh xạ tới những pha vòng đời và những cột mốc)*
- *Ai làm nó? (vai trò và trách nhiệm của đội)*
- *Làm thế nào chúng ta biết nó đủ? (những điểm kiểm tra, các độ đo và những tiêu chuẩn của hiệu suất)*

Nhu cầu cho sự cân bằng là một sự xem xét quan trọng trong định nghĩa chính sách tổ chức. Một cách thường xuyên, các tổ chức kết thúc tại một trong hai cực: không quá trình nào

được hình thành, hoặc quá nhiều sự tiêu chuẩn hoá và hệ thống hành chính. Những chính sách tổ chức hiệu quả có một số những đề tài định kỳ:

- *Những đề tài này ngắn gọn và tránh những phát biểu chính sách mà phải mất một tập tài liệu dầy 6 inch mới đủ.*
- *Những đề tài này giới hạn những chính sách cho sball, sau đó ép buộc chúng*
- *Những đề tài này tránh sử dụng từ should trong những phát biểu chính sách. Hơn nữa một thực đơn (menu) của những tùy chọn cần một bộ ngắn gọn các tiêu chuẩn uỷ nhiệm.*
- *Những sự từ bỏ là ngoại lệ, không phải là luật*
- *Những chính sách phù hợp được viết tại mức độ phù hợp*
- *Điểm cuối cùng xứng đánh được thảo luận sau hơn. Có rất nhiều cấu trúc tổ chức khác nhau ở toàn bộ công nghiệp phát triển phần mềm. Hỗu hết các ông ty phần mềm cường độ cao có ba mưc tổ chức riêng biệt, với một chính sách khác nhau tại mỗi lớp:*
- *Mức tổ chức cao nhất: những tiêu chuẩn dùng để đẩy mạnh (1) chiến lược và những cải tiến cho quá trình dài hạn, (2) công nghệ chung đưa vào và giáo dục, (3) khả năng so sánh của dự án và hiệu suất đơn vị doanh nghiệp, và (4) kiểm soát chất lượng uỷ nhiệm.*
- *Mức độ dòng của doanh nghiệp trung bình: những tiêu chuẩn dùng để đẩy mạnh (1) chiến thuật và những cải tiến cho những quá trình ngắn hạn, (2) công nghệ trên lĩnh vực đặc biệt đưa vào và huấn luyện, (3)sử dụng lại các thành phần, các quá trình, dầo tạo, những công cụ, và kinh nghiệm cá nhân, và (4) chấp nhận những tiêu chuẩn tổ chức.*
- *Mức dự án thấp nhất: những tiêu chuẩn dùng để nâng cao hiệu quả trong chất lượng đạt được, lịch trình, và những đích giá, (2) đào tạo dự án đặc biệt, (3) chấp nhận những yêu cầu của khách hành, và (4) và tuân theo những tiêu chuẩn đơn vị tổ chức/doanh nghiệp.*

Tiêu chuẩn hoá nên tập trung vào các đơn vị dòng của doanh nghiệp, không phải vào mức độ tổ chức hay dự án. Đòn bẩy từ tiêu chuẩn hoá là hiệu quả hơn tại mức đơn vị doanh nghiệp, nơi mà có hầu hết … và tái sử dụng qua những dự án, những quá trình, và những công cụ. Tiêu chuẩn hoá của những kỹ thuật và công cụ phát triển phần mềm qua những dòng cua doanh nghiệp đã chỉ ra là rất khó khăn, bởi vì những tính chất, những công cụ, những phương thức và những văn hoá người giữ tiền đặ cọc của quá trình có thể rả khác biệt. Nỗ lực tiêu chuẩn hoá qua các lĩnh vực có ít kết quả chung cũng phát biểu chính sách sơ lược hoặc quá trình của sự huỷ bỏ được sử dụng thường xuyên. Tiêu chuẩn hoá tại mức độ quá cao là một nghi vấn; như vậy đang tiêu chuẩn hoá tại mức quá thấp. Nếu tất cả các đội của dự án phải bỏ đi những thiết bị của bản thân họ, tất cả tiến trình dự án và môi trường dự án sẽ được tối ưu

hoá địa phương. Qua thời gian, cơ sở hạ tầng của tổ chức cho dự phát triển và cải tiến quá trình sẽ bào mòn.

Chính sách tổ chức là tài liệu định nghĩa cho những chính sach phần mềm của tổ chức. Trong bất kỳ sự định giá quá trình, đây là tạo tác hữu hình nêu lên cái bạn cần. Từ tài liệu này, các nhà phê bình có thể xem xét và hỏi các nhân sự của dự án và xác định nếu tổ chức làm cái nó nêu ra. Hình 12-5 đưa ra một phác thảo chung cho một chính sách tổ chức.

I. Process-primitive definitions
   A. Life-cyrcle phases (inception,elaboration,construction,transition)
   B. Checkpoints....
   C. Artifact....
   D. Roles...

II. Organizational..
   A.
   B.
   C.
   D.
   E.
   F.
   G.
   H.

III. Walve policy

IV. Appendixes
   A.
   B.

H×nh 12.5 D¹ng cña chÝnh s¸ch tæ chøc

Môi trường tổ chức*(organization environment)*

Môi trường tổ chức cho sự tự động hoá quá trình mặc định sẽ cung cấp rất nhiều câu trả lời làm cách nào có thể thực hiện cũng như những kỹ thuật và công cụ cho tự động hoá quá trình nhiều như thực hành. Một số thành phần điển hình của những khối xây dưng sự tự động hoá của một tổ chức giống như sau:

- *Những chọn lọc công cụ tiêu chuẩn hoá (thông qua đầu tư bởi tổ chức trong giấy phép hoặc sự điều đình của chiết khấu thuận lợi với những nhà cung cấp công cụ do đó các đội của dự án được thú đẩy về mặt kinh tế để sử dụng những công cụ đó), dùng thúc đẩy những luồng công việc chung và ROI cao hơn trong đào tạo.*
- *Khái niệm tiêu chuẩn cho những mẫu, ví dụ như UML, cho tất cả các mẫu thiết kế, hoặc Ada 95 cho tất cả các phong tục phát triển, những mẫu thực hiện giới hạn tin cậy.*

- *Những phụ trợ công cụ ví dụ như khung mẫu hiện tượng giả hiện hữu (phần mô tả kiến trúc, tiêu chuẩn đánh giá, những mô tả phiên bản, những sự định giá tình trạng) hoặc sự thực hiện theo yêu cầu của khách hàng.*
- *Khung mẫu hoạt động (kế hoạch lặp lại, các hoạt động cột mốc chính, các bảng kiểm soát cấu hình)*
- *Những thành phần hữu dụng gián tiếp khác của một cơ sở hạ tầng của một tổ chức*

- *Một thư viện tham khảo của những kinh nghiệm tiền lệ cho việc lập kế hoạch, định giá, và cải tiến các thông số hiệu hiệu suất quá trình; trả lời các câu hỏi Như thế nào thì tốt? Nhiều như thế nào? Tại sao?*
- *Nghiên cứu các trường hợp hiện hữu, bao gồm các dấu chuẩn khách quan của hiệu suất cho những dự án thành công theo sau quá trình tổ chức.*
- *Một thư viện của những ví dụ mẫu dự án ví dụ như kế hoạch phát triển phần mềm, những mô tả kiến trúc, và các kinh nghiệm định giá tình trạng.*
- *Những kiểm toán giảvà khả năng theo dõi theo yêu cầu cho luồng công việc định giá quá trình bên ngoài ví dụ như Engineering Software Institute's Capability Maturity Model (SEI CMM)*

Những môi trường của người liên quan*(stakeholder environment)*

Sự chuyên tiếp tới một quá trình phát triển lặp lại hiện đại với hỗ trợ tự động hoá không nên hạn chế với đội phát triển. Rất nhiều dự án bằng hợp đồng lớn bao gồm những người trong các tổ chức ngoài đại diện những người liên quan khác tham gia trong quá trình phát triển. Họ có thể gồm cả sự tìm kiếm hợp đồng đại lý theo dõi, nhân sự hỗ trợ kỹ thuật người sử dụng cuối, đại diện của những đại lý và những người khác.

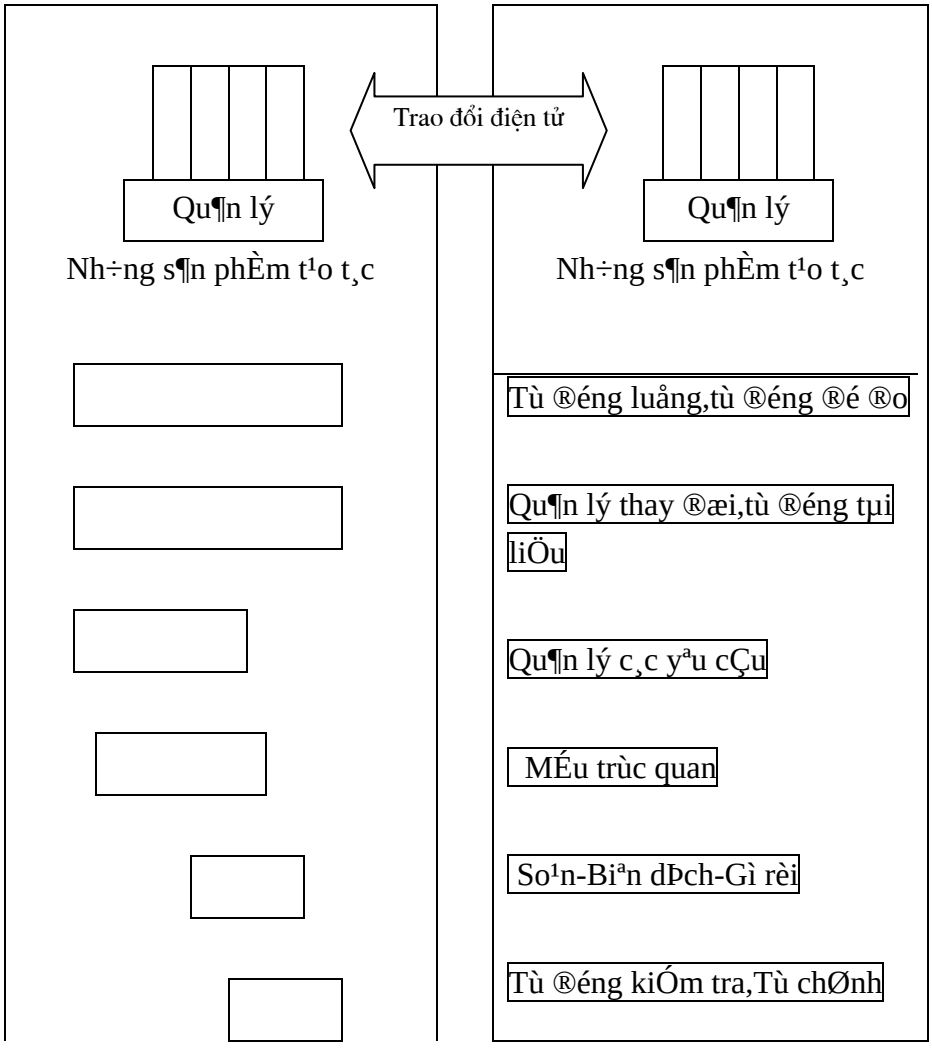
Những đại diện người liên quan này cũng cần truy nhập tài nguyên môi trường phát triển do đó họ có thể đóng góp giá trị tới toàn bộ nỗ lực. Nếu một nhóm người liên quan bên ngoài không có tài nguyên môi trường để chấp nhận sản phẩm và tạo tác trực tuyến thì phương tiện để trao đổi thông tin chỉ là giấy tờ. Tình trạng này sẽ là kết quả của những vấn đề đã mô tả trong chương 6 giống như cố hữu trong quá trình truyền thống.

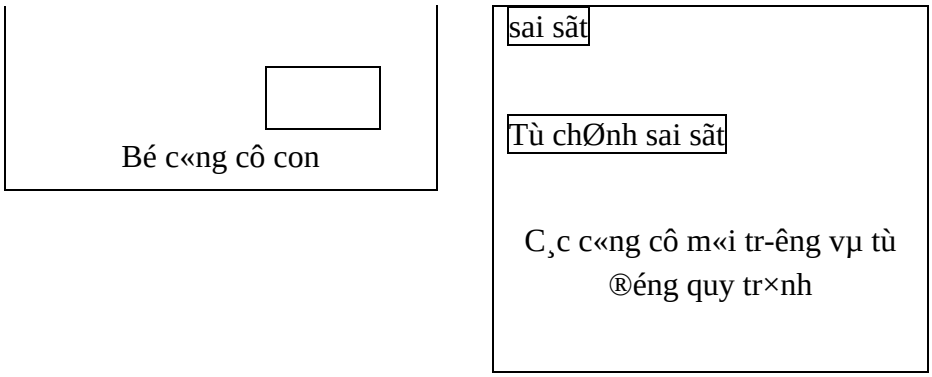
Một khả năng truy nhập môi trường trực tuyến bởi những người liên quan bên ngoài cho phép họ tham gia trong những quá trình sau:

- *Chấp thuận và sử dụng sự phát triển khải năng thực hiện cho việc đánh giá tay-trên*
- *Sử dụng các công cụ trực tuyến, dữ liệu, và các bản báo cáo giống như các tổ chức phát triển phần mềm sử dụng để quản lý và điều khiển dự án*
- *Tránh du lịch quá mức, trễ nải thay đổi giấy tờ, các chuyển đổi khuôn dạng, chi phí vận chuyển và chi phí giấy tờ và các chi phí đau đầu khác.*

Hình 12-6 minh hoạ mộ số cơ hội mới cho các hoạt động giá tri thêm vào bởi những người giữ tiền đặ cọc ngoài trong nỗ lực hợp đồng lớn. Đây là một số lý do quan trọng để mở rộng tải nguyên môi trường phát triển vào phạm vi người giữ tiền đặ cọc cụ thể.

- *Các mầu kỹ thuật không chỉ là giấy tờ. Các mẫu điện tử trong các ký hiệu nghiêm ngặt ví dụ như các mô hình trực quan và mã nguồn được quan sát hiệu qảu hơn bằng cách sử dụng các công cụ với trình duyệt tốt.*
- *Sự định giá độc lập của những mẫu liên quan được khuyến khích bằng chỉ đọc điện tử truy cập dữ liẹeu trực tuyến ví dụ như các thư viện giới hạn cấu hình và cơ sở dữ liệu quản lý sự thay đổi. Xem lại và kiểm tra, định giá sự đứt đoạn/làm lại, phân tích độ đo thậm chí kiểm nghiệm bêta có thể được thực hiện một cách độc lập của với những đội ngũ phát triển*
- *Thậm chí tài liệu giấy tờ nên được chuyển giao một cách điện tử để giảm chi phí sản phẩm và thời gian khép kín.*





Hình 12-6 Môi trường mở rộng vào phạm vi của nhữngngười liên quan

Một khi các tài nguyên môi trường có thể truy cập điện tử bởi người giữ tiền đặ cọc, phản hồi thủ đoạn và liên tục là hiệu quả hơn, hữu hình hơn, và hữu dụng hơn. Trong việc thực thi ví dụ như một môi trường dùng chung, nó rất quan trọng cho đội phát triển để tạo ra một mô trường mở và cung cấp tài nguyên đầy đủ để sắp xếp khách hàng truy cập. Nó cũng quan trọng cho những người giữ tiền đặt cọc tránh lạm dụng sự truy cập này, tham gia bởi các giá trị thêm vào, và tránh gián đoạn phát triển. Công nghệ internet và intranet làm kinh tế và ít giấy tờ các môi trường.

Mở rộng tài nguyên môi trường vào phạm vi người giữ tiền đặ cọc đưa ra một số vấn đề. Có bao nhiêu truy cập miễn phí được cung cấp? Ai tài trợ đầu tư công cụ và môi trường? Làm thế nào bảo mật thông tin trao đổi? Làm cách nào đồng bộ quản lý sự thay đổi? Một số sự thay đổi văn hoá được thảo luận trong chương 17.

Chương 13
Kiểm soát dự án và Công cụ xử lý

Đề tài chính của một tiên trình phát triển phần mếm kiểu mới khắc phục sự quản lý tập trung của phần mềm phức tạp:

1 . Đạt được đúng thiết kế bởi chú trọng đến kiến trúc trước tiên.

2. Quản lý sự rủi ro qua phát triển lặp.

3. Giảm độ phức tạp với các kĩ thuật dựa trên thành phần.

4. Tạo sự tiến triển phần mềm và chất lượng xác thực thông qua sự quản lý thay đổi được công cụ

Các điểmchính

- Tiến độ mục tiêu dự án và chất lượng của những sản phẩm phần mềm phải đo được suốt chu trình phát triển phần mềm.
- Những giá trị metric cung cấp một viễn cảnh quan trọng để quản lý quá trình. Những khuynh hướng metric cung cấp cái khác.
- Metric hữu ích nhất trực tiếp được rút ra từ sự tiến triển những hiện tượng giả.
- Sự phân tích khách quan và

thu thập dữ liệu được tự động hóa là thiết yếu tới sự thành công của bất kỳ chương trình metric nào.Những đánh giá chủ quan và các kĩ thuật thu thập thủ công hầu như thất bại.

5. Tự động các hoạt động chi phí và kế toán qua cách dùng kĩ sư khứ hồi và môi trường tích hợp.

Vấn đề thứ tư là chủ đề của chương này. Thật là khó khăn để quản lý cái gì không thể đo được một cách khách quan. Đây là một trong những vấn đề chính với tiến trình phần mềm thông thường, ở đây những sản phẩm trung gian phần lớn là giấy tờ. Metrics phần mềm phân phối các hoạt động và sản phẩm của tiến độ phát triển và tích hợp phần mềm. Bất kì tiến trình phần mềm mà metrics được chi phối bởi các thủ tục thao tác thủ công và các hoạt động tập trung nhân lực sẽ giới hạn thành công. Trong tiến trình phát triển kiểu mới, phần lớn các sotfware metrics là đơn giản, thước đo mục tiêu của viễn cảnh khác nhau như thế nào của sản phẩm và dự án là thay đổi

Chất lượng sản phẩm phần mềm và các quá trình đưa đến kết quả dự án phải đo được thông qua chu trình phát triển phần mềm. Mục đích của metrics phần mềm là cung cấp nhóm phát triển và nhóm quản lý như sau:

- Một quá trình ước tính chính xác ngày tháng
- Nhìn được chất lượng của sự tiến triển sản phẩm phần mềm
- Cơ sở để ước tính chi phí và lịch thực hiện để hoàn thành sản phẩm với độ chính xác tăng qua thời gian.

13.1.Bảy metrics cơ bản

Nhiều metrics khác nhau có thể có giá trị trong việc quản lý tiến trình kiểu mới. Tôi đã sắp đặt bảy metrics cơ bản nên được dùng trên tất cả các dự án phần mềm. Trong đó, ba là biểu thị sự quản lý và bốn là biểu thị chất lượng.

Biểu thị quản lý

Công việc và sự tiến hành (công việc được thực hiện qua thời gian)

Chi phí ngân sách và phí tổn (chi phí phải chịu qua thời gian)

Nhân viên và đội ngũ động (thay đổi nhân sự qua thời gian)

Biểu thị chất lượng

Thay đổi sự lưu thông và ổn định

Sự đứt đoạn và mô đun hoá (sự đứt đoạn trung bình mỗi thay đổi theo thời gian)

Làm lại và tính tương thích (làm lại trung bình mỗi thay đổi theo thời gian)

Thời gian trung bình giữa các thất bại (MTBF) và tỉ lệ sai sót qua thời gian

Bảng 13-1 mô tả các software metrics cơ bản. Mỗi metric có hai chiều: một *giá trị* tĩnh được dùng như là mục tiêu, và một *khuynh hướng* động được dùng để quản lý sự thu được của mục tiêu đó. Trong khi các giá trị metrics cung cấp một chiều của tầm nhìn, khuynh hướng metrics cung cấp một triển vọng quan trọng hơn để quản lý tiến trình. Các khuynh hướng metrics với đối với thời gian cung cấp tầm nhìn tiến trình và sản phẩm đang tiến triển như thế nào. Phát triển lặp là về các thay đổi quản lý và các thay đổi thước đo là khía cạnh quan trọng của chương trình metrics. Các giá trị tuyệt đối của tính sản phẩm và cải thiện chất lượng là vấn đề thứ hai tới khi mục đích căn bản của quản lý đã đạt được: giá trị tiên đoán được và hiệu quả kế hoạch làm việc cho mức cho trước của chất lượng.

Table: 13.1 Nhìn chung về bảy metric cơ bản

Metric	Mục đích	Viễn cảnh
Công việc và tiến độ	Đặt kế hoạch lặp lại, kế hoạch so với thực tế, biểu thị sự quản lý	SLOC, các điểm chức năng, các điểm mục tiêu, kịch bản, các trường hợp thử nghiệm, SCOs
Giá dự toán và các chi phí	Nắm được tài chính, kế hoạch so với thực tế, biểu thị sự quản lý	Chi phí mỗi tháng, nhân viên làm cả ngày mỗi tháng, phần trăm dự tính chi phí.
Bố trí nhân viên và đội ngũ động	Kế hoạch nhân lực so với thực tế, mức thuê và mức tiêu hao	Lượng người thêm mỗi tháng, lượng người chuyển đi mỗi tháng
Lưu lượng thay đổi và độ ổn định	Kế hoạch lặp lại, biểu thị sự quản lý của sự quy tụ lịch trình	SCOs mở so với SCOs đóng bởi các dạng (0,1,2,3,4), bởi phiên bản / thành phần / hệ thống con
Sự đứt đoạn và tính mô đun hoá	Sự quy tụ, chia nhỏ phần mềm, biểu thị chất lượng	Thực hiện lại SLOC mỗi tháng bởi các dạng (0,1,2,3,4), bởi phiên bản / thành phần / hệ thống con
Thực hiện lại và tính tương thích	Sự quy tụ, làm lại phần mềm, biểu thị chất lượng	Trung bình số giờ mỗi thay đổi bởi các dạng (0,1,2,3,4), bởi phiên bản / thành phần / hệ thống con
MTBF và kỳ hạn	Kiểm tra phạm vi/tính thích hợp, độ tin cậy để sử dụng, biểu thị chất lượng	Đếm số thất bại, số giờ thử nghiệm tới khi thất bại, bởi phiên bản/thành phần / hệ thống con

Phụ lục C cung cấp một sự bắt nguồn ngắn gọn và mô tả chi tiết những metrics này. Chúng đã được chứng tỏ trong thực hành trên các dự án sử dụng phát triển lặp. Trường hợp xem xét trong phụ lục D đưa ra một mô tả rất chi tiết về cách mà những metrics có thể triển khai trên một dự án thực tế.

Bảy metrics cơ bản có thể được dùng bằng nhiều cách để hỗ trợ quản lý các dự án và cơ cấu. Trong một dự án phát triển lặp hoặc một cách cơ cấu được cấu trúc xung quanh dòng phần mềm kinh doand, giá trị mang tính lịch sử của các vòng lặp và dự án trước cung cấp dữ liệu mẫu cho việc hoạch định các vòng lặp và dự án tiếp theo. Do vậy, một khi nhóm các metrics được thu thập, một dự án hoặc một cơ cấu có thể cải thiện khả năng của nó để định liệu chi phí, kế hoạch hoặc hiệu quả chất lượng của các hoạt động tương lai.

Bảy metrics cơ bản được dựa trên cảm nhận chung và kinh nghiệm với các chương trình metrics thành công hoặc không thành công. Các thuộc tính của chúng như sau:

Chúng đơn giản, khách quan, dễ tập hợp, dễ giải thích và khó để giải thích nhầm.

Sự thu thập có thể được tự động và không can thiệp.

Chúng cung cấp cho các nhận định chắc chắn thông qua vòng đời và được bắt nguồn từ những chặng đường phát triển sản phẩm cơ bản hơn là từ nhận định chủ quan.

Chúng có ích đối với cả quản lý và kỹ thuật cá nhân trong tiến trình giao tiếp và chất l-ượng trong một khuôn dạng nhất định.

Tính đúng đắn của chúng cải thiện qua vòng đời.

Thuộc tính cuối cùng là quan trọng và đáng để đề cập tiếp. Metric được áp dụng cho giai đoạn kỹ thuật (chi phối bởi các quyết định tự do và quyết tâm mạo hiểm) sẽ ít chính xác hơn những metric đợc áp dụng cho giai đoạn sản xuất (chiếm giữ bởi các hoạt động thực hiện và thay đổi quản lý). Do đó, metric đợc qui định là đợc dành cho giai đoạn kỹ thuật, khi chi phí mạo hiểm cao và giá trị quản lý đợc nâng lên. Hoạt động metric trong suốt giai đoạn kỹ thuật đợc chuyển phần lớn về việc thiết lập những mốc ban đầu và những mong muốn trong kế hoạch giai đoạn sản xuất.

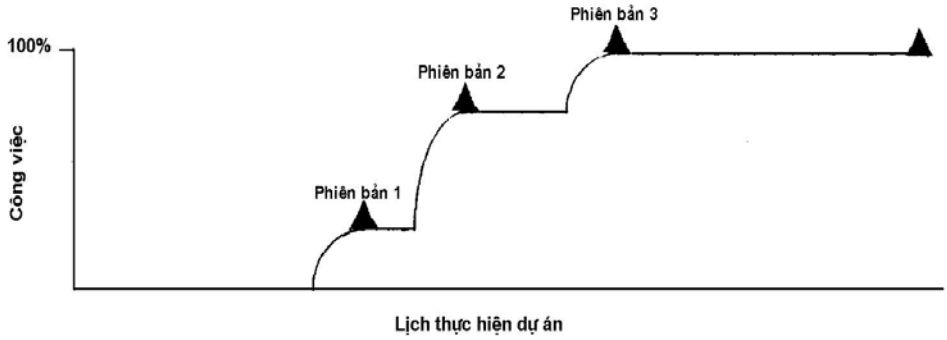
13.2.Biểu thị quản lý

Có ba tập hợp nền tảng của metric quản lý: kỹ thuật tiến triển, tình trạng tài chính, và tiến độ bố trí nhân viên. Bằng việc rà soát những triển vọng này, ban quản lý có thể nhận định chung một dự án là trên dự toán hay trên lịch thực hiên. Tình trạng tài chính rất dễ hiểu, như nó vốn vậy. Phần lớn người quản lý biết chi phí tài nguyên bằng giá cả và lịch thực hiện. Vấn đề là bao nhiêu tiến độ kĩ thuật đã được thực hiện. Những dự án truyền thống mà các sản phẩm trung gian là tất cả các tài liệu giấy tờ dựa trên nhận định chủ quan của tiến độ kĩ thuật hoặc được đo bằng số các tài liệu đã hoàn thành. Trong khi những tài liệu này phản ánh tiến độ trong việc chi tiêu, chúng rất không có vẻ biểu thị bằng công việc hữu ích đã được tiến hành.

Các biểu thị quản lý giới thiệu ở đây bao gồm các trạng thái tài chính chuẩn dựa trên một hệ thống giá trị thu được, những metric tiến độ kĩ thuật khách quan áp đặt tới tiêu chuẩn đo đầu tiên cho từng đội ngũ thuộc cơ cấu, và các metric bố trí nhân viên phải cung cấp cái nhìn sâu bên trong đội ngũ động.

13.2.1.Công việc và tiến độ

Các hoạt động đa dạng của một dự án phát triển lặp có thể được đo bằng việc định nghĩa một ước tính công việc trong một metric khách quan, sau đó theo dõi tiến độ (công việc hoàn thành qua thời gian) đối với kế hoạch đó(Hình 13-1). Từng đội có tổ chức chính có ít nhất một viễn cảnh tiến độ căn bản để đo mỗi đội. Đối với các



Hình 13-1: *Tiến độ mong muốn cho một dự án thông thường với ba phiên bản chính*

đội chuẩn được đề cập trong chương 11, viễn cảnh mặc định của metric này sẽ là như sau:

• Đội kiến trúc phần mềm : dùng những trường hợp đã chỉ dẫn

• Đội ngũ phát triển phần mềm: SLOC dưới đường gốc thay đổi quản lý

• Đội ngũ đánh giá phần mềm: SLOC mở, thời gian tiến hành thử nghiệm hợp với chuẩn đánh giá

• Đội ngũ quản lý phần mềm : những cột mốc đã hoàn thành

13.2.2 Giá dự toán và chi phí

Để bảo trì điều khiển quản lý, đo giá những chi phí qua chu trình hoạt động dự án là luôn luôn cần thiết. Thông qua sự sử dụng của metric cho công việc và tiến độ, nhiều sự định giá khách quan hơn của tiến độ kỹ thuật có thể được thực hiện để so sánh với giá những chi phí. Với một quá trình phát triển lặp đi lặp lại, thật quan trọng để lập kế hoạch những hoạt động ngắn hạn (thông thường là một khung thời gian ít hơn sáu tháng) trong chi tiết và bỏ những hoạt động dài hạn như đánh giá thô để được tinh lọc như sự lặp đi lặp lại hiện thời là sự cuộn xuống và việc đặt kế hoạch cho sự lặp đi lặp lại tiếp theo trở thành thiết yếu.

Việc theo dõi tiến độ tài chính thông thường dựa trên một khuôn dạng chỉ định về cơ cấu. Một cách tiếp cận chung tới phép đo hiệu quả tài chính là dùng một hệ thống giá trị được trả,hệ thống cung cấp chi tiết cao giá và lịch trình chi tiết. Điểm yếu chính của nó đối với dự án phần mềm mà có truyền thống không có khả năng để đánh giá tiến độ kỹ thuật (% thực hiện) một cách khách quan và chính xác. Trong khi đây sẽ luôn luôn là trường hợp trong giai đoạn kỹ thuật (của) một dự án,hệ thống giá trị trả trước tỏ ra có hiệu quả cho giai đoạn sản xuất, nơi có độ trung thực cao theo dõi thực tế đới với những kế hoạch và kết quả có thể đoán trước được. Metric cốt lõi khác cung cấp một khung cho dữ liệu sao lưu (qui số lượng được) hiện thực và chi tiết để lập kế hoạch và ghi nhận, đặc biệt trong giai đoạn sản xuất của một dự án phần mềm, khi những chi phí giá và lịch thực hiện cao nhất.

Những quá trình phần mềm hiện đại đáng phải chịu tới phép đo hiệu quả tài chính thông qua một cách tiếp cận giá trị cho trước. Những tham số cơ bản bên trong hệ thống giá trị được trả, thông thường biểu diễn bằng đơn vị đô la, như sau :

Kế hoạch chi phí : Khung kế hoạch tiêu thụ cho một dự án qua chương trình đã lập của nó. Cho đa số các dự án phần mềm (và các dự án khác có cường độ nỗ lực cao), khung này nói chung theo dõi khung nhân viên.

Tiến độ thực tế : Kĩ thuật hoàn thành kỹ thuật liên quan tới tiến độ then kế hoạch nằm bên dưới khung tiêu thụ. Trong một dự án tiềm năng, tiến độ thực tế và tiến triển đã lên kế hoạch là gần nhau.

Giá thực tế :Khung tiêu thụ thực tế cho một dự án qua hoạch định thực của nó. Trong một dự án tiềm năng, tiến độ thực tế và tiến triển đã lên kế hoạch là gần nhau.

Giá trị thu được: Giá trị thể hiện giá theo hoạch định của tiến độ thực tế.

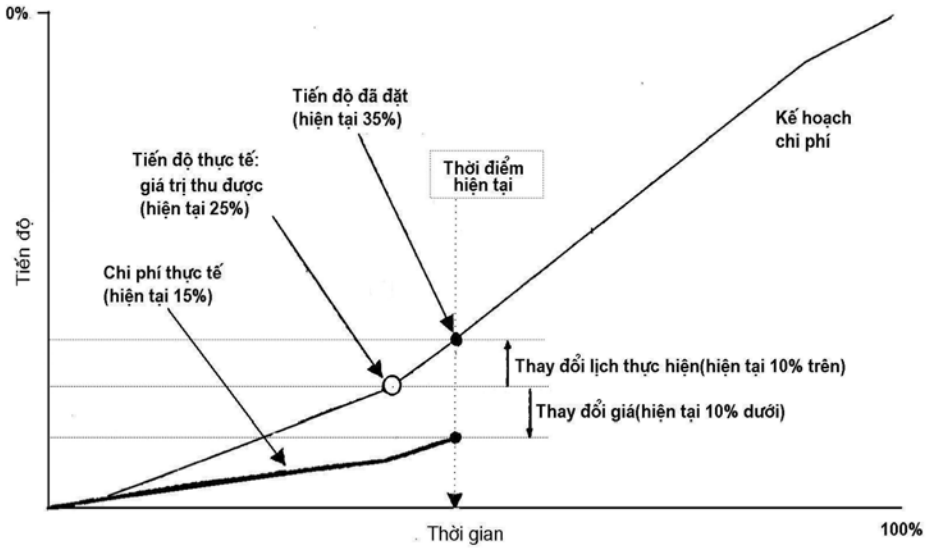
Sự thay đổi giá : sự khác nhau giữa giá thực tế giá trị kiếm được. Những giá trị dương tương ứng với trường hợp vượt quá dự toán; những giá trị âm tương ứng với trường hợp ở trong dự toán.

Sự khác biệt lịch thực hiện : sự khác nhau giữa giá theo kế hoạch và giá trị kiếm được. Những giá trị dương tương ứng với tình trạng trế lịch thực hiện; những giá trị âm tương ứng với tình trạng vượt trước lịch thực hiện.

Hình 13 2 cung cấp một viễn cảnh đồ thị của những tham số này và cho thấy một ví dụ đơn giản của một tình trạng dự án.

Mục đích chính của các metric lõi khác là cung cấp những đội quản lý và kỹ sư với một cách tiếp cận khách quan hơn để đánh giá sự tiến độ thực tế với sự chính xác lớn hơn. Phụ thuộc tất cả các tham số bên trong của một hệ thống giá trị kiếm được, tiến độ thực tế xa vời nhận định chủ quan nhất. Bởi vì đa số các quản lý biết chính xác giá mà họ gánh chịu và bao nhiêu lịch thực hiện họ đã sử dụng, tính biến thiên trong việc tạo ra các nhận định chính xác của tình hình tài chính bởi vậy được đặt trung tâm trong tính đúng đắn của nhận định tiến độ thực tế.

Để hiểu rõ hơn độ mạnh và yếu của một hệ thống giá trị kiếm được, hãy xem xét sự phát triển, cách viết của quyển sách này, nó cũng tương tự trong nhiều cách để phát triển phần mềm. Tiến độ thực tế có thể dễ dàng được theo dõi bởi trạng thái hiện thời của mỗi chương, trọng lượng trung bình bởi số trang được lập kế hoạch cho chương đó.



Hình 13-2. *Các tham số cơ bản của một hệ thống giá trị kiếm được*

Tôi đã theo dõi tình trạng của mỗi phần (một chuỗi các chương có liên quan) sử dụng những trạng thái sau và những giá trị kiếm được (phần trăm hoàn thành đạt được) :

0 tới 50 %: nội dung không đầy đủ

50 %: phác thảo nội dung; tác giả đã hoàn thành văn bản và mỹ thuật phác thảo đầu tiên

65 %: đường gốc văn bản ban đầu; hoàn thành soạn thảo văn bản ban đầu

75 %: đường gốc có thể duyệt lại; hoàn thành soạn thảo văn bản và mỹ thuật

80 %: cập nhật đường gốc; tính chắc chắn giữa các chương được kiểm tra

90 %: xem lại đường gốc; tác giả đã kết hợp những ý kiến của nhà phê bình.

100 %: kết thúc soạn thảo; người biên tập đã hoàn thành một sự kiểm tra cuối cùng.

Những nhận định "phần trăm hoàn thành" được gán có một cách chủ quan dựa trên kinh nghiệm của tôi viết về những tài liệu phức tạp. Tôi lập kế hoạch làm việc cho 10 tháng và tiêu $10,000 cho hỗ trợ công việc. Bảng 13-2 minh hoạ tiến độ của tôi và giá trị kiếm được tại tháng 4 của nỗ lực này. Mặc dầu tôi đã phác thảo khoảng 400 trang trong tổng số 425 trang, tôi đánh giá tiến độ của tôi chỉ ở mức hoàn thành 60%, lấy trọng số trung bình.

Nếu tôi đánh dấu một kế hoạch cho sơ lược chi phí của tôi qua thời gian như trong Hình 13-2, tôi có thể dễ dàng đã đánh giá liệu có phải tôi trên lịch thực hiên và trên dự toán.

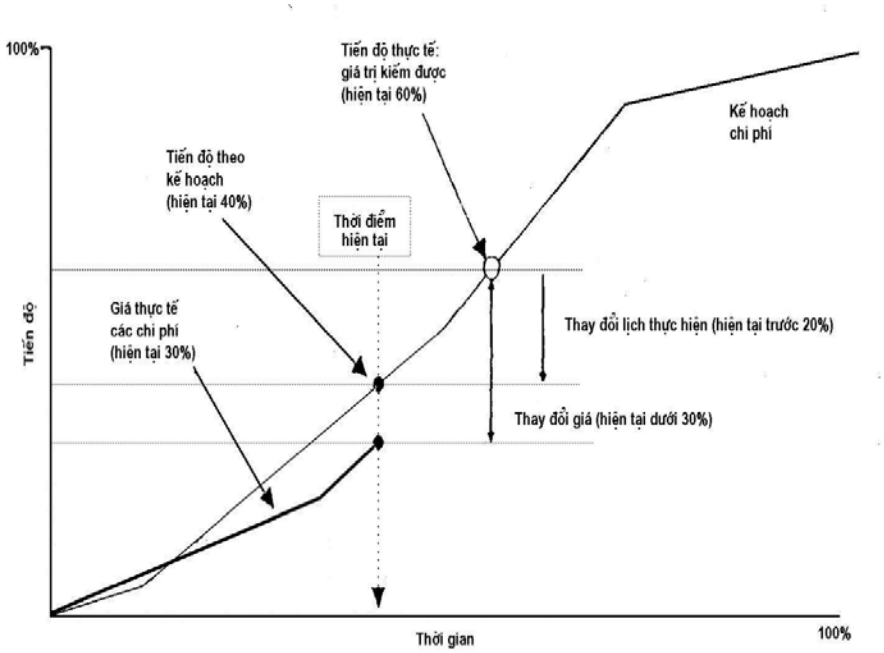
Bảng 13-2. *Đo tiến độ thực tế của sự phát triển sách (ví dụ)*

Thành phần	Số trang	%	Trạng thái
Phần I	60	75%	Xem lại được đường gốc
Phần II	75	75%	Xem lại được đường gốc
Phần III	90	65%	Đường gốc văn bản ban đầu
Phần IV	30	65%	Đường gốc văn bản ban đầu
Phần V	130	50%	Phác thảo nội dung
Các phần khác(lời nói đầu, bảng thuật ngữ, mục lục)	40	25%	Nội dung thiếu

Hình 13-3 minh hoạ kế hoạch và nhận định của tôi vào tháng 4, khi tôi ở 20% phía trước lịch kế hoạch và 30% trong dự toán.

Ví dụ này cung cấp một khung tốt cho việc bàn bạc các thuộc tính then chốt của kế hoạch và nhận định tiến độ thực tế: thiết lập một cơ sở khách quan, phát triển một cấu trúc phân nhỏ công việc thích hợp, và vạch kế hoạc với tính đúng đắn tương xứng.

Tôi thiết lập tiêu chuẩn khách quan cho phần trăm hoàn thành của một thành phần cho trước. Trong khi những tiêu chuẩn này có thể là tối ưu cho tác giả khác, chúng là chính xác cho quyển sách này. Chúng được dựa trên kinh nghiệm của tôi là một tác giả, phong cách phát triển cá nhân của chính tôi, và một mối quan hệ dài lâu với người biên tập kỹ thuật của tôi. Tương tự, cho những dự án phần mềm, trình độ văn hóa của đội ngũ, kinh



Hình 13-3 *Đánh giá tiến độ của sách (ví dụ)*

nghiệm của đội ngũ, và phong cách phát triển(quá trình, khó khăn và trưởng thành) cần phải điều khiển tiêu chuẩn đã đánh giá tiến độ khách quan.

Tôi phát triển một cấu trúc chia nhỏ công việc bằng cách chia công việc của tôi thành những phần (nhóm các chương) ,cái mà là cách tiếp cận dễ nhất để theo dõi tiến độ. Như với phần mềm này, đây là một điều tự nhiên để làm một khi đã có một cấu trúc hợp lý cho quyển sách. Tuy nhiên, tôi đã thay đổi kiến trúc (phác thảo và hướng) vài lần trong một số tháng đầu tiên. Trình bày chi tiết theo dõi bởi sớm thành phần trong chu trình cuộc sống phát triển của quyển sách đưa ra việc làm lại không hấp dẫn mà làm nản lòng tôi từ việc tạo ra những cải tiến kiến trúc. Một cấu trúc phân nhỏ công việc để theo dõi tiến độ của toàn bộ dự án (bao gồm những đóng góp của tác giả, người biên tập, những nghệ sĩ, những nhà phê bình, những nhạc sỹ, và nhà xuất bản) được tổ chức bởi quá trình, chỉ với tiến độ xây dựng theo dõi ngặt bởi thành phần.

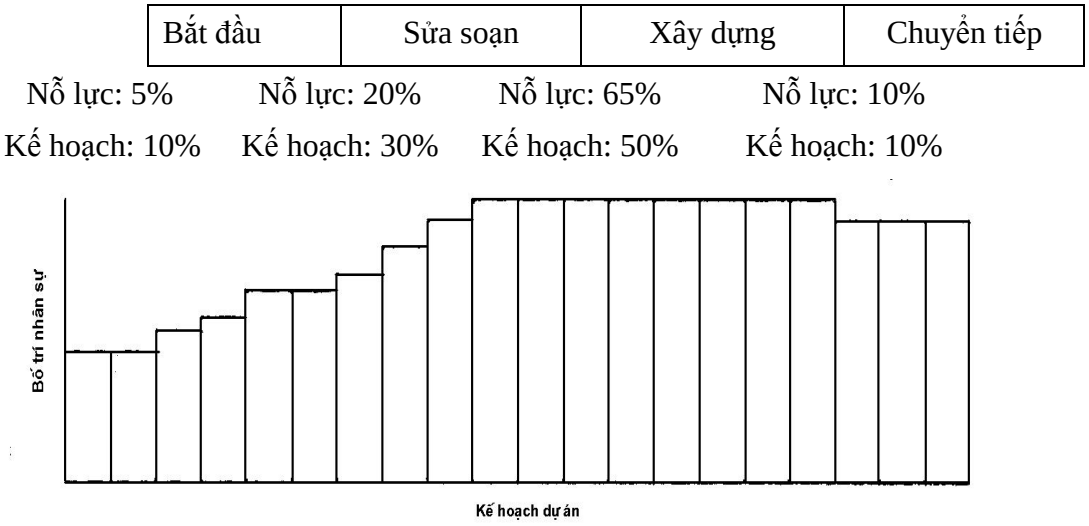
Tôi lập kế hoạch công việc với tính đúng đắn thích hợp cho một dự án người một người. Tôi chọn không trình bày chi tiết theo dõi tiến độ cho đến khi tôi thiết lập một nội dung phác thảo đầy đủ cho một thành phần đã cho. Tôi đã theo dõi sớm tiến độ thông qua một sự định giá chủ quan đơn giản của mức bổ sung của tài liệu phác thảo đầu tiên. Cùng tinh thần đấy nên áp dụng cho những dự án lớn hơn, sử dụng mức của việc lập kế hoạch tin cậy cân xứng với trạng thái hiện thời của dự án và khả năng của việc lập lại kế hoạch.

13.2.3.bố trí nhân viên và nhóm động

Một sự phát triển lặp đi lặp lại nên bắt đầu với một nhóm nhỏ cho đến khi những nguy cơ trong yêu cầu và kiến trúc đã được giải quyết thoả đáng. Phụ thuộc vào sự gối chồng của sự lặp

đi lặp lại và những phạm vi chỉ định khác về dự án, bố trí nhân viên có thể thay đổi. Để riêng biệt, một trong những dạng của các cố gắng phát triển (như xây dựng một hệ thống thông tin tập đoàn), khung bố trí nhân viên trong hình 13-4 là tiêu biểu. Thật hợp lý để chờ đợi nhóm bảo trì nhỏ hơn đội phát triển cho những kiểu phát triển này. Đối với một sự phát triển sản phẩm thương mại, kích thước của nhóm bảo trì và phát triển có thể là như nhau. Khi dùng lâu, liên tục cải thiện sản phẩm, sự bảo trì là chỉ là xây dựng liên tục của những phiên bản mới và tốt hơn.

Theo dõi việc bố trí nhân viên thực tế đối với lại theo kế hoạch là một metric quản lý dễ hiểu và cần thiết. Có một biểu thị quản lý quan trọng khác của những thay đổi trong động lượng dự án: mối quan hệ giữa sự tiêu hao và sự tăng thêm. Gia tăng trong đội ngũ nhân viên có thể làm chậm lại sự tiến độ toàn bộ dự án khi những người mới tiêu thụ thời gian hiệu quả những người đang có trong việc đẩy nhanh tiến độ. Giảm sự tiêu hao những người giảm là một dấu hiệu của thành công. Những kỹ sư được thúc đẩy cao bởi việc tạo ra tiến độ trong việc có cái gì đó để làm; đây là đề tài trở lại luôn, làm nền tảng cho một tiến trình phát triển lặp hiệu quả. Nếu sự thúc đẩy này không ở đó, thì những kỹ sư giỏi sẽ chuyển nơi khác. Một sự gia tăng trong sự tiêu hao không được vạch kế hoạch trước - tức là, một số người rời khỏi một dự án một cách vội vã - là một



Hình 13-4 . *Mô tả sơ lược bố trí nhân sự điển hình*

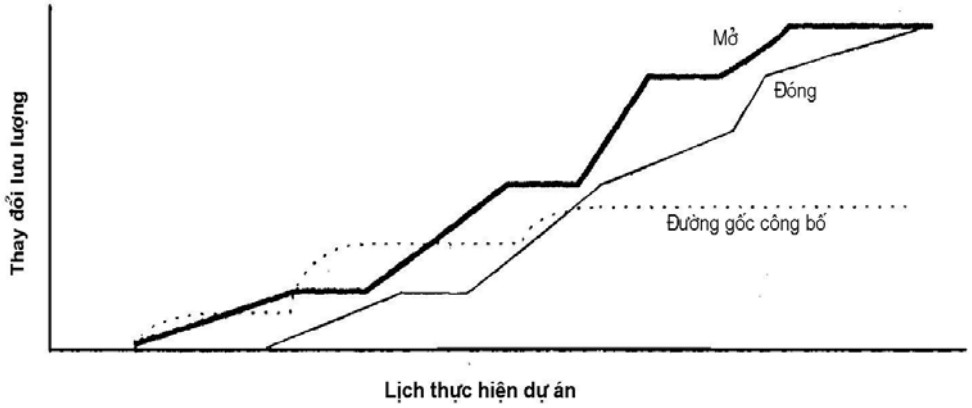
trong phần lớn các biểu thị rõ ràng, rằng một dự án được dự định cho những rắc rối. Nguyên nhân của những hao hụt như vậy có thể đa dạng, nhưng chúng thường mang tính bất mãn cá nhân với các phương thức quản lý, sự thiếu hụt sự nỗ lực chung, hoặc xác suất thất bại gặp phải trong mục đích được vạch.

13.3 Biểu thị chất lượng

Bốn biểu thị chất lượng được dựa chủ yếu trên sự đo đạc của thay đổi phần mềm chéo với các đường gốc phát triển của dữ liệu kỹ thuật (như mô hình thiết kế và mã nguồn). Những metric này được phát triển đầy đủ hơn trong Phụ lục C.

13.3.1 Lưu lượng thay đổi và tính ổn định

Lưu lượng thay đổi toàn cục là một biểu thị chỉ định của tiến độ và chất lượng. Lưu lượng thay đổi được định nghĩa như là số bậc thay đổi phần mềm mở và đóng trên chu trình hoạt động (Hình 13-5). Metric này có thể được thu thập bằng các kiểu thay đổi, bằng phiên bảng, sự bắt chéo tất cả các phiên bản, bởi thành phần, bởi hệ thống con,… Được cặp đôi với metric tiến độ và công việc, nó cung cấp cái nhìn sâu vao trong sự ổn định của phần mềm và tính hội tụ của nó tới sự ổn định (hoặc phân kì đến sự thiếu ổn định). Sự ổn định được định nghĩa như là mối quan hệ giữa các SCO mở và đóng. Lưu lượng thay đổi liên quan đến lịch công bố phiên bản cung cấp một cái nhìn sâu vào bên trong lịch thực hiện mang tính có thể đoán trước được. Ba metric về chất lượng kế tiếp tập trung hơn về chất lượng của sản phẩm.



Hình 13-5. *Mong đợi tính ổn định qua vòng đời của một dự án vững mạnh.*

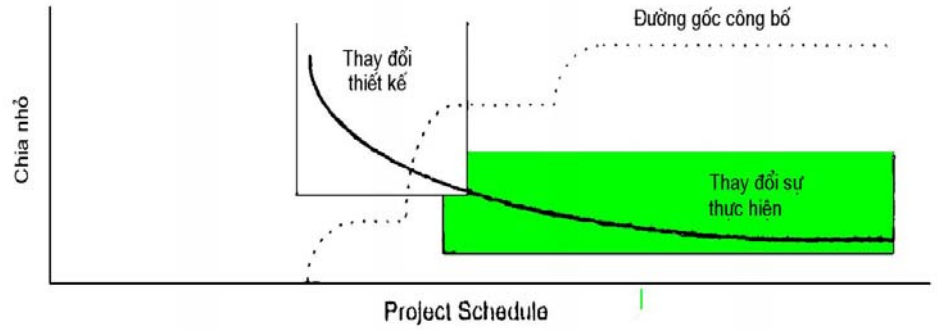
13.3.2.Chia nhỏ và tính modul hoá

Chia nhỏ được định nghĩa như là phạm vi trung bình của sự thay đổi, cái mà là một lượng của đường gốc phần mềm cần làm lại (trong SLOC, điểm chức năng, các thành phần, hệ thống con, các file,…). *Tính modul hoá* là khuynh hướng đứt đoạn trung bình qua thời gian. Đối với một dự án vững mạnh, khuynh hướng mong đợi là giảm hoặc ổn định (Hình 13-6).

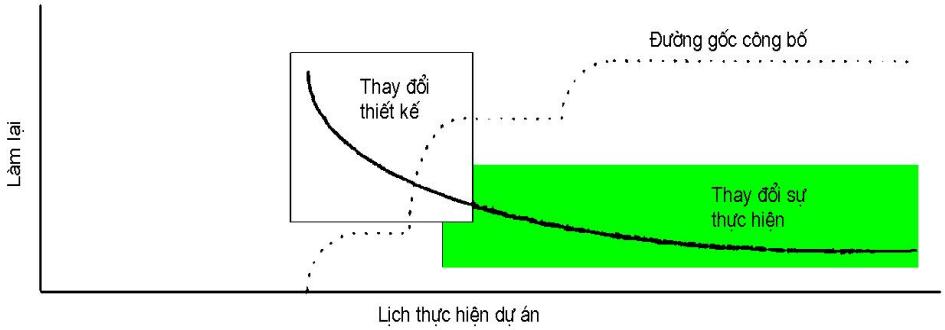
Biểu thị này cung cấp cái nhìn sâu bên trong đặc tính tốt hay xấu của thay đổi phần mềm. Trong một tiến trình phát triển lặp kế thừa thành thục, các thay đổi sớm được mong đợi kết quả nhiều vụn vặt hơn so với những thay đổi trễ hơn. Các khuynh hướng đứt đoạn tăng theo thời gain biểu thị rằng khả năng duy trì sản phẩm là đáng ngờ.

13.3.3.Làm lại và tính tương thích

Làm lại được định nghĩa như là giá trung bình của thay đổi, cái là nỗ lực để phân tích, giải quyết, và kiểm nghiệm lại tất cả các thay đổi với các đường gốc phần mềm. *Tính tương thích* được định nghĩa như là khuynh hướng làm lại qua thời gian. Đối với một dự án vững mạnh, khuynh hướng mong đợi là giảm hoặc ổn định (Hình 13-7).



Hình 13-6 Mong đợi tính modul hoá qua vòng đời của một dự án vững mạnh



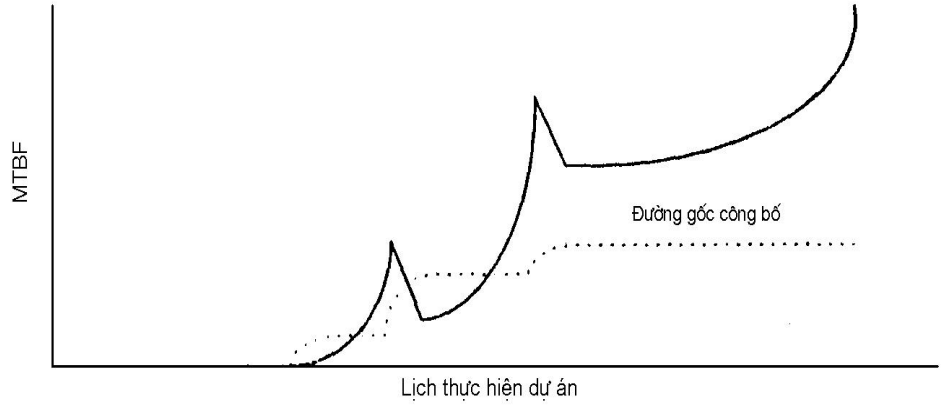
Hình 13-7. *Mong đợi tính ổn định qua vòng đời của một dự án vững mạnh*

Không phải tất cả các thay đổi được tạo lập ngang bằng nhau. Một số thay đổi có thể được tạo ra trong một giờ làm việc, trong khi số khác chiếm nhiều tuần làm việc. Metric này cung cấp cái nhìn sâu bên trong thước đo sự làm lại. Trong một tiến trình phát triển lặp thành thục, các thay đổi sớm (các thay đổi cấu trúc ảnh hưởng đa thành phần và nhiều người) được mong đợi đòi hỏi thêm sự làm lại hơn những thay đổi về sau (thay đổi sự thi hành qui về một thành phần hoặc một người). Khuynh hướng làm lại là tăng với thời gian rõ ràng biểu thị rằng khả năng duy trì cảu sản phẩm là đáng ngờ.

13.3.4 MTBF và tính thành thục

MTBF là thời gian thông thường trung bình giữa các lỗi phần mềm. Theo các thuật ngữ thô, MTBF được tính bằng các chia thời gian thử nghiệm bởi số các SCO kiểu 0 và kiểu 1. *Tính thành thục* được định nghĩa như là khuynh hướng MTBF qua thời gian (Hình 13-8).

Cái nhìn sớm vào bên trong tính thành thục đòi hỏi rằng một cơ sở hạ tầng thử nghiệm hiệu quả được thiết lập. Các tiếp cận thử nghiệm theo thông lệ cho các chương trình phần mềm tập trung vào thử nghiệm hoàn thành thu được bao trùm lên mọi dòng mã, mọi nhánh và v.v.



Hình 13-8. *Mong đợi sự thành thục qua một vòng đời dự án*

Trong những hệ thống phần mềm thành phần hoá và phân tán ngày nay, bao quát những thử nghiệm hoàn thiện như vậy là có thể đạt được chỉ với các thành phần riêng biệt. Các hệ thống của các thành phần là được kiểm tra một cách hiệu quả hơn bằng cách dùng các kĩ thuật thống kê. Hệ quả là, các metric đo tính thành thục do các thống kê trên thời gian sử dụng hơn là bao quát sản phẩm.

Các lỗi phần mềm có thể phân loại thành hai kiểu: xác định và không xác định. Các nhà vật lý sẽ đặc tính hoá các kiểu này thành Bohr-bugs và Heisen-bugs một cách tương ứng. Bohr-bugs miêu tả một lớp các lỗi thường xuyên xảy ra khi phần mềm được khuyến khích theo một cách nào đó. Những lỗi này bị gây ra chủ yếu do các lỗi mã hoá, và các thay đổi được cách ly một cách điển hình đối với một thành phần độc lập. Heisen-bugs là các sai sót phần mềm trùng với một sự xuất hiện có khả năng xảy ra nào đó của một tình trạng cho trước. Những lỗi này phần lớn luôn là các lỗi thiết kế (các thay đổi đòi hỏi thường xuyên đa thành phần) và một cách điển hình là không lặp lại thậm chí khi phần mềm được khuyến khích thep một cách hiển nhiên giống thế. Để cung cấp thử nghiệm tương ứng bao quát và giải quyết mô hình Heisen-bugs có ý nghĩa một cách thống kê, việc thử nghiệm thống kê rộng rãi dưới hiện thực và các kịch bản sử dụng ngẫu nhiên hoá là cần thiết.

Các chương trình phần mềm theo thông lệ thi hành một đơn chương trình trên một một bộ xử lý đơn giản thông thường chứa chỉ mô hình Bohr-bugs. Hiện tại, các hệ thống phân tán với số lớn các thành phần có hoạt động liên quan thi hành chéo trên một mạng các bộ xử lý là chỗ yếu đối với mô hình Heisen-bugs, cái mà rất phức tạp để tìm ra, phân tích và giải quyết. Cách tốt nhất để hoàn thiện một sản phẩm phần mềm là thiết lập một cơ sở hạ tầng thử nghiệm ban đầu cho phép thực thi các kịch bản sử dụng ngẫu nhiên hoá sớm trong vòng đời và phát

triển liên tục chiều sâu và chiều rộng của các kịch bản sử dụng để tối ưu hoá bao quát sự đan xen các thành phần có độ tin cậy chuẩn.

Như các đường gốc được thiết lập, chúng nên được đưa ra liên tục để kiểm tra các kịch bản. Từ gốc của việc thử nghiệm, các metric đo tính tin cậy có thể được dỡ ra. Cái nhìn có ý nghĩa vào bên trong sự thành thục sản phẩm có thể thu được bằng cực đại hoá thời gian thử nghiệm (qua môi trường thử nghiệm độc lập, các thử nghiệm hồi quy tự động, việc thử nghiệm thống kê ngẫu nhiên, việc thử nghiệm sau nhiều giờ tập trung cao,...). Thử nghiệm tiếp cận cung cấp một cơ cấu có tác động mạnh cho việc đẩy mạnh tự động trong các hoạt động thử nghiệm sớm trong vòng đời cũng như trong thực hành. Kĩ thuật này có thể được sử dụng để kiểm soát các cải tiến hiệu quả và thước đo tin cậy.

13.4.Các dự tính vòng đời

Không có sự bắt nguồn mang tính toán học hoặc hình thức để sử dụng bảy metric cốt lõi. Tuy nhiên, có các lý do chỉ định để lựa chọn chúng:

Các biểu thị chất lượng được bắt nguồn từ sự tiến triển sản phẩm hơn là từ các giả tưởng.

Chúng cung cấp cái nhìn bên trong các vật thải được tạo bởi tiến trình. Các metric vụn vặt và làm lại là một chuẩn đo của phần lớn các tiến trình chế tạo.

Chúng thừa nhận tính kế thừa động tự nhiên của một tiến trình phát triển lặp. Hơn là chú trọng vào giá trị, chúng tập trung ngặt vào các khuynh hướng hoặc các thay đổi đối với thời gian.

Tổ hợp của cái nhìn từ giá trị hiện tại và khuynh hướng hiện tại cung cấp các biểu thị hữu hình cho các hoạt động quản lý.

Các giá trị thực tế của những metric này có thể thay đổi rộng rãi qua các dự án, các tổ chức, và các phạm vi. Các khuynh hướng liên quan ngang qua các giai đoạn dự án, tuy nhiên, nên theo các mẫu chung được trình bày trong bảng 13-3. Một tổ chức phát triển trưởng thành nên có khả năng miêu tả các đích có nhiều tính dứt khoát và chính xác cho đường của nó thuộc kinh doanh và các tiến trình chỉ định.

Bảng 13-3 Mẫu mặc định của sự tiến hoá các metric vòng đời

Metric	Khởi đầu	Sửa soạn	Xây dựng	Chuyển tiếp
Tiến độ	5%	25%	90%	100%
Cấu trúc	30%	90%	100%	100%
ứng dụng	<5%	20%	85%	100%
Chi phí	Thấp	Trung bình	Cao	Cao

Cố gắng	5%	25%	90%	100%
Thời hạn	10%	40%	90%	100%
Bố trí nhân viên	Nhóm nhỏ	Tăng lên	Không đổi	Thay đổi
Độ ổn định	Không ổn định	Vừa phải	Vừa phải	ổn định
Kiến trúc	Không ổn định	Vừa phải		ổn định
ứng dụng	Không ổn định	Không ổn định	Vừa phải	ổn định
Tính modul hoá	50%-100%	25%-50%	<25%	5%-10%
Kiến trúc	>50%	>50%	<15%	<5%
ứng dụng	>80%	>80%	<25%	<10%
Độ tương thích	Thay đổi	Thay đổi	Tốt	Tốt
Kiến trúc	Thay đổi	Vừa phải	Tốt	Tốt
ứng dụng	Thay đổi	Thay đổi	Vừa phải	Tốt
Độ thành thục	Mẫu đầu tiên	Phân mảnh	Dùng được	Mạnh
Kiến trúc	Mẫu đầu tiên	Dùng được	Mạnh	Mạnh
ứng dụng	Mẫu đầu tiên	Phân mảnh	Mạnh	Mạnh

13.5 Các metric phần mềm thực dụng

Đo đạc là có ích, nhưng nó không tạo ra bất kỳ ý tưởng cho người ra quyết định. Nó chỉ cung cấp dữ liệu để giúp họ yêu cầu quyết định đúng, hiểu ngữ cảnh, và tạo ra các quyết định khách quan. Bởi vì tính tự nhiên động cao của các dự án phần mềm, các thước đo này phải có mặt tại bất cứ thời điểm nào. tính đo được tới các tập con khác nhau của tiến triển sản phẩm (phiên bản, thành phần, lớp), và được quản lý sao cho các khuynh hướng có thể được nhận định (sự dẫn xuất đầu tiên và thứ hai đối với thời gian). Tình trạng này đã đạt được trong thực

hành chỉ trong các dự án nơi mà các metric được bảo quản trực tuyến như một sản phẩm phụ tự động của môi trường thiết kế/tích hợp.

Các đặc tính cơ bản của một metric tốt là như sau:

Nó được coi là có ý nghĩa bởi khách hàng, người quản lý và người trình diễn. Nếu bất kỳ một trong số người này không xem metric là có ý nghĩa, nó sẽ không được dùng. “Khách hàng luôn luôn đúng” là phương châm của người bán, không phải là một nguyên lý kỹ thuật. Các khách hàng đi tới nhà cung cấp công trình phần mềm bởi vì các nhà cung cấp có nhiều kinh nghiệm hơn họ trong việc phát triển và quản lý phần mềm. Các khách hàng luôn chấp nhận các metric được chỉ ra là có ý nghĩa đối với người phát triển.

Nó cho thấy rõ tương quan có thể qui số lượng được giữa các lo lắn tiến trình và hiệu quả kinh doanh. Chỉ những mục đích có tổ chức và mục tiêu thực là thuộc về tài chính: giảm giá, tăng lợi nhuận và tăng số dư.

Nó là mục tiêu và được xác định không mơ hồ. Tính mục tiêu nên dịch snag một số dạng của các biểu diễn số (như là các số, tỷ lệ phần trăm, tỉ số) như là đối lại với các thể hiện nguyên văn (như là tuyệt vời, tốt, vừa phải, nghèo nàn). Tính mơ hồ được tối thiểu hoá qua các đơn vị dễ hiểu của phép đo (như tháng làm việc, SLOC, thay đổi, điểm chức năng, lớp, kịch bản, nhu cầu), cái mà khó có thể định nghĩa chính xác trong thế giới công nghệ phần mềm.

Nó biểu lộ khuynh hướng. Đây là một đặc tính quan trọng. Việc hiểu các thay đổi trong một giá trị metric với khía cạnh thời gian, các dự án sau này, các phiên bản sau này, là một viễn cảnh vô cùng quan trọng, đặc biệt là cho các mô hình phát triển lặp ngày nay. ít có một metric đã cho điều khiển các hoạt động thích hợp một cách trực tiếp. Thông thường hơn, một metric biểu thị một viễn cảnh. Nó đưa tới quyết định có căn cứ (nhà quản lý, nhóm, hoặc một thực thể xử lý thông tin khác) để thông dịch metric và quyết định hành động nào là cần thiết.

Nó là một sản phẩm phụ tự nhiên của tiến trình. Metric không đưa ra các giả tưởng mới hoặc các hoạt động cao trên mức; nó được bắt nguồn trực tiếp từ công nghệ chính và quản lý dòng chảy công việc.

Nó được hỗ trợ bởi quá trình tự động hoá. Kinh nghiệm đã chỉ ra rằng phần lớn các metric thành công là những metric được thu thập và báo cáo bởi các công cụ tự động theo từng phần bởi vì các công cụ phần mềm đòi hỏi các định nghĩa chặt chẽ của dữ liệu và chúng xử lý.

Khi các metric bộc lộ một vấn đề, cái quan trọng là đạt được các triệu chứng và chuẩn đoán chúng. Metric thông thường bộc lộ các hiệu ứng, nguyên nhân đòi hỏi sự tổng hợp đa viễn cảnh và lí do. Lấy ví dụ, lý do vẫn đòi hỏi phải thông dịch các hoàn cảnh sau một cách đúng đắn:

Một số ít các yêu cầu thay đổi cho một đường gốc phần mềm có lẽ có ý nghĩa rằng phần mềm đã thành thục và không lỗi, hoặc nó có ý nghĩa rằng nhóm thử nghiệm đang đi nghỉ.

Một thứ tự thay đổi phần mềm được mở trong một thời gian dài có nghĩa rằng vấn đề đơn giản để chẩn đoán và giải pháp đòi hỏi sự làm lại đáng kể, hay nó có nghĩa rằng một vấn đề rất tốn thời gian đề chuẩn đoán và giải pháp đòi hỏi một thay đổi đơn giản tới một dòng mã.

Một sự gia tăng nhân lực lớn trong một tháng nào đó cao thể gây ra tiến độ tăng tỉ lệ nếu họ được đào tạo những người là sản phẩm từ ban đầu. Nó có thể gây ra tiến độ chậm lại nếu họ không được đào tạo mức thuê mới người yêu cầu sự hỗ trợ rộng rãi từ những người sản xuất để tăng tốc độ.

Đánh giá giá trị không thể tạo bằng các metric; chúng phải để cho những thực thể thông minh hơn như các nhà quản lý dự án phần mềm.

13.6 Metric tự động hoá

Có nhiều cơ hội để tự động háo các hoạt động kiểm soát dự án của một dự án phần mềm. Để quản lý đối với một kế hoạch, một khung kiểm soát dự án phần mềm được quản lý một phiên bản trực tuyến của trạng thái của sự tiến triển giả tưởng cung cấp một lợi thế then chốt. Khái niệm này được giới thiệu bởi Airlie Software Council [Brown,1996], sử dụng phép ẩn dụ một dự án “dashboard”. ý tưởng là cung cấp một khung hiển thị tích hợp dữ liệu từ nhiều nguồn để chỉ ra trạng thái hiện tại của một số khía cạnh của phần mềm. Lấy ví dụ, người quản lý dự án phần mềm sẽ muốn một sự biểu lộ với các giá trị dự án tổng thể, người quản lý việc thử nghiệm muốn xem một sự biểu lộ tập trung trên các metric chỉ định tới phiên bản beta sắp tới, và người quản lý phát triển có thể chỉ thích các dữ liệu về các hệ thống con và các thành phần mà họ chịu trách nhiệm. Panel có thể hỗ trợ các đặc trưng chuẩn như đèn báo cảnh báo, điểm bắt đầu, tỉ lệ biến đổi, các dạng thức số, các dạng thức tương tự để biểu thị một sự khái quát hoàn cảnh hiện tại. Nó có thể cung cấp các khả năng mở rộng để phân tích hoàn cảnh chi tiết. Sự hỗ trợ tự động hoá có thể cải tiến sự quản lý, nhìn vào bên trong tiến độ và các khuynh hướng chất lượng và cả, thiện sự chấp nhận các metric bởi nhóm kỹ thuật.

Để thực hiện một SPCP đầy đủ, cần thiết để định nghĩa và phát triển điều sau:

Metric nguyên thuỷ: các biểu thị, các khuynh hướng, sự so sánh và sự tiến triển

Một giao diện người dùng đồ hoạ: Hỗ trợ GUI cho vai trò người quản lý dự án phần mềm và sự uyển chuyển để hỗ trợ các vai trò khác.

Metric thu thập đại diện: dữ liệu được trích từ các công cụ môi trường để quản lý các kí hiệu kĩ thuật cho các tập hợp giả tưởng đa dạng

Metric phục vụ quản lý dữ liệu: hỗ trợ quản lý dữ liệu cho metric hiển thị GUI và lưu trữ dự liệu được lấy ra bởi các đại diện.

Metric định nghĩa: metric thực tế biểu diễn cho các tiến độ yêu cầu (được trích từ tập các mẫu yêu cầu), tiến độ thiết kế (được trích từ tập các mẫu thiết kế), tiến độ thực hiện (được trích từ tập các mẫu thiết kế), tiến độ đánh giá (được trích từ tập các mẫu triển khai) và các chiều

tiến độ khác (được trích từ các nguồn thủ công, các hệ thống quản lý tài chính, các mẫu quản lý,…).

Người làm: điển hình là giám sát viên và quản trị viên.

Các giám sát viên chỉ định (gọi là vai trò) bao gồm các quản lý dự án phần mềm, trưởng nhóm phát triển phần mềm, các kiến trúc sư phần mềm và khách hàng. Đối với mọi vai trò, có một cấu hình panel chỉ định và phạm vi của dữ liệu được biểu diễn. Từng vai trò thực thi các trường hợp chung như nhau nhưng với trọng điểm khác nhau.

Giám sát viên: xác định cách bài trí panel từ các cơ cấu có sẵn, các đối tượng đồ hoạ và sự liên kết tới dữ liệu dự án; tra vấn dữ liệu được hiển thị tại các mức khác nhau của sự khái quát

Quản trị viên: cài đặt hệ thống, xác định các cơ cấu mới, các đối tượng đồ hoạ và các liên kết; vận dụng các chức năng lưu trữ; xác định các cấu trúc hợp thành và phân tích để hiển thị nhiều mức của sự khái quát.

Hiển thị toàn bộ được gọi là panel. Bên trong panel là các đối tượng đồ hoạ là các dạng bài trí (như các mặt đồng hồ và các biểu đồ) thông tin. Mỗi đối tượng đồ hoạ hiển thị một metric. Một panel điển hình bao gồm một số đối tượng đồ hoạ được đặt trong một cach bài trí hình học đặt biệt. Một metric chỉ ra trong một đối tượng đồ hoạ được gán nhãn với một kiểu metric, mức tóm lược và tên bản sao (như các dòng mã, hệ thống con, server1). Các metric có thể hiển thị theo hai chế độ: giá trị, tham chiếu tới một điểm cho trước trong thời gian, hoặc đồ thị, tham chiếu tới nhiều điểm phối hợp trong thời gian. Chỉ một số các dạng hiển thị là có thể áp dụng cho các metric đồ hoạ.

Các metric có thể được hiển thị với hoặc vắng các giá trị điều khiển. Một giá trị điều khiển là một sự dự tính hiện có, hoặc là tuyệt đối hoặc là tương đối, để dùng cho phép so sánh với metric thay đổi động. Lấy ví dụ, kế hoạch cho một metric tiến độ cho trước là một giá trị điều khiển để so sánh thực tế của metric đó. Các ngưỡng là một ví dụ khác của các giá trị điều khiển. Ngang qua một ngưỡng có thể đưa đến một trạng thái thay đổi cần xem là hiển nhiên đối với người dùng. Các giá trị điều khiển có thể được chỉ ra trong cùng đối tượng đồ hoạ như metric tương ứng, cho sự so sánh hình ảnh đối với người dùng.

Các biểu thị có thể hiển thị dữ liệu dữ liệu trong các dạng nhị phân (như đen và trắng), tam phân (như đỏ, vàng, xanh), số (số nguyên hoặc số thực), hoặc một số các kiểu liệt kê khác (một dãy có thể các giá trị rời rạc như thứ hai...thứ bảy, sẵn sàng-ngắm-bắn, tháng giêng..tháng mười hai). Các biểu thị cũng cung cấp một cơ cấu có thể được dùng để tóm lược một điều kiện hoặc phạm vi kết hợp với metric khác, hoặc các quan hệ giữa các metric và các giá trị điều khiển kết hợp với chúng.

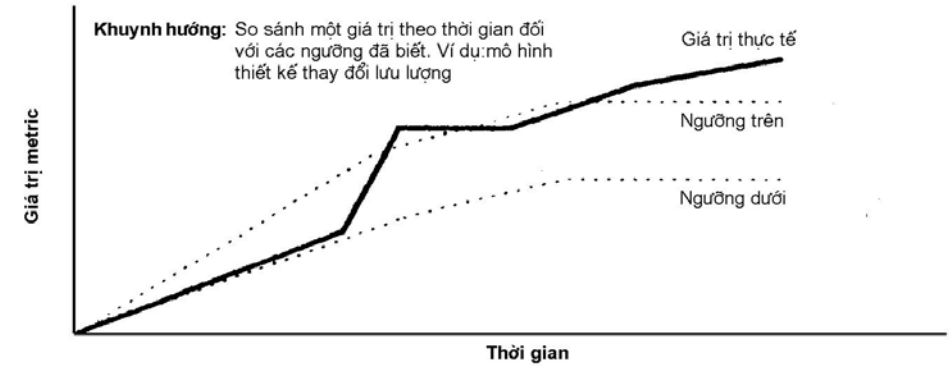
Một khuynh hướng đồ thị biểu diễn giá trị qua thời gian và cho phép các ngưỡng trên và dưới được xác định. Ngang qua một ngưỡng có thể liên kết với một biểu thị kết hợp để mô tả một sự thay đổi trạng thái có thể thấy rõ được từ xanh đến đỏ hay ngược lại. Các khuynh hướng hỗ trợ người dùng lựa chọn mức tăng thời gian (như ngày, tuần, tháng, quý, năm). Một

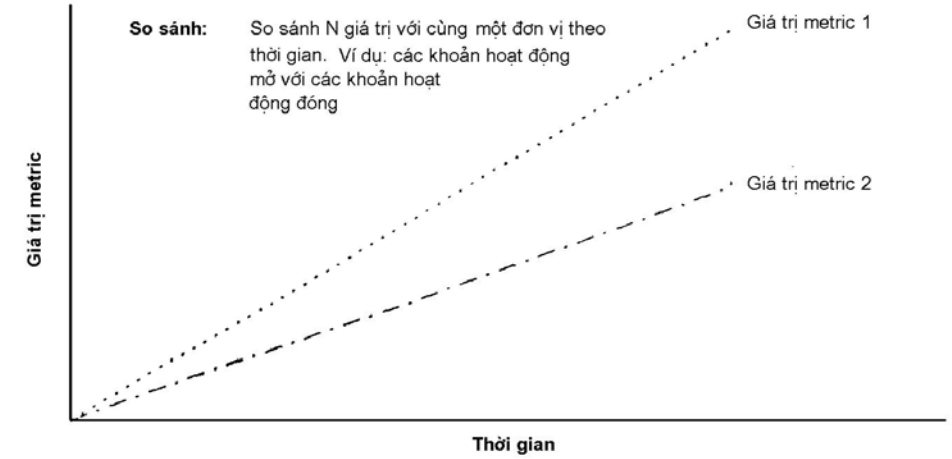
đồ thị so sánh biểu diễn đa giá trị qua thời gian. Hội tụ hay phân kì trong số các giá trị có thể được liên kết tới một biểu thị. Một đồ thị tiến độ thể hiện phần trăm hoàn thành, ở đây các phần tử của tiến độ được chỉ ra như là sự dịch chuyển giữa các trạng thái và giá trị kiếm được kết hợp với từng trạng thái. Các khuynh hướng, sự so sánh, các tiến độ được minh hoạ trong Hình 13-9.

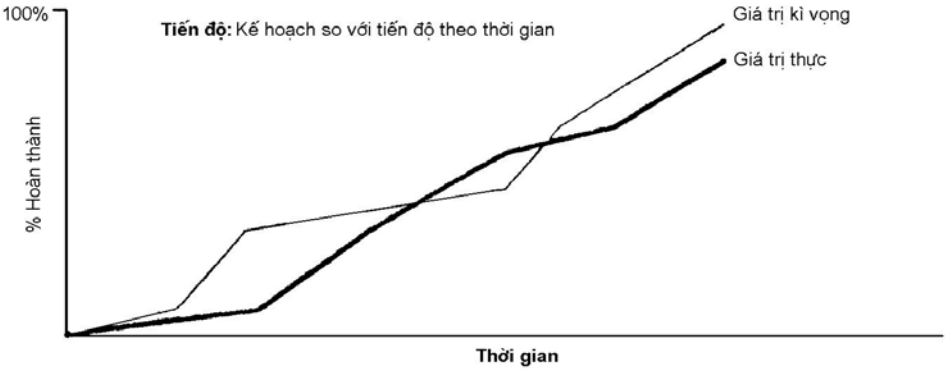
Metric thông tin có thể được tóm lược theo sau một định nghĩa mang tính người dùng, cấu trúc tuyến tính. (Lấy ví dụ, các dòng mã có thể được tóm lược bởi đơn vị, hệ thống con và dự án). Dự án là hạn định mức trên cùng đối với các dữ liệu thuộc một tập hợp (ngữ cảnh mức trên cùng). Người dùng có thể định nghĩa các cấu trúc tóm tắt cho các mức thấp hơn, lựa chọn mức hiển thị dựa trên các cấu trúc được định nghĩa trước đây, và khoan xuống một số được tóm tắt hoá bằng các xem xét các mức chi tiết thấp hơn.

Hình 13-10 minh hoạ một ví dụ đơn giản của một SPCP cho một dự án. Trong trường hợp này, vai trò người quản lý phần mềm xác định một hiển thị mức trên cùng với bốn đối tượng đồ hoạ.

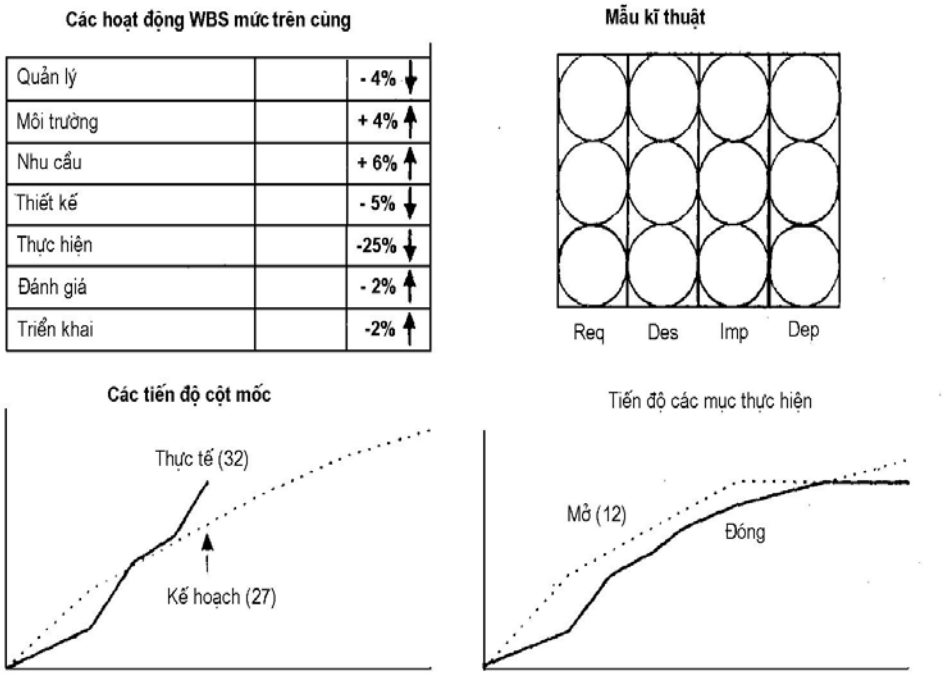
Trạng thái hoạt động dự án. Đối tượng đồ hoạ ở phía trái trên cung cấp một sự khái quát của trạng thái của các phần tử WBS mức trên cùng. Bảy phần tử có thể được mã hoá đỏ, vàng và xanh để phản chiếu trạng thái giá trị kiếm được hiện tại. (Trong hình 13-10, chúng được mã với màu trắng với sự chuyển màu xám). Lấy ví dụ, màu xanh sẽ thể hiện *phía trước dự án*, màu vàng biểu thị *trong 10% kế hoạch*, và màu đỏ sẽ nhận biết các phần tử lớn hơn 10% giá hoặc sự thay đổi lịch thực hiện. Đối tượng đồ hoạ này cung cấp vài ví dụ của các chỉ báo: ba mầu, phần trăm thực tế và các suy dẫn đầu tiên hiện thời (mũi tên hướng lên có nghĩa là khá hơn, mũi tên hướng xuống có nghĩa là xấu đi).







H×nh 13-9. *C¸c vÝ dô cña c¸c líp metric nÒn t¶ng*



Trạng thái mẫu kĩ thuật. Đối tượng đồ hoạ ở phía phải trên cung cấp một sự khái quát của trạng thái của các mẫu kĩ thuất tiến triển. Đèn Req sẽ hiển thị một nhận định của trạng thái hiện thời của các mô hình sử dụng và các chỉ định yêu cầu. Đèn Des sẽ như thế đối với các mô hình thiết kế, đèn Imp cho đường gốc mã nguồn và đèn Dep cho chương trình thử nghiệm.

Các tiến độ cột mốc. Đối tượng đồ hoạ trên góc trái dưới cung cấp một nhận định tiến độ của sự đạt được các cột mốc đối với kế hoạch và cung cấp các chỉ báo của các giá trị hiện thời.

Tiến độ mục hành động. Các đối tượng đồ hoạ ở góc dưới phải cung cấp một viễn cảnh khác của tiến độ, chỉ ra số các vấn đề mở và đóng hiện thời.

Hình 13-10 là một ví dụ của một thể hiện metric tiến độ. Mặc dù ví dụ là tầm thường, nó cung cấp một cái nhìn vào trong khả năng cơ bản của một hiển thị SPCP.

Khuôn dạng và nội dung của bất cứ panel dự án là có thể chỉnh được với các lựa chọn của người quản lý dự án phần mềm để theo dõi các metric của sở thích mức trên cùng. Một số người quản lý sẽ muốn chỉ dữ liệu tóm tắt và một số ít khuynh hướng then chốt trong hiển thị mức cao nhất của họ. Một số khác se muốn nhiều khuynh hướng và chỉ định chi tiết. Một SPCP nên hỗ trợ phép đo và cung cấp khả năng để đào sâu vào chi tiết của bấy kì metric cho trước nào. Lấy ví dụ, truy vấn một đèn đỏ cho các mẫu triển khai sẽ sinh ra mức chi tiết tiếp theo trong thời gian (một biểu đồ khuynh hướng) hoặc trong mô tả (trạng thái thử nghiệm chi tiết cho từng phiên bản, từng hệ thống con,…).

Trường hợp sử dụng mức trên cùng sau, miêu tả khái niệm hoạt động cơ bản cho một SPCP, tương ứng với một giám sát viên tương tác với một panel điều khiển:

Bắt đầu SPCP. SPCP bắt đầu và chỉ ra phần lớn thông tin hiện thời được lưu khi người dùng lần cuối cùng sử dụng SPCP.

Lựa chọn các thông số panel. Người dùng lựa chọn từ một danh sách các thông số panel mặc định đã được định nghĩa. SPCP hiển thị các thông số được lựa chọn.

Lựa chọn giá trị hoặc metric đồ hoạ. Người dùng lựa chọn xem metric nên được hiển thị cho một điểm cho trước trong thời gian hay trong một đồ thị như một khuynh hướng. Các giá trị mặc định là phần lớn các giá trị đo gần đây có sẵn. Mặc định cho các khuynh hướng là hàng tháng.

Lựa chọn thêm các điều khiển. Người dùng trỏ tới một đối tượng đồ hoạ và yêu cầu các giá trị điều khiển đó cho metric đó và trỏ tới thời điểm sẽ được hiển thị. Trong trường hợp các khuynh hướng, các điều khiển được chỉ ra phụ với metric.

Khoan xuống khuynh hướng. Người dùng trỏ tới một đối tượng đồ hoạ hiển thị một khuynh hướng và khoan xuống để quan sát khuynh hướng đối với metric.

Khoan xuống để trỏ tới thời điểm. Người dùng trỏ tới một đối tượng đồ hoạ hiển thị một khuynh hướng và khoan xuống để quan sát giá trị đối với metric.

Khoan xuống các mức thấp hơn của thông tin. Người dùng trỏ tới một đối tượng đồ hoạ hiển thị một thời điểm và khoan xuống để quan sát mức kế tiếp của thông tin.

Khoan xuống mức thấp hơn của các chỉ báo. Người dùng trỏ tới một đối tượng đồ hoạ hiển thị một chỉ báo và khoan xuống để quan sát sự chia nhỏ của mức kế tiếp của các chỉ báo.

SPCP là một ví dụ của một metric tiếp cận tự động hoá thu thập, tổ chức, và báo cáo các giá trị và xu thế được trích trực tiếp từ các mẫu kĩ thuật tiến triển. Các kĩ sư phần mềm sẽ chấp nhận các metric nếu và chỉ nếu các metric được tự động hoá bởi môi trường.

Chương 14

Sự biến đổi tiến trình - tailoring the process

Các điểm quan trọng:

- *Khung công việc của một tiến trình phải được định hình để xác định các đặc điểm của dự án .*
- Qui mô của một dự án - trong trường hợp đặc biết là kích thước nhóm thường dành cho khung công việc của tiến trình hơn là cho bất cứ yếu tố nào.
- Các yếu tố quan trọng khác bao gồm mối quan hệ giữa các cổ đông , tính mềm dẻo và chuyên nghiệp của tiến trình, sự rủi ro của kiến trúc và kinh nghiệm trong lĩnh vực.
- Trong khi việc xác định các thao tác sẽ rất đa dạng thì tinh thần của các tiến trình là như nhau.

Hệ thống quản lí phần mềm nỗ lực mở rộng phạm vi của các lĩnh vực. Trong khi có một số đề tài và kĩ thuật chung nó luôn cần thiết biến đổi tiến trình để xác định sự cần thiết của các dự án sắp tới .Một nhà phát triển công cụ phần mềm thương mại cùng với sự điều khiển đầy đủ sự đầu tư của nó sẽ được sử dụng trong rất nhiều tiến trình khác nhau mà những nhà phân tích phần mềm dựa vào hợp đồng để tự động hoá an toàn hệ thống năng lượng hạt nhân. Không có vấn dề gì với các tiến trình chuyên nghiệp ,và hiệu quả của việc quản lí phần mềm có giá trị đối với một hệ thống phân tích lớn hơn là những gì chúng đã làm cho một hệ thống nhỏ. Tuy nhiên, liên quan tới các mục tiêu kinh tế, việc đầu tư để quản lí phần mềm tốt hơn sẽ có giá trị với bất cứ tổ chức phần mềm nào

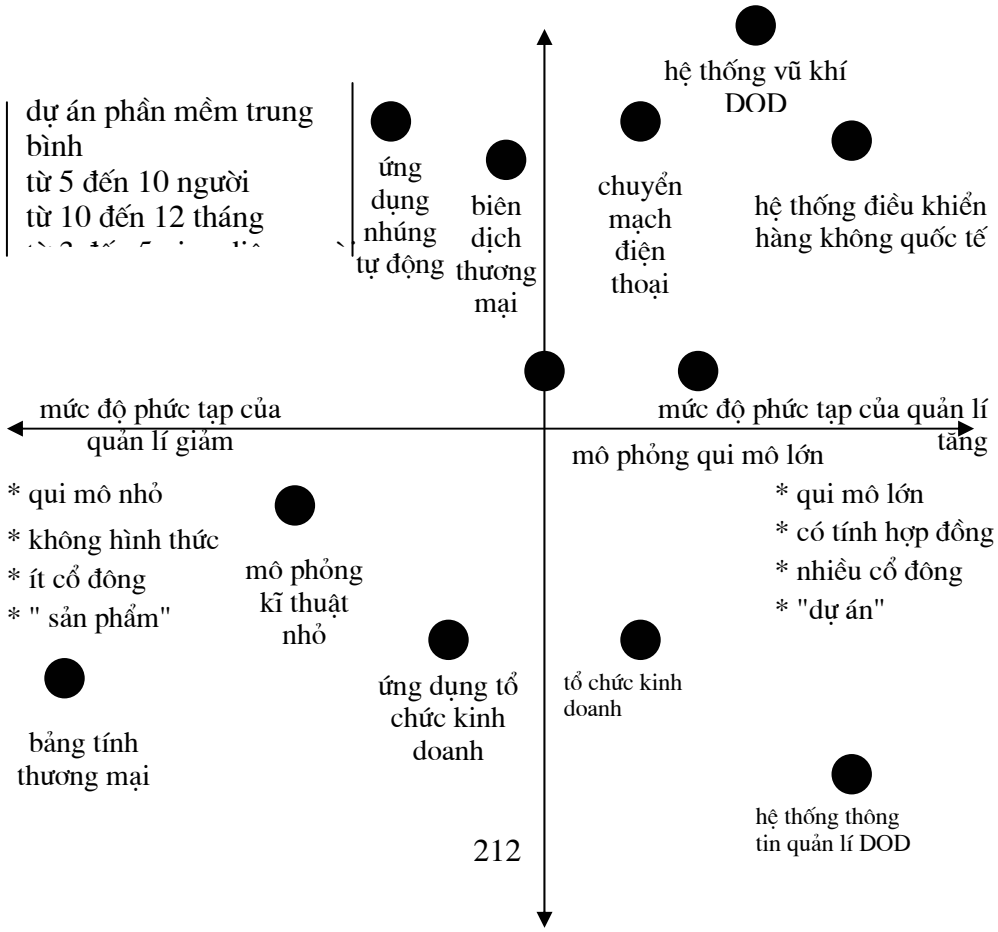
14.1 Phân biệt các tiến trình

Quá trình quản lí điều chỉnh trong một lĩnh vực hay một dự án cụ thể thì có 2 yếu tố phân biệt : mức độ phức tạp của kĩ thuật và mức độ phức tạp của quản lí. Hình 14 -1 minh hoạ hai hướng có thể thay đổi được của quá trình và đưa ra một số ví dụ về các dự án ứng dụng .Cách thức xem xét, chất lượng của việc điều khiển tạo tác, mức độ ưu tiên và số các thông số của các quá trình khác bị chi phối bởi vị trí của dự án đó trên 2 hướng . Hình 14 2 tổng kết lại các mức ưu tiên khác nhau theo 2 hướng .

Cơ cấu một quá trình không phải là một dự án với quá trình thao tác cụ thể theo một công thức đã được xác định trước để thành công . Các quyết định phải được xem xét, và các phương pháp, kĩ thuật, văn hoá, qui cách và tổ chức phải được điều chỉnh cho lĩnh vực cụ thể để đạt được thành công. Sự khác nhau cơ bản giữa các quá trình được tổ chức thành 6 thông số: thông số kích thước của dự án và 5 thông số tác động đến việc thể hiện các dự án và qui mô kinh tế trong COCOMOII. Có nhiều ý kiến cho rằng quản lí dự án phần mềm phải được chú ý trong khi điều chỉnh cơ cấu quá trình để có thể tạo ra những hành động thiết thực.

Chiều tăng mức độ phức tạp của kĩ thuật

- nhúng, xử lí thời gian thực, phân phối, chịu lỗi
- đặc tính cao , di chuyển được
- chưa từng có , cấu trúc mới



Mức độ phức tạp kĩ thuật giảm

- Tự động 1 chiều
- Thực thi tương tác
- Nhiều hệ thống đã có, ứng dụng lại

Hình 14.1 Hai hướng thay đổi của tiến trình

14.1.1.Qui mô

Trong việc biến đổi cơ cấu qui trình phần mềm để xác định được những yêu cầu của dự án yếu tố quan trọng nhất cólẽ là toàn bộ qui mô các ứng dụng của phần mềm. Có rất nhiều cách để xác định qui mô này bao gồm : số lượng dòng mã ở chương trình nguồn, số các hàm được chỉ định, số lượng các trường hợp sử dụng và số tiền. Từ triển vọng biến đổi tiến trình, cách xác định qui mô chủ yếu của nó là kích cỡ của nhóm. Khi mà số các vấn đề tăng lên thì một điều hết sức quan trọng là sự trao đổi giữa các cá nhân với nhau. Mặt khác một qui mô không kinh tế có thể tác động nghiêm trọng đến mục tiêu của dự án.

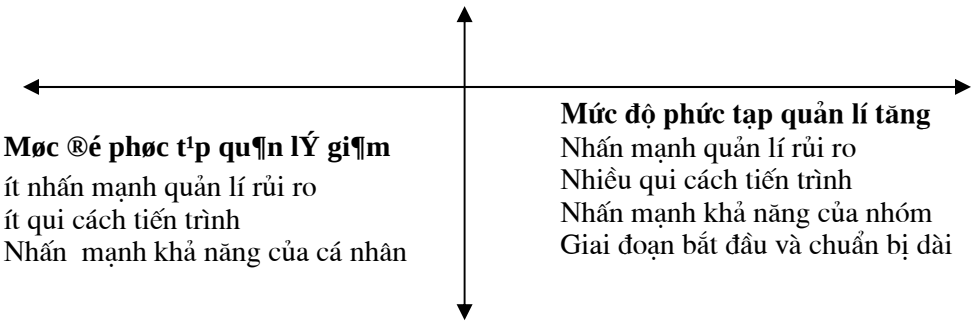
Mức độ phức tạp của kĩ thuật tăng

Yêu cầu nhiều kinh nghiệm trong lĩnh vực

Giai đoạn khởi đầu và chuẩn bị dài

ít lặp lại sự quản lí rủi ro

ít đoán trước được giá cả và kế hoạch



Mức độ phức tạp của kĩ thuật giảm

Nhấn mạnh hơn tài sản hiện có

Giai đoạn khởi đầu và chuẩn bị ngắn hơn

ít làm lại

Có thể đoán trước được giá cả và kế hoạch

Kinh nghiệm trong việc làm dự án của tôi đã cho thấy 5 là số người tối ưu trong một nhóm công nghệ. Nhiều sinh viên thì chỉ ra rằng hầu hết mọi người có thể quản lí từ 4 đến 7 công việc cùng một lúc. Thực hiện một phép ngoại suy đơn giản ta có: Về cơ bản có một số cách quản lí khác nhau đạt được yêu cầu quản lí cho nhóm 1 người (rất nhỏ), nhóm 5 người (nhỏ), nhóm 25 người (vừa), nhóm 125 người (lớn) , nhóm 625 người (rất lớn). Khi kích thước nhóm tăng lên thì trình độ quản lí được chỉ dẫn tăng lên khoảng 5 lần. Mô hình này được sử dụng để mô tả cho các tiến trình khác nhau trong các dự án với kích thước khác nhau.

Dự án kích thước rất nhỏ không yêu cầu khả năng quản lí cao (lập kế hoạch, khả năng truyền đạt, phối hợp, đánh giá quá trình, duyệt, và khả năng chỉ đạo).Cần một ít sự chỉ dẫn cho việc tạo tác trung gian. Dòng công việc đi theo một hướng . Việc tiến hành phụ thuộc nhiều vào kĩ năng của cá nhân.

Dự án nhỏ (5 người) yêu cầu trình độ quản lí không cao lắm nhưng người lãnh đạo nhóm phải hướng nhóm vào mục đích chính. Cần có một ít trao đổi về tạo tác trung gian giữa các thành viên trong nhóm. Đặc biệt là dự án đó rất dễ lập kế hoạch , quản lí sát sao và rất dễ thay đổi. Có một số ít luồng công việc riêng rẽ. Việc thi hành vẫn chủ yếu dựa vào kĩ năng của từng cá nhân. Sự thuần thục của quá trình là tương đối không quan trọng. Công cụ của từng cá nhân có thể tác động lên việc thi hành.

Dự án có kích thước trung bình (25 người) yêu cầu trình độ quản lí cao vừa phải bao gồm khả năng đồng bộ hoá luồng công việc của nhóm và cân bằng tài nguyên của người quản lí dự án phần mềm. Luồng công việc của cả nhóm cần thiết phải được xem xét , kết hợp và đánh giá . Cần có sự trao đổi tạo tác trung gian giữa các nhóm. Dự án được lập kế hoạch và chỉ đạo một cách chính thức và tác động của sự thay đổi thường là nhẹ nhàng. Có một số luồng công việc đồng thời và mỗi luồng lại bao gồm nhiều luồng riêng rẽ. Việc thi hành dựa trên kĩ năng của từng cá nhân là chủ yếu đặc biệt là của người lãnh đạo nhóm. Tính thuần thục của quá trình đã có giá trị. Môi trường có tác động đến việc thi hành nhưng sự thành công có thể đạt được với các công cụ chắc chắn.

Dự án có kích thước lớn (125 người) yêu cầu trình độ quản lí đáng kể cho cả người quản lí toàn bộ dự án và một số người quản lí dự án con để đồng bộ luồng công việc ở từng mức độ của dự án chính và các dự án con và cân bằng tài nguyên. Việc làm có ý nghĩa cho luồng công

việc cả nhóm là sự truyền đạt , xem xét , kết hợp , và đánh giá. Việc trao đổi tạo tác trung gian được nhấn mạnh để có thể trao đổi kết quả của công nghệ giữa các nhóm khác nhau. Dự án được lập kế hoạch và điều khiển chính thức và sự thay đổi nhìn chung là đắt. Một số lượng lớn luồng công việc đồng thời là cần thiết , mỗi luồng đó bao gồm nhiều luồng riêng rẽ. Việc thi hành chủ yếu dựa vào kĩ năng của từng cá nhân đặc biệt là những người quản lí dự án con và người lãnh đạo nhóm. Việc thi hành dự án phụ thuộc vào số người trung bình vì 2 lí do sau đây:

+Có rất nhiều các công việc cho bất kì dự án lớn nào đặc biệt là với những dự án có nhiều luồng công việc.

+ Khả năng bổ sung và duy trì một số lượng lớn những người đặc biệt là nhỏ

Tính thuần thục quá trình là cần thiết đặc biệt trong việc lập kế hoạch và điều khiển được thể hiện bằng các mặt : sự giao việc, sự xúc tiến công việc và dự tính cổ đông.Một môi trường tích hợp được yêu cầu để thay đổi việc quản lí, tạo tác tự động và duy trì tính ổn định của các tạo tác mới.

Dự án rất lớn (625 người) đòi hỏi trình độ quản lí rất cao đối với cả người quản lí nhiều dự án phần mềm và những người quản lí dự án con để đồng bộ luồng công việc ở các mức độ của toàn bộ dự án và những dự án con và để cân bằng nguồn tài nguyên. Việc truyền đạt , xem xét, kết hợp , và đánh giá là việc làm có ý nghĩa đối với luồng công việc của cả nhóm . Việc trao đổi tạo tác được nhấn mạnh để trao đổi các kết quả của công nghệ ở các nhóm khác nhau. Dự án được lên kế hoạch và hướng dẫn một cách chính thức và việc thay đổi kế hoạch thường dẫn đến việc lập lại kế hoạch. Có nhiều luồng công việc đồng thời trong nhóm và mỗi cái lại bao gồm các luồng riêng rẽ. Việc tiến hành công việc dựa trên kĩ năng của từng cá nhân đặc biệt là của người quản lí dự án con và người lãnh đạo nhóm. Việc tiến hành dự án còn phụ thuộc vào số người trung bình.

Sự thuần thục và kinh nghiệm trong lĩnh vực sẽ tránh được rủi ro và đảm bảo sự đồng bộ hoá nhu cầu của những cổ đông. Trong sự phát triển của các nhóm thì một môi trường thuần thục, có khả năng tích hợp cao và phổ biến là sự cần thiết để thay đổi quản lí, sản xuất tự động , duy trì tính ổn định trong các tạo tác mới và cải tiến việc đầu tư vào các quá trình phổ biến, công cụ phổ biến , khái niệm phổ biến và độ đo phổ biến

Bảng 1-4 tổng kết lại điểm khác nhau cơ bản trong các gốc tiến trình đối với dự án lớn và dự án nhỏ.

Bảng 1-4 phân biệt tiến trình dựa vào sự khác nhau về kích thước của các dự án.

Các gốc tiến trình	**Nhóm nhỏ hơn**	Nhóm lớn hơn
Pha vòng đời	Ranh giới giữa các giai đoạn là không rõ ràng	Sự chuyển tiếp giữa các giai đoạn được xác định tốt để đồng bộ việc xúc

		tiến các hoạt động đồng thời
Tạo tác	Tập trung vào kĩ thuật tạo tác Có một số ranh giới riêng rẽ Yêu cầu rất ít tạo tác quản lí	Thay đổi việc quản lí kĩ thuật tạo tác cái mà có thể đưa đến nhiều ranh giới. Tạo tác quản lí quan trọng
Nỗ lực phân phối luồng công việc	Cần nhiều người có khả năng đóng nhiều vai trò trong nhiều luồng công việc	Cần tỉ lệ chuyên gia cao Nhiều người và nhiều nhóm tập trung vào một luồng công việc cụ thể
Điểm kiểm tra	Có nhiều sự kiện không chính thức trong việc duy trì sự ổn định của kĩ thuật Không có sự phá vỡ kế hoạch	Rất ít sự kiện chính thức Việc đồng bộ giữa các nhóm phải thực hiện hàng ngày.
Nguyên tắc quản lí	Lập kế hoạch , điều khiển dự án và tổ chức không chính thức	Lập kế hoạch , điều khiển dự án và tổ chức chính thức
Nguyên tắc tự động	Quản lí riêng rẽ	Cơ sở hạ tầng đảm bảo sự phù hợp , môi trường cập nhật có giá trị cho tất cả các nhóm.Tích hợp các công cụ thêm vào để cung cấp cho việc điều khiển dự án và điều khiển sự thay đổi

14.1.2.Liên kết hoặc cạnh tranh

Trình độ của sự hợp tác và phối hợp giữa các cổ đông (người mua, nhà phát triển phần mềm, người sử dụng, những nhà thầu nhỏ, người bảo dưỡng ...) có thể dẫn đến việc xác định một cách có ý nghĩa tiến trình đó như thế nào. Giới hạn của quá trình này có thể đi từ cấu kết đến đối lập với nhau. Nhóm cấu kết có một mục tiêu chung, bổ xung khả năng cho nhau và có

sự liên hệ chặt chẽ. Nhóm đối lập có mục tiêu đối lập nhau, cạnh tranh hoặc không có kĩ năng đầy đủ, liên hệ rời rạc .

Một sản phẩm được đầu tư, phát triển, tiếp thị và bán bởi cùng một tổ chức để đạt được một mục đích chung (ví dụ là lợi nhuận). Một tổ chức nhỏ, có thứ tự có thể được đầu tư vào nhóm có sự cố kết về kĩ năng cơ bản và có sự trao đổi từng ngày tốt giữa các thành viên trong nhóm .

Việc thiết lập một hợp đồng lớn không có sự cạnh tranh giữa các nhóm là rất khó. Một nhà thầu hợp đồng thường rất ít khi có đủ điều kiện cho phần mềm và sự thành thạo trong lĩnh vực do đó thường phải hợp sức với các nhà thầu con - người mà cạnh tranh với họ về mục tiêu lợi nhuận. Nhà đầu tư và người dùng muốn cực tiểu hoá giá cả và cực đại hoá tính năng của phần mềm, giảm thời gian tiếp thị trong khi đó người thầu lại muốn cực đại hoá lợi nhuận. Những nhóm lớn mà không có khả năng sắp xếp và đồng bộ hoá nhu cầu của các cổ đông đang bị thách thức. Tất cả các yếu tố đó có khuynh hướng dẫn đến phân rã nhóm có cố kết và phải được quản lí liên tục. Hình 14.2 tổng kết các điểm khác biệt cơ bản trong các gốc tiến trình cho rất nhiều mức độ cố kết giữa các cổ đông.

Bảng14.2: Sự phân biệt các tiến trình dựa vào sự khác nhau trong sự cố kết giữa các cổ đông

Các gốc tiến trình	*ít cổ đông, nhóm cố kết*	**Nhiều cổ đông , quan hệ đối lập**
Pha vòng đời	Ranh giới giữa các giai đoạn là không rõ ràng	Sự chuyển biến giữa các giai đoạn được xác định tốt để đồng bộ tiến độ giữa các hoạt động đồng thời .
Tạo tác	ít có sự yêu cầu tạo tác quản lí chi tiết	Tạo tác quản lí là rất quan trọng, đặc biệt là trong kinh doanh, và đánh giá tình trạng.
Nỗ lực cấp phát luồng công việc	Không đánh giá cao	Đánh giá cao để đảm bảo chắc chắn sự nhất trí giữa các cổ đông
Các điểm kiểm tra	Có nhiếu sự kiện không chính thức	3 hoặc 4 sự kiện chính thức Có rất nhiều kĩ thuật không chính thức đi qua nên cần thiết phải đồng bộ quyết định kĩ thuật Đồng bộ giữa các nhóm gây cản trở cho tiến độ trong từng

		tuần
Nguyên tắc quản lí	Lập kế hoạch điều khiển dự án và tổ chức không chính thức	Lập kế hoạch điều khiển dự án và tổ chức chính thức
Nguyên tắc tự động	Không có ý nghĩa	Cần thiết môi trường cổ đông trực tuyến

14.1.3.Tiến trình mềm dẻo hay không mềm dẻo

Mức độ không mềm dẻo ,một cách chính thức và được thay đổi thành quyền tự do vốn có trong hợp đồng của dự án cụ thể (tài liệu có thể xem được, trong trường hợp kinh doanh và trong kế hoạch phát triển) sẽ có một tác động quan trọng lên việc thực hiện tiến trình của dự án. Có rất nhiều hợp đồng không chặt chẽ như trong xây dựng một sản phẩm thương mại trong một đơn vị thương mại của một công ty phần mềm(như ứng dụng Microsoft hay tập đoàn phần mềm Rational phát triển công cụ), sự phức tạp của quản lí là rất nhỏ. Trong các loại tiến trình phát triển, tập hợp tính năng, thời gian tiếp thị, ngân sách và chất lượng có thể tất cả đều được thoả hiệp tự do và thay đổi không nhiều. Ví dụ nếu công ty muốn loại trừ một số tính năng trong sản phẩm dưới sự phát triển để chiếm được chia sẻ thị trường từ sự cạnh tranh bằng việc tăng khả năng phát hành các sản phẩm. Nó có thể thực thi được quyết định này dưới 1 tuần. Toàn bộ nỗ lực kết hợp có thể liên quan đến chỉ những người quản lí phát triển, quản lí thị trường, đơn vị quản lí kinh doanh kết hợp với một số ràng buộc quan trọng.

Một mặt khác với các hợp đồng rât chặt chẽ , có thể mất nhiều tháng để cho phép thay đổi kế hoạch phát hành. Ví dụ để ngăn ngừa nỗ lực phát triển lớn của khách hàng, nó có thể được mong muốn kết hợp với một sản phẩm thương mại mới vào trong tất cả các thiết kế cho thế hệ tiếp theo của hệ thống điều khiển hàng không. Loại thay đổi này có thể yêu cầu sự phối hợp giữa những nhà thầu khoán , cơ quan tài trợ và người sử dụng (có thể là hiệp hội điều khiển hàng không và các công ty hàng không quan trọng) các cơ quan chứng nhận (chính quyền hàng không liên bang) sự kết hợp giữa các nhà thầu khoán cho phần cứng của các hệ thống. Một hệ thống có qui mô lớn , mất giá thảm khốc có sự cứng nhắc lớn về hợp đồng và yêu cầu các cách tiếp cận quản lí có hiệu quả. Hình 14.3 tổng kết các điểm khác nhau cơ bản trong các gốc của tiến trình với nhiều mức độ mềm dẻo của các tiến trình

Gốc của tiến trình	Các tiến trình mềm dẻo	Các tiến trình không mềm dẻo
Các pha vòng đời	Chịu đựng sự phóng túng trong giai đoạn	Yêu cầu nhiều nền tảng tin cậy cho ràng buộc của

	cam kết	pha khởi đầu
Tạo tác	Có khả năng thay đổi trong các trường hợp kinh doanh	Điều khiển sự thay đổi một cách cẩn thận trong các trường hợp kinh doanh
Nỗ lực phân phối luồng công việc	Không có ý nghĩa	Tăng trình độ quản lí và đánh giá luồng công việc
Các đIểm kiểm tra	Có nhiều sự kiện không chính thức đối với tính ổn định kĩ thuật bảo dưỡng	3 hoặc 4 sự kiện chính thức, đồng bộ hoá giữa cãc nhóm cổ đông ngăn cản tiến độ trong các ngày hoặc trong các tuần
Nguyên tắc quản lí	Không có ý nghĩa	Yêu cầu trung thực hơn trong việc lập kế hoạch và điều khiển dự án
Nguyên tắc tự động	Không có ý nghĩa	Không có ý nghĩa

14.1.4.Sự thuần thục tiến trình

Mức độ thuần thục tiến trình của các tổ chức phát triển phần mềm được xác định bằng mô hình khả năng thuần thục của viện công nghệ phần mềm [SEI, 1993 ; 1993b ; 1995]là chìa khoá dẫn đến sự phức tạp của quản lí. Việc quản lí một tiến trình thuần thục (mức độ 3 hoặc hơn) thì đơn giản hơn nhiều so với việc quản lí một tiến trình không thuần thục (mức độ 1 hoặc 2). Các tổ chức với 1 tiến trình thuần thục cụ thể có kinh nghiệm cao trong sự phát triển phần mềm và đã có các tiến trình có mức độ cao để đối chiếu nhằm dự đoán trước việc lập kế hoach và thực thi tiến trình. Loại đối chiếu này bao gồm xác định tốt các cách thức, các công cụ tự động hoá tiến trình , đào tạo nhân lực, lập độ đo, tạo mẫu, và mẫu luồng công việc. Việc biến đổi các tiến trình thuần thục của tổ chức cho các dự án cụ thể thường là các công việc dễ làm. Hình 14.4 tổng kết các điểm khác nhau cơ bản trong các gốc tiến trình với nhiều mức độ thuần thục của tiến trình.

Hình 14.4 sự phân biệt các tiến trình bằng sự khác nhau về mức độ thuần thục của các tiến trình

Gốc tiến trình	Tổ chức có mức độ thuần thục 3 hoặc 4	Tổ chức có mức độ thuần thục 1
Pha vòng đời	Thiết lập tốt điều kiện cho sự chuyển tiếp các giai đoạn	Không có ý nghĩa
Tạo tác	Thiết lập tốt khuôn dạng , nội dung , cách thức của sản phẩm	Mẫu tự do
Nỗ lực phân phối luông công việc	Thiết lập tốt các nền tảng	Không có nền tảng
Các điểm kiểm tra	Thiết lập tốt sự kết hợp của các sự kiện chính thức và không chính thức	Không có ý nghĩa
Nguyên tắc quản lí	Dự đoán trước việc lập kế hoach và đánh giá tình trạng của mục tiêu	Lập kế hoạch và điều khiển dự án không chính thức
Nguyên tắc tự động	Yêu cầu trình độ tự động hoá cao cho hành trình khứ hồi của công nghệ , quản lí sự thay đổi và trang bị dụng cụ cho tiến trình	Sự tự động hoá ít hoặc không kết nối với các lĩnh vực tự động hoá

14.1.5.Rủi ro kiến trúc

Mức độ thực hiện được của kĩ thuật đã được chứng minh là một hướng quan trọng trong việc xác định tiến trình của một dự án cụ thể trước khi có một hợp đồng cho một sản phẩm có qui mô lớn. Có nhiều nguồn gốc của sự rủi ro. Một số nguồn gốc quan trọng và tái diễn là các hoạt động của hệ thống (tiện ích tài nguyên , thời gian đáp ứng, thông lượng, độ chính xác) tiềm lực để thay đổi (thêm các chức năng mới, kết hợp các kĩ thuật mới, sự thích hợp với điều kiện thao tác động) và sự tin cậy của hệ thống (dự đoán trước hành vi, chịu đựng lỗi). Một trình độ mà ở đó sự rủi ro có thể bị loại đi trước khi bắt đầu xây dựng có thể có sự phân nhánh đột ngột trong việc điều chỉnh tiến trình. Hình 14.5 tổng kết các điểm khác nhau cơ bản trong các gốc tiến trình với nhiều mức độ của rủi ro kiến trúc.

Hình 14.5 Phân biệt tiến trình dựa vào sự khác nhau về rủi ro kiến trúc

Các gốc tiến trình	Sự thể hiện tính khả thi có kiến trúc hoàn chỉnh	Sự thể hiện tính khả thi không có kiến trúc
Pha vòng đời	Làm lại nhiều giai đoạn khởi đầu và chuẩn bị	ít có sự làm lại sớm , có nhiều sự xây dựng lại
Tạo tác	Sớm có độ rộng và sâu trong kĩ thuật tạo tác	Không có ý nghĩa
Nỗ lực phân phối công việc	Mức độ nỗ lực thiết kế cao Mức độ thực hiện và đánh giá thấp	Mức độ thực hiện và đánh giá cao để giải quyết sự tăng lên của việc loại bỏ và làm lại
Các điểm kiểm tra	Nhấn mạnh vào các luận chứng có thể thực hiện được	Nhấn mạnh vào sự chỉ dẫn , tài liệu và mô phỏng
Nguyên tắc quản lí	Không có ý nghĩa	Không có ý nghĩa
Nguyên tắc tự động	Yêu cầu sớm nhiều tài nguyên môi trường trong vòng đời	ít có sự yêu cầu sớm môi trường lệnh trong vòng đời

14.1.6.Kinh nghiệm trong lĩnh vực

Kinh nghiệm trong lĩnh vực của một tổ chức phát triển chi phối khả năng hội tụ đến một kiến trúc có thể chấp nhận được sau một số ít nhất lần làm lại. Một tổ chức đã xây dựng 5 thế hệ công tắc điều khiển rada có khả năng hội tụ tới 1 vạch ranh giới kiến trúc thích hợp cho ứng dụng rada mới với 2 hoặc 3 mẫu phát hành lại. Một tổ chức có kĩ năng xây dựng ứng dụng đầu tiên của rasda có thể yêu cầu 4 hoặc 5 mẫu phát hành trước khi hội tụ đến một vạch ranh giới thích hợp. Hình 14.6 tổng kết các đIểm khác nhau cơ bản trong các gốc của tiến trình với nhiều trình độ kinh nghiệm trong lĩnh vực.

Hình 14.6 sự phân biệt các tiến trình dựa vào kinh nghiệm trong lĩnh vực

Gốc tiến trình	Nhóm có kinh nghiệm	Nhóm không có kinh nghiệm
Pha vòng đời	Có giai đoạn công nghệ ngắn	Có giai đoạn công nghệ dài
Tạo tác	ít có sự loại bỏ và làm lại trong tập các đòi hỏi và	Nhiều sự loại bỏ và làm lại trong tập các đòi hỏi và thiết

	thiết kế	kế
Nỗ lực phân phối luồng công việc	Nhu cầu và mục đích ở mức độ thấp	Nhu cầu và mục đích ở mức độ cao
Các điểm kiểm tra	Không có ý nghĩa	Không có ý nghĩa
Nguyên tắc quản lí	ít nhấn mạnh quản lí rủi ro Không cần sự đánh giá tình trạng một cách thường xuyên	Yêu cầu đánh giá tình trạng một cách thường xuyên
Nguyên tắc tự động	Không có ý nghĩa	Không có ý nghĩa

14.2 Ví dụ về dự án qui mô nhỏ chống lại dự án qui mô lớn

Việc phân tích sự khác nhau về pha, luồng công việc , tạo tác của 2 dự án dựa trên kết thúc đối lập của dải quản lí phức tạp thể hiện sự khác nhau giữa 2 dự án có thể có. Theo sự tổng quát hoá nhằm chỉ ra các hướng của sự mềm dẻo, mức độ ưu tiên, tính trung thực mà có thể thay đổi khi khung công việc của tiến trình được áp dụng cho các ứng dụng , dự án và lĩnh vực khác nhau.

Hình 14.7 minh hoạ sự khác nhau về phân phối kế hoạch đối với các dự án lớn và nhỏ qua các pha vòng đời. Một dự án kinh doanh nhỏ (ví dụ một ứng dụng Windows gồm 50000 dòng lệnh nguồn viết bằng Visual Basic, được xây dựng bằng nhóm có 5 người) có thể yêu cầu 1 tháng bắt đầu, 2 tháng chuẩn bị, 5 tháng xây dựng và 2 tháng chuyển tiếp. Một dự án kinh doanh lớn, phức tạp (300.000 dòng nguồn của một chương trình nhúng được xây dựng bởi nhóm 40 người) có thể yêu cầu 8 tháng bắt đầu 14 tháng chuẩn bị , 20 tháng xây dựng và 8 tháng chuyển tiếp. So sánh tỉ lệ thời gian tiêu thụ của từng pha trên toàn bộ vòng đời để thấy được sự khác nhau rõ ràng hơn.

Sự khác nhau lớn nhất là thời gian có liên quan mà ở đó kiến trúc vòng đời xảy ra. điều đó có liên quan đến thời gian của giai đoạn công nghệ so với thời gian của giai đoạn sản phẩm. Với dự án nhỏ tỉ lệ đó vào khoảng 30/70 với dự án lớn tỉ lệ gần bằng 45/55.

Bảng 14.7 Việc phân phối kế hoạch qua các pha đối với các dự án lớn và nhỏ.

Công nghệ	Sản phẩm

Lĩnh vực	Sự bắt đầu	Sự chuẩn bị	xây dựng	Chuyển tiếp
Dự án kinh doanh nhỏ	10%	20%	50%	20%
Dự án kinh doanh lớn và phức tạp	15%	30%	40%	15%

Một điểm mấu chốt trong sự khác nhau của hai dự án là tác dụng của rất nhiều tiến trình thành phần vào sự thành công hay thất bại của dự án. Điều đó phản ánh vai trò quan trọng của lãnh đạo và trình độ quản lí rủi ro Bảng 14.8 Danh sách các luồng công việc sắp xếp theo thứ tự ưu tiên của chúng.

Danh sách đó cho biết tỉ mỉ về một số điểm mấu chốt trong việc phân biệt sự thành công. Không có tiến trình thành phần nào là không quan trọng mặc dù có một số tiến trình thì quan trọng hơn một số tiến trình khác.

- Thiết kế là điểm mấu chốt của cả hai lĩnh vực. Một sản phẩm thương mại thiết kế tốt là điểm phân biệt cơ bản trong thương trường và là cơ sở thuận lợi cho việc phát hành sản phẩm mới. Dự án lớn, phức tạp được thiết kế tốt là cơ sở cho việc dự đoán và quản lí lợi nhuận .
- Quản lí có vai trò quan trọng đối với các dự án lớn, nơi mà hậu quả của lỗi lập kế hoạch, lỗi phân phối tài nguyên, sự không thống nhất về nhu cầu của các cổ đông, và các yếu tố không cân bằng khác có thể có hậu quả nghiêm trọng đối với toàn bộ động lực của nhóm. Quản lí không có ý nghĩa quan trọng lắm đối với các nhóm nhỏ, nơi mà khả năng mất liên lạc ít xảy ra và hậu quả của nó không nghiêm trọng.
- Sự triển khai có vai trò quan trọng hơn trong các sản phẩm kinh doanh nhỏ bởi vì số người sử dụng lớn trên cơ sở môi trường và cá nhân đa dạng. Một dự án lớn và phức tạp cụ thể có sự triển khai ở 1 vị trí. Hệ thống kế thừa và những thao tác liên tục có thể đem đến một số rủi ro, tuy nhiên những vấn đề này thường dễ hiểu và có tập các mục tiêu rõ ràng ,cố định.

Bảng 14.8 Sự khác nhau trong mức ưu tiên luồng công việc giữa dự án lớn và nhỏ

Thứ hạng	Dự án kinh doanh nhỏ	Dự án kinh doanh lớn , phức tạp
1	Thiết kế	Quản lí
2	Thi hành	Thiết kế
3	Triển khai	Yêu cầu
4	Yêu cầu	đánh giá

5	đánh giá	Môi trường
6	Quản lí	Thi hành
7	Môi trường	Triển khai

Một số điểm khác nhau mấu chốt đã có trong việc thực thi rất nhiều loại tạo tác của dự án. Hình 14.9 cung cấp một ví dụ để nhận thức về sự khác nhau đó

Bảng 14.9 Sự khác nhau về tạo tác giữa dự án lớn và nhỏ

Tạo tác	Dự án kinh doanh nhỏ	Dự án kinh doanh lớn và phức tạp
Cấu trúc phân mảnh công việc	1 trang bảng tính với 2 trình độ của các phần tử WBS	Hệ thống quản lí tài chính với 5 hoặc 6 trình độ của phẩn tử WBS
Trường hợp kinh doanh	Bảng tính và sổ ghi chép ngắn	3 quyển sách dự kiến bao gồm sách kĩ thuật , sách vê giá cả và kinh nghiệm liên quan
Lệnh xem xét được	Bài khái niệm 10 trang	Bản ghi chi tiết kĩ thuật hệ thống con 200 trang
Kế hoạch phát triển	Kế hoạch 10 trang	Kế hoạch phát triển 200 trang
Bản chỉ định phát hành và số lượng phát hành	Có 3 bản chỉ định phát hành tạm thời	8 đến 10 bản chỉ định phát hành tạm thời
Bộ miêu tả kiến trúc	Nhiều nhất 5 người sử dụng , 50 biểu đồ UML , 20 trang văn bản và đồ hoạ	Nhiều nhất 25 người , 200 biểu đồ UML 100 trang văn bản và đồ hoạ
Phần mềm	500.000 dòng mã lệnh viết bằng Visual Basic	300.000 dòng mã lệnh viết bằng C++
Mô tả phát hành	10 trang ghi chú phát hành	100 trang tóm tắt
Triển khai	Khoá đào tạo người sử dụng Bán các tài liệu hướng	Kế hoạch chuyển tiếp Kế hoạch cài đặt

	dẫn	
Sách hướng dẫn sử dụng	Giúp đỡ trực tuyến và sách hướng dẫn sử dụng 100 trang	Sách hướng dẫn sử dụng 200 trang
Đánh giá tình trạng	Xem xét dự án từng quí	Xem xét quản lí dự án hàng tháng

Chương 15
Những sơ thảo về dự án tiên tiến
(Modern project profiles)

Trong các chương đầu đã giới thiệu về việc quản lý phần mềm truyền thống. Trong đó miêu tả một cách khách quan sơ thảo một dự án truyền thống, một nền kinh tế phần

mềm truyền thống và những nguyên tắc truyền thống về quản lý phần mềm. Các chương tiếp theo đã mô tả một khung tiến trình và những qui tắc quản lý cần thiết để cho một trạng thái chuyển sang một tiến trình phần mềm hiện đại.Các chương 15,16,17 hoàn tất việc giới thiệu một khung quản lý phần mềm hiện đại và đề cập lại các kết quả truyền thống đồng thời miêu tả việc thực hiện tiến trình quản lý phần mềm hiện đại.

Những điểm mấu chốt:

- Trong các sơ thảo về dự án tiên tiến, xu hướng khả quan hoặc không khả quan sẽ sớm trở nên rõ ràng hơn trong vòng đời phần mềm.
- Tích hợp xuất hiện sớm và liên tục làm nhiệm vụ kiểm tra các mẫu thiết kế.
- Hầu hết những rủi ro sẽ được giải quyết cho đến cuối pha chi tiết hoá (elaboration phase). Các pha xây dựng (construction phase) và chuyển tiếp (transition phase) – pha mà giá phải trả cho sự rủi ro thường lớn nhất – có thể đem đến nhiều bất ngờ.
- Cột mốc lớn tập trung ở các kết quả đã được kiểm chứng.

Chương 1 đã giới thiệu năm vấn đề cố hữu của các dự án truyền thống. Một tiến trình tiên tiến khai thác một số cách tiếp cận nhất định để giải quyết những vấn đề nói trên:

1. Tích hợp phải kéo dài và những sai sót trong thiết kế sau này được giải quyết bằng cách đưa tích hợp vào giai đoạn công nghệ. Điều này đạt được qua việc tích hợp liên tục của một cấu trúc được hỗ trợ bởi các cách kiểm chứng có thể thực thi của "kịch bản" chính.
2. Việc giải quyết những rủi ro sau này được thực hiện bằng cách chú trọng cách tiếp cận cấu trúc đầu tiên, trong đó các thành phần cấp cao của hệ thống được xây dựng kĩ lưỡng trong vòng đời phần mềm.
3. Tình trạng trì trệ trong việc phân tích sự phân rã theo chức năng (xuất phát từ các yêu cầu) được tránh bởi việc tổ chức những đặc tả cấp thấp theo nội dung hơn là theo sự phân rã sản phẩm (bởi hệ thống con, các thành phần,…)
4. Những mối quan hệ đối nghịch giữa các cổ đông được tránh bằng cách nhà sản xuất sẽ cung cấp những kết quả rõ ràng hơn và khách quan hơn trong suốt vòng đời phần mềm,
5. Việc tập trung vào các tài liệu và xem xét các cuộc gặp gỡ như truyền thống được thay thế bằng việc tập trung vào các kết quả có thể kiểm chứng được và một tập các mẫu đã được chỉ định rõ, với những chú thích chặt chẽ hơn và có thể tự động mở rộng khả năng hỗ trợ cho môi trường không giấy.

Những cách thức mà các dự án tiên tiến mạnh giải quyết năm vấn đề truyền thống sẽ được thảo luận chi tiết ở phần sau. Vì cách giải quyết vấn đề 4 và 5 có mối quan hệ chặt chẽ nên sẽ cùng được trình bày trong mục 15.4. Mục 15.5 và 15.6 đề cập đến những dự án tiên tiến trong ngữ cảnh 10 nguyên tắc quản lý phần mềm tốt nhất và một loạt các ứng dụng thực tiễn tốt nhất của phần mềm.

15.1 .Tích hợp liên tục

Phát triển lặp tạo ra cấu trúc đầu tiên cho phép tích hợp xuất hiện như hoạt động kiểm tra pha thiết kế và cho phép dò tìm lỗi thiết kế để sớm khắc phục trong vòng đời phần mềm. Cách tiếp cận này tránh được tích hợp quá tải (big-bang integration) ở cuối một dự án bằng cách tích hợp liên tục trong suốt dự án. Hình 15-1 minh hoạ sự khác nhau giữa sơ thảo tiến trình của một dự án tiên tiến mạnh và sơ thảo tiến trình của dự án truyền thống (đã được giới thiệu ở hình 1-2). Cách tiếp cận cấu trúc đầu tiên buộc tích hợp phải được đưa vào pha thiết kế qua việc xây dựng

các thể nghiệm. Các thể nghiệm này không loại bỏ được sai sót trong thiết kế, thay vào đó chúng khiến những sai sót này xảy ra vào giai đoạn công nghệ, khi những sai sót đó có thể được giải quyết có hiệu quả trong ngữ cảnh mục tiêu của vòng đời phần mềm. “Cơn ác mộng tích hợp xuôi (downstream), việc sửa chữa tạm thời và những tình thế khó khăn của phần mềm được tránh khỏi. Kết quả là một thiết kế mạnh hơn và dễ bảo trì hơn ra đời.

Bảng 15-1. *Sự khác nhau trong phân phối giá trị cho luồng công việc giữa một tiến trình truyền thống và một tiến trình hiện đại.*

LUỒNG CÔNG VIỆC CỦA CÔNG NGHỆ PHẦN MỀM	NHỮNG CHI PHÍ CHO TIẾN TRÌNH TRUYỀN THỐNG	NHỮNG CHI PHÍ CHO TIẾN TRÌNH TIÊN TIẾN
Quản lý	5%	10%
Môi trường	5%	10%
Các yêu cầu	5%	10%
Thiết kế	10%	15%
Thực thi	30%	25%
Đánh giá	40%	25%
Triển khai	5%	5%
Tổng số	100%	100%

Tích hợp liên tục trong quá trình phát triển lặp còn cung cấp sự hiểu biết sâu sắc hơn về sự cân bằng chất lượng. Các đặc tính vốn có của hệ thống trong cấu trúc (như tốc độ thực hiện, khả năng chịu lỗi và khả năng bảo trì) ngày càng rõ ràng hơn trong quá trình, khi các kết quả vẫn có khả năng hiệu chỉnh mà không ảnh hưởng tới chỉ tiêu giá thành và thời gian. Một đề tài lặp của những dự án phát triển lặp có sơ thảo về phân phối giá trị rất khác với các quá trình truyền thống.

Bảng 15-1 chỉ ra sự khác nhau trong một sơ thảo tiến trình tiên tiến về sự phân bổ giá trị giữa các luồng công việc khác nhau của dự án. Bảng này là kết quả của sự kết hợp đơn giản giữa bảng 1-1 (một kiểu phân phối truyền thống) và bảng 10-1 (một kiểu phân phối tiên tiến mặc định). Theo kinh nghiệm của bản thân, sự khác nhau chính trong một tiến trình tiên tiến thành công là ở những chi phí cho việc đánh giá và thử nghiệm trong toàn bộ vòng đời phần mềm. Những dự án truyền thống, rơi vào tình trạng tích hợp không hiệu quả và phát hiện chậm những vấn đề thiết kế quan trọng, tiêu tốn gần 40% tổng số tài nguyên hay thậm chí hơn thế cho tích hợp và các hoạt động kiểm tra. Trong khi các dự án tiên tiến với một tiến trình lặp hoàn thiện cung cấp một sản phẩm mà chỉ mất khoảng 25% tài nguyên cho các hoạt động này.

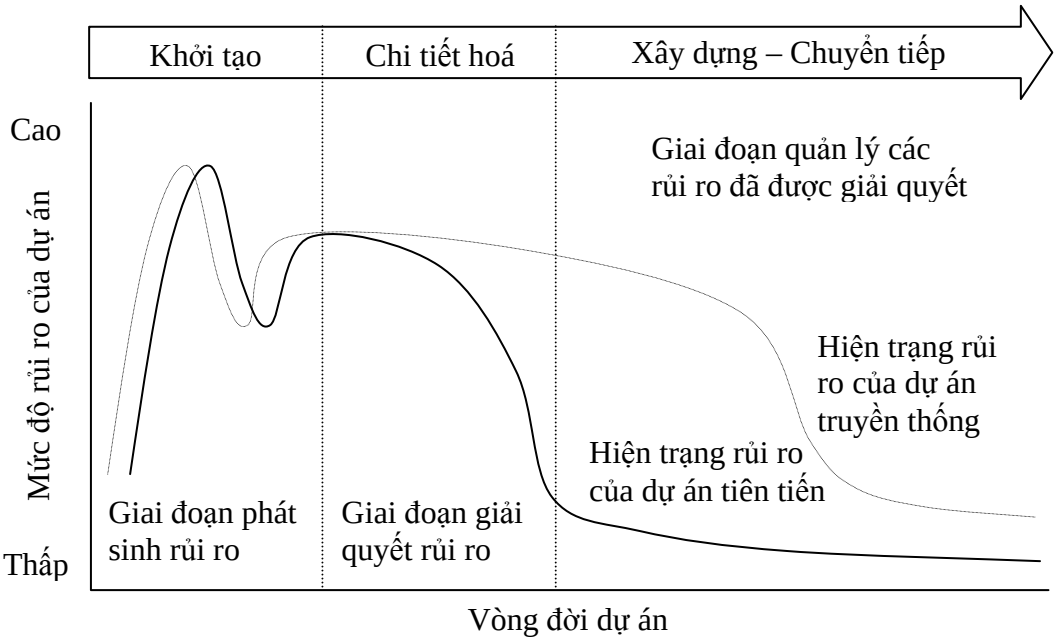
15.2.**Giải quyết sớm những rủi ro**

Giai đoạn công nghệ trong vòng đời phần mềm (gồm pha khởi tạo và pha chi tiết hoá) tập trung vào việc đối mặt với những rủi ro và giải quyết chúng trước khi trao tài nguyên cho giai đoạn sản xuất. Những dự án truyền thống thường tiến hành những công việc đơn giản trước, vì thế

việc kiểm nghiệm sớm được xúc tiến. Một tiến trình tiên tiến bắt đầu với 20% quan trọng các yêu cầu, phân lớp người sử dụng, các thành phần và những rủi ro. Đây là điểm cốt lõi trong nguyên tắc quan trọng nhất của tôi: tiếp cận theo cấu trúc đầu tiên. Xác định cấu trúc hầu như không bao gồm các bước đơn giản, hữu hình mà chúng ta dễ dàng đạt được. Kết quả của toàn bộ vòng đời phần mềm được đúc rút qua hơm 30 năm quản lý phần mềm cung cấp những kinh nghiệm quí báu trong việc kiểm soát những rủi ro.

- 80% công nghệ dành cho 20% các yêu cầu. Nỗ lực để hiểu thấu đáonhững yêu cầu đặt ra trước khi trao tài nguyên cho việc phát triển trên qui mô lớn. Không nên sớm tập trung vào những yêu cầu về độ tin cậy cao và tính theo dõi được.
- 80% giá trị phần mềm phục vụ cho 20% các thành phần. Chi tiết hoá các thành phần có giá trị giới hạn đầu tiên để cho việc lập kế hoạch và điều chỉnh giá trị sớm được hiểu rõ trong vòng đời phần mềm.
- 80% các lỗi sinh ra do 20% các thành phần. Chi tiết hoá các thành phần có độ tin cậy chuẩn để việc đánh giá các hoạt động có đủ căn cứ đạt tới độ tin cậy cần thiết.
- 80% các phần mềm bị phân mảnh và phải làm lại là do 20% những thay đổi. Chi tiết hoá các thành phần có mức độ thay đổi tới hạn đầu tiên để những thay đổi sớm xuất hiện .
- 80% nguồn tiêu thụ tài nguyên (thời gian thực thi, không gian đĩa, bộ nhớ) là từ 20% các thành phần. Việc chi tiết hoá các thành phần có tốc độ thực thi chuẩn để đầu tiên để sự cân bằng công nghệ về độ tin cậy, khả năng thay đổi và hiệu quả kinh tế được thực hiện sớm ngay khi có thể trong vòng đời phần mềm.
- 80% sự tiến triển của công việc được thực hiện nhờ 20% năng lực con người. Cần phải chắc chắn rằng nhóm khởi tạo việc lập kế hoạch dự án và thiết kế cấu trúc có chất lượng cao nhất. Một kế hoạch đầy đủ và cấu trúc tương xứng có thể thành công với một nhóm xây dựng ở mức trung bình. Nhưng khi kế hoạch không trọn vẹn hay cấu trúc không tương xứng có thể sẽ không thành công, thậm chí với nhóm các chuyên gia xây dựng.

Hình 15-2 so sánh sơ thảo kiểm soát rủi ro của một dự án tiên tiến với sơ thảo của một kiểu truyền thống đã giới thiệu trong hình 1-3.



Hình 15-2. *Hiện trạng rủi ro của dự án tiên tiến điển hình*

15.3 Những yêu cầu phát triển

Cách tiếp cận truyền thống phân rã hệ thống những yêu cầu thành các hệ thống con, rồi thành các thành phần và nhỏ nhất là khối yêu cầu. Cơ cấu những yêu cầu được tổ chức như vậy để việc theo dõi được đơn giản. Với một phần mềm sớm nhấn mạnh đầu tiên vào những yêu cầu, thứ hai là thiết kế, tiếp đó hoàn thiện tính theo dõi được giữa yêu cầu và thiết kế các thành phần, thì xu hướng tự nhiên là làm cho việc thiết kế cấu trúc phát triển thành một tổ chức gần như song song với tổ chức cấu trúc các yêu cầu. Không có gì đáng ngạc nhiên khi việc phân rã theo chức năng vấn đề không gian dẫn tới sự phân rã theo chức năng giải pháp không gian.

Hầu hết các cấu trúc hiện đại mà sử dụng các thành phần thương mại, các thành phần kế thừa, các tài nguyên được phân phối và các phương pháp hướng đối tượng có khả năng thoả mãn đầy đủ các yêu cầu. Hiện nay có nhiều mối quan hệ phức tạp giữa việc trình bày các yêu cầu và việc thiết kế các thành phần.

Các yêu cầu ở mức cao nhất được giữ lại như viễn cảnh, còn các yêu cầu ở mức thấp được giữ lại trong tiêu chuẩn phát triển được gắn vào mỗi mẫu trung gian. Các mẫu này, minh hoạ trong hình 15-3, được dự định phát triển cùng với tiến trình với độ tin cậy ngày càng cao. Đây là điểm khác biệt cơ bản với các cách tiếp cận việc quản lý các yêu cầu theo truyền thống, trong đó độ tin cậy này được theo đuổi quá sớm trong vòng đời phần mềm.

15.4 Sự hợp tác giữa các cổ đông

Rất nhiều khía cạnh trong tiến trình phát triển truyền thống làm cho mối quan hệ giữa các cổ đông đi đến không tin tưởng lẫn nhau, dẫn đến khó khăn trong việc cân bằng các yêu cầu, các tính năng của sản phẩm và các kế hoạch. Một tiến trình lặp nhiều hơn, với những mối quan hệ công việc có hiệu quả hơn giữa các cổ đông, cho phép sự cân bằng được xây dựng trên cơ sở sự hiểu biết khách quan giữa mọi người. Tiến trình này đòi hỏi các khách hàng, người sử dụng và các “monitor” có cả chuyên môn về ứng dụng và phần mềm, chú trọng việc phân phối các hệ thống sử dụng được (hơn là những tiêu chuẩn bắt buộc không rõ ràng và các dạng hợp đồng) và cho phép người thầu thu lợi từ những hợp đồng tốt. Tiến trình này cũng đòi hỏi một cơ cấu phát triển tập trung để đạt được sự hài lòng của khách hàng và chất lượng sản phẩm cao (để có thể thu lợi).

Việc chuyển tiếp từ sự trao đổi các mẫu giấy tờ sang kiểm nghiệm các kết quả trung gian là một trong những cơ chế quyết định thúc đẩy quan hệ hợp tác giữa các cổ đông. Những cột mốc lớn cung cấp các kết quả rõ ràng và các thông tin phản hồi từ người sử dụng. Như bảng 15-2 chỉ ra, các thiết kế đều bị xem là có lỗi cho đến khi được chứng thực: dự án không phát triển tiếp đến chừng nào các đối tượng đều đã được kiểm chứng. Điều kiện ràng buộc này không loại bỏ việc “thương lượng lại” của các đối tượng và các mốc kết quả lớn cho ta hiểu sâu sắc hơn về sự cân bằng vốn có trong các yêu cầu, thiết kế, kế hoạch và công nghệ.

Trong bảng 15-2, những kết quả dự kiến có thể vẫn chưa khả quan. Một tiến trình lặp tiên tiến mà chú trọng tới các kết quả có thể kiểm chứng qua thực tế (hơn là những bản tóm tắt trên giấy) đòi hỏi tất cả các cổ đông phải được giáo dục về điểm khác nhau quan trọng giữa các kết quả dự kiến và các kết quả thực tế. Chẳng hạn, một lỗi thiết kế được phát hiện sớm khi chi phí cho việc sửa lỗi là có thể chấp nhận, thì tiến trình vẫn được xem là khả quan hơn là coi lỗi đó như một vấn đề lớn.

Bảng 15-2. *Các kết quả của những mốc lớn trong một tiến trình tiên tiến.*

KẾT QUẢ DỰ KIẾN	KẾT QUẢ THỰC TẾ
Những kiểm nghiệm sớm đã bộc lộ những vấn đề về thiết kế và sự nhập nhằng trong một cấu trúc.	Những kiểm nghiệm đã bộc lộ những tính năng quan trọng và những rủi ro của những hệ thống phần mềm phức tạp, khi những rủi ro có thể được khắc phục trong ngữ cảnh mục tiêu của vòng đời phần mềm.
Vấn đề về các yêu cầu đặt ra được bộc lộ nhưng khả năng theo dõi được các yêu cầu chi tiết còn thiếu.	Những thay đổi yêu cầu được xem trọng như sự cân bằng trong thiết kế.
Việc thiết kế bị xem như có lỗi cho đến khi được chứng thực về tính đúng đắn.	Những tiến bộ và những vấn đề về công nghệ đều rõ ràng cho sự hợp nhất trong các dự án lặp tiếp theo.

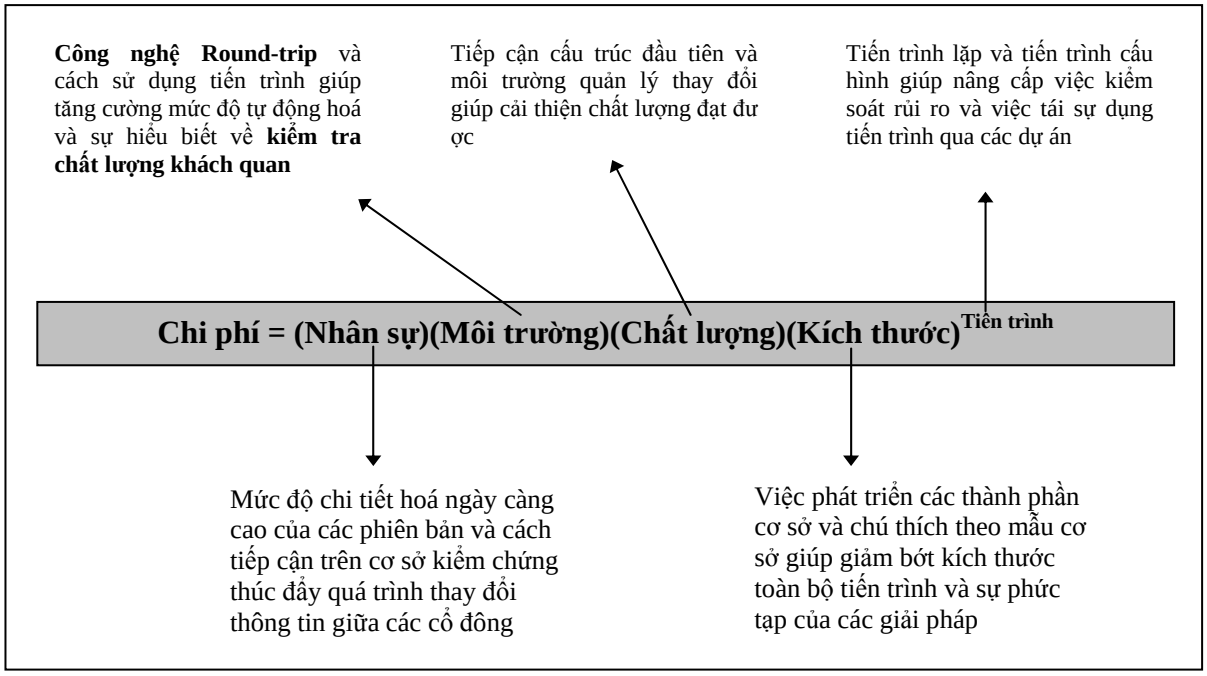
15.5 10 Nguyên tắc quản lý phần mềm tốt nhất

10 nguyên tắc quản lý phần mềm tốt nhất của tôi được giới thiệu ở chương 4 như là cơ sở của khung tiến trình phần mềm và những nguyên lý cơ bản của nó. Để tóm tắt một sơ thảo dự án tiên tiến, những đoạn sau đây đề cập lại mỗi nguyên tắc và mô tả sự mong đợi dự án mà có kết hợp với ứng dụng thành công của mỗi nguyên tắc này. Đồng thời cung cấp sự mô tả ngắn gọn, ở mức cao các tính năng và lợi nhuận của một tiến trình tiên tiến dưới con mắt của nhà quản lý dự án phần mềm.

1. Tiến trình dựa trên cách tiếp cận theo **cấu trúc đầu tiên**. Việc sớm chú trọng vào cấu trúc tạo ra một nền tảng vững chắc cho 20% của stuff (các yêu cầu, các thành phần, phân lớp người sử dụng, những rủi ro và lỗi) chi phối toàn bộ thành công của dự án.
2. Thiết lập một **tiến trình vòng đời lặp** sớm đối mặt với những rủi ro. Một khung kế hoạch động được hỗ trợ bởi một tiến trình lặp làm cho việc kiểm soát rủi ro có hiệu quả hơn và việc thực thi có thể dự đoán trước. Giải quyết những vấn đề cấp bách đầu tiên dẫn đến pha xây dựng hoàn toàn có thể dự đoán được mà không có bất ngờ nào xảy đến, cũng như việc trình bày những nguồn kinh phí và việc lập lịch không có khả năng dự đoán trước được hạn chế đến mức tối đa.
3. Chuyển tiếp những phương pháp thiết kế để nhấn mạnh **việc phát triển các thành phần cơ sở**. Tính phức tạp của kết quả phần mềm gần như là một chức năng của một số các mẫu tích hợp nhân tạo. Phải tìm cách giải quyết để giảm phần nào sự phức tạp trong quản lý.
4. Thiết lập một **môi trường quản lý thay đổi**. Cơ chế động của phát triển lặp, bao gồm những luồng công việc được đồng thời thực hiện bởi các nhóm khác nhau trên các mẫu dùng chung, đòi hỏi các giới hạn được kiểm soát cao.
5. Tăng cường thay đổi qua các công cụ hỗ trợ **công nghệ khứ hồi** (round-trip engineering). Sự tự động hoá cho phép các nhóm có nhiều thời gian hơn cho giai đoạn công nghệ và giảm bớt thời gian cho những nhiệm vụ đầu tiên.
6. Nắm giữ các mẫu thiết kế một cách nghiêm ngặt, với **chú thích theo mẫu cơ sở**. Một chú thích công nghệ cho thiết kế cho phép việc điều khiển phức tạp, định giá khách quan và những phân tích tự động.

7. Cung cấp tiến trình cho việc **kiểm tra chất lượng khách quan** và tiến trình đánh giá. Những chỉ báo về sự tiến triển và chất lượng được lấy trực tiếp từ các mẫu, cung cấp những nhận thức hữu ích hơn về các xu hướng và sự tương quan với các yêu cầu.
8. Sử dụng **cách tiếp cận trên cơ sở kiểm chứng** để đánh giá các mẫu trung gian. Tích hợp xuất hiện sớm và liên tục trong suốt vòng đời phần mềm. Các kết quả trung gian khách quan và rõ ràng.
9. Lập các phiên bản trung gian trong các nhóm phiên bản được sử dụng với **mức độ chi tiết hoá ngày càng cao**. Các yêu cầu, các bản thiết kế, và các kế hoạch cần phải tương xứng. Các phiên bản phần mềm hữu ích sẵn có trong vòng đời phần mềm.
10. Thiết lập một **tiến trình có cấu hình có thể thay đổi** một cách hiệu quả (về kinh tế). Các phương pháp, công nghệ, công cụ, và kinh nghiệm có thể được áp dụng trực tiếp trên một phạm vi rộng.

Trong cuốn sách này, tôi đã nhấn mạnh tầm quan trọng của sự cân bằng. Từ những viễn cảnh, thái độ hết sức khách quan của các nhà quản lý dự án phần mềm là coi trọng như nhau 10 nguyên tắc trên. Hình 15-4 tóm tắt sự ngang bằng này trong ngữ cảnh sự cân bằng của nền kinh tế phần mềm cơ sở.



Hình 15-4. *Ứng dụng cân bằng các nguyên tắc tiên tiến để đạt hiệu quả kinh tế*

15.6 Những ứng dụng thực tiễn tốt nhất của quản lý phần mềm

Những ứng dụng thực tiễn tốt nhất của quản lý phần mềm được tạo ra bởi các tác giả và các tổ chức công nghiệp. Một trong những kết quả rõ ràng là phần mềm Acquisition Best Practices Initiative, được đỡ đầu bởi Bộ quốc phòng Mĩ để "cải tiến và tổ chức lại tiến trình quản lý phần mềm của chúng ta." Brown đã tóm tắt bước khởi đầu [Brown, 1996], gồm có 3 thành phần: Hội đồng phần mềm Airlie (gồm có những người có công lớn trong công nghiệp phần mềm), 7 nhóm thuộc các lĩnh vực khác nhau (bao gồm những người làm việc trong ngành công nghiệp và cho chính phủ), một nhóm quản lý chương trình (bao gồm những nhà quản lý dự án có kinh nghiệm). Mỗi thành phần đưa ra những lời khuyên và các kết quả, và xem xét lại công việc của các thành phần khác.

Hội đồng phần mềm Airlie được tổ chức nhằm mục đích tập hợp các nhà quản lý thành công trong các dự án phần mềm lớn, các tác giả có tiếng trên thế giới, và những uỷ viên ban quản trị chịu trách nhiệm về sự phát triển phần mềm ở những công ty lớn. Một trong những sản phẩm của Hội đồng là bộ 9 ứng dụng thực tiễn tốt nhất. Hội đồng đã cố gắng chú trọng vào những ứng dụng thực tiễn mà có ảnh hưởng lớn nhất đến việc cải tiến những nguyên tắc quản lý phần mềm cho những dự án phần mềm lớn và điều chỉnh những phức tạp bên trong nó.

9 ứng dụng thực tiễn tốt nhất sẽ được mô tả dưới đây, cùng với lời chú thích chúng kết hợp với khung tiến trình, những qui định trong quản lý, và 10 nguyên tắc tốt nhất mà tôi đã đề cập (Những lời trích dẫn được in nghiêng.)

1. *Kiểm soát rủi ro hình thức*

▲ Sử dụng một tiến trình lặp đối mặt với rủi ro.

2. *Chấp nhận các giao diện.*

> ◢ *Trong khi chúng ta có thể sử dụng những từ khác, thì điều này thực sự có ý tưởng giống nguyên tắc tiếp cận cấu trúc đầu tiên của tôi. Việc hiểu được những giới hạn cấu trúc buộc dự án phải chấp nhận các giao diện ngoài khác nhau và các giao diện trong quan trọng, tất cả vốn có trong cấu trúc.*

3. *Việc kiểm tra hình thức.*

▲ Sự đánh giá luồng công việc trong suốt vòng đời phần mềm cùng với những công việc mang tính công nghệ khác, phải làm cân bằng một số thiếu sót khác nhau trong các chiến lược. Chiến lược ít quan trọng nhất có thể là việc kiểm tra hình thức, bởi vì chi phí cao, tỉ lệ phát hiện sai sót thấp đối với những sai sót cấu trúc trải rộng trên nhiều thành phần và sự phức tạp về thời gian.

4. *Lập lịch trên cơ sở hệ đo và quản lý.*

▲ Nguyên tắc quan trọng này có quan hệ trực tiếp với nguyên tắc **chú thích theo mẫu cơ sở** và **việc kiểm tra chất lượng khách quan**. Thiếu những chú thích nghiêm ngặt cho các mẫu, việc đo mức độ tiến triển và chất lượng các mẫu chỉ là sự đánh giá khách quan.

5. *Các cổng nhị phân ở mức "inch-pebble".*

▲ Ứng dụng này rất dễ hiểu sai. Rất nhiều dự án ứng dụng chính xác cách tiếp cận này sớm trong vòng đời và đã trình bày một kế hoạch rất chi tiết với phí tổn cao. 3 tháng sau đó, khi một vài yêu cầu thay đổi hoặc cấu trúc thay đổi, phần lớn kế hoạch chi tiết phải được giới hạn lại. Một cách tiếp cận tốt hơn sẽ đạt được độ tin cậy thích hợp với năng lực hiểu biết về các yêu cầu và cấu trúc. Tốt hơn các "inch-pebble", tôi muốn giới thiệu việc thiết lập các cột mốc trong giai đoạn công nghệ tiếp sau "inche-pebble" trong giai đoạn sản xuất. Đây là thông điệp chính đằng sau nguyên tắc
mức độ chi tiết hoá ngày càng cao của tôi.

6. *Tính rõ ràng của quá trình tiến triển chống lại việc lập kế hoạch.*

▲ Ứng dụng này - những thông tin thân thiện, cởi mở giữa các thành viên của dự án - thực sự cần thiết. Điều này rất rõ ràng mà không cần thêm bất cứ lời bình luận nào.

7. *Theo dõi sai sót chống lại những mục tiêu chất lượng.*

▲ Nguyên tắc quan trọng này có liên quan trực tiếp tới các nguyên tắc **cấu trúc đầu tiên** và **kiểm tra chất lượng khách quan** của tôi. Việc tạo rồi loại bỏ (make-or-break) sai sót và những mục tiêu chất lượng mang tính kiến trúc. Việc theo dõi sai sót là yêu cầu cần thiết để dẫn tới thành công.

8. *Quản lý cấu hình.*

▲ Hội đồng phần mềm Airlie nhấn mạnh việc quản lý cấu hình như chìa khoá để kiểm soát sự phức tạp và theo dõi những thay đổi đối với tất cả các mẫu. Hội đồng cũng nhận thấy tầm quan trọng của tự động hoá.

9. *Trách nhiệm giải trình quản lý nhân sự.*

▲ Nguyên tắc quản lý này đã quá rõ ràng mà không cần thêm bất cứ lời chú thích nào.

Có một sự tương đồng và sự trùng hợp ý tưởng giữa 10 nguyên tắc của tôi và những ứng dụng thực tiễn tốt nhất của Hội đồng phần mềm Airlie. Tuy vậy, tôi cho rằng Hội đồng còn bỏ sót vài nguyên tắc quan trọng: phát triển khả năng cấu hình và thành phần cơ sở, mẫu cơ sở, việc kiểm nghiệm. Sự bỏ sót này thật sự gây ngạc nhiên, bởi vì lý do căn bản nảy sinh các nguyên tắc thành phần cơ sở và mẫu cơ sở của tôi là vì chúng giúp làm giảm sự phức tạp trong quá trình phát triển. Điều này cũng là mục đích của Hội đồng phần mềm Airlie. Nguyên tắc kiểm nghiệm buộc tích hợp xuất hiện liên tục trong suốt vòng đời phần mềm và thúc đẩy mối quan hệ tốt đẹp giữa các cổ đông qua các phương tiện thông tin có hiệu quả. Bởi vì Hội đồng phần mềm Airlie chú trọng vào một lĩnh vực đặc thù - những hệ thống lớn, mang tính quốc gia – nên khả năng cấu hình ít quan trọng hơn.

Hai ứng dụng thực tiễn mà tôi không tính đến là việc kiểm tra và các cổng nhị phân ở mức “inche-pebble” mặc dù chúng rất hữu ích, rất được nhấn mạnh trong thực tiễn. Và còn có những nguyên tắc quan trọng khác nên được kể đến.

Chương 16
Thế hệ tiếp theo của quản lý kinh tế phần mềm
(Next generation software economic)

- Thế hệ tiếp theo của quản lý kinh tế phần mềm có thể mang lại trình độ quản lý kinh tế tốt hơn về qui mô và tăng lợi nhuận trong đầu tư. Chúng là những chỉ báo thực sự cho một ngành kinh doanh chắc chắn.
- Các tiến bộ kỹ thuật xa hơn trong công nghệ khứ hồi ngay lập tức tạo ra bước nhảy lượng tử tiếp theo trong nền kinh tế phần mềm.
- Mô hình ước lượng giá trong tương lai cần phải được dựa trên các đơn vị khối ban đầu tốt hơn được định nghĩa bằng các chú giải công nghệ phần mềm dễ hiểu như là UML (mô hình ngôn ngữ thống nhất của lập trình hướng đối tượng).

Thế hệ tiếp theo của quản lý kinh tế phần mềm hiện đang được một số tổ chức phần mềm tiên tiến áp dụng. Nhiều kỹ thuật, quy trình và phương pháp được mô tả trong khuôn khổ của cuốn sách này đã được áp dụng trong nhiều năm. Tuy nhiên, với một tổ chức phần mềm có quy mô trung bình thì việc đưa vào thử nghiệm một quy trình hoàn thiện và hiện đại là khó có trong khả năng. Chương này sẽ giới thiệu một vài giả thuyết táo bạo về tương lai của kinh tế học phần mềm. Một cấu trúc chung được đề xuất cho mô hình định giá được coi là có thể phù hợp nhất với khuôn khổ các quá trình trong cuốn sách này. Tôi nghĩ rằng cách tiếp cận mới này sẽ làm tăng tính chính xác và đúng đắn của việc dự toán giá của phần mềm và có thể chỉ ra những cải tiến đáng kể trong việc tiết kiệm qui mô của phần mềm. Những cải tiến này sẽ là khả thi với các tiến bộ trong môi trường phát triển phần mềm. Cuối cùng, ta sẽ xem xét lại các tiêu chuẩn của Boehm về hiệu năng của dự án phần mềm cổ điển và mô tả bằng thuật ngữ đối tượng cách mà khuôn khổ các tiến trình có thể phát triển toàn bộ nền kinh tế phần mềm thu được từ một dự án hay từ một tổ chức.

16.1 Mô hình định giá thế hệ tiếp theo

Các chuyên gia phần mềm có nhiều quan điểm khác nhau rõ rệt về kinh tế phần mềm và các biểu hiện của nó trong các mô hình định giá phần mềm là: các dòng mã nguồn khác với các điểm chức năng. Tiết kiệm về qui mô hay tăng qui mô. Các giải pháp về hiệu năng hay các giải pháp về chất lượng. Java hay C++. Hướng đối tượng hay hướng chức năng. Các khối thương mại hay sự phát triển theo đơn đặt hàng. Tất cả các vấn đề này thể hiện một cuộc cạnh tranh về công nghệ bị vây quanh bởi những sự khoa trương. Những sự thổi phồng hay chê bai quá đáng theo quan điểm của bạn làm cho rất khó phân biệt giữa sự thật và sự quá lời. Sự bất đồng mạnh mẽ là một chỉ báo cho một dòng chảy mạnh của một ngành công nghiệp, trong đó các kỹ thuật và công nghệ cạnh tranh với nhau đang được hoàn thiện một cách nhanh chóng. Tuy nhiên, có một kết quả là ta tiếp tục không có khả năng dự đoán chính xác các tài nguyên cần thiết cho một phần mềm cho trước. Hiện nay một sự dự toán xấp xỉ là khả thi nhưng một dự toán chính xác là không thể có được. Sẽ rất khó để có thể phát triển mô hình ước tính dựa vào kinh

nghiệm khi mà các dữ liệu đầu vào của dự án là bị nhiễu và hầu như không có tương quan với nhau, chúng dựa trên các tiến trình và kỹ thuật cơ sở khác nhau.

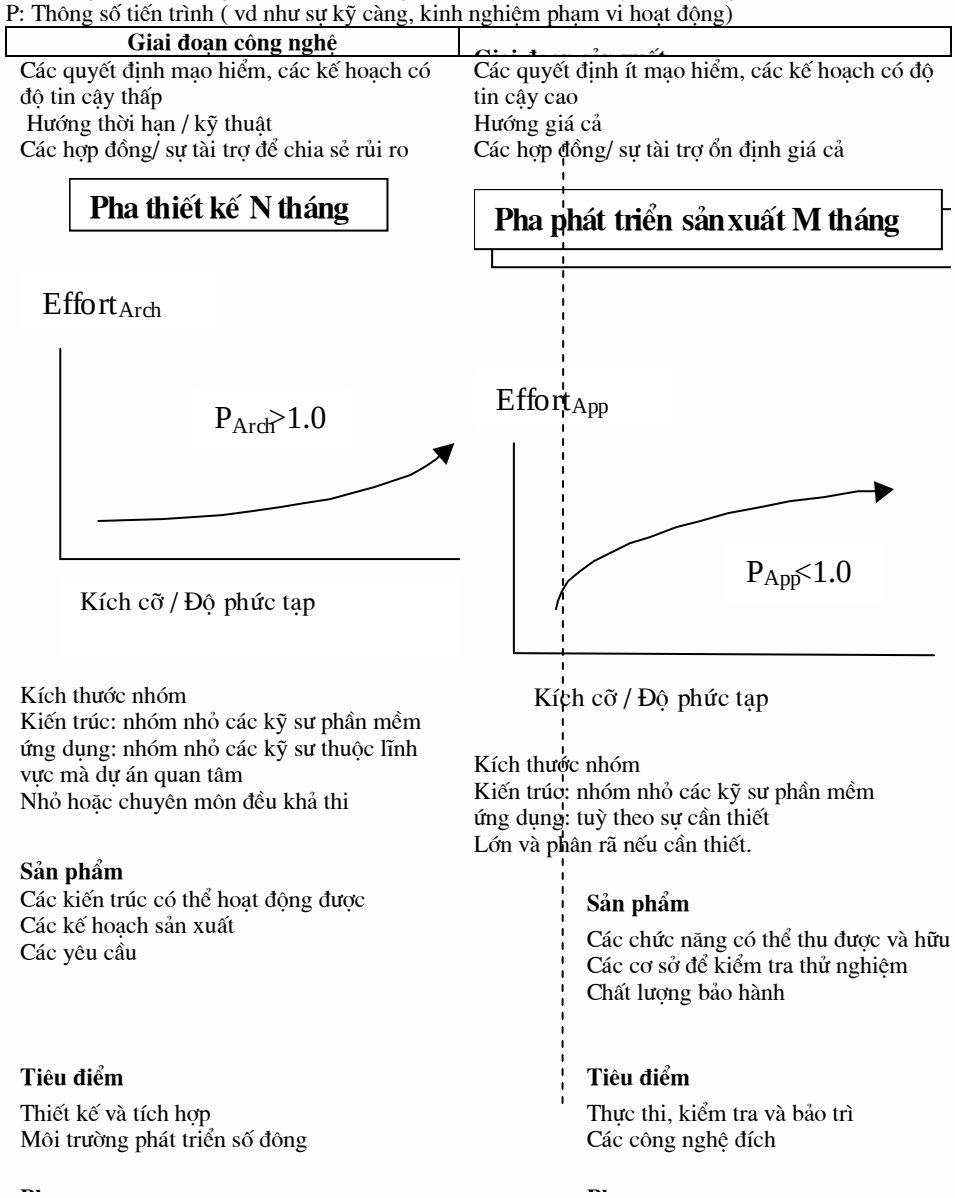
Một số mô hình định giá phần mềm phổ biến hiện nay không phù hợp với một tiến trình phần mềm lặp hướng vào một cách tiếp cận kiến trúc theo cách thứ 1.

Mặc dù nhiều cải tiến đã được các đại lý cung cấp công cụ định giá phần mềm phát triển để mở rộng kho dữ liệu thu được từ các dự án mới được cập nhật, nhiều cách định giá vẫn tiếp tục sử dụng các tiến trình cổ truyền dựa trên kinh nghiệm để ước lượng một dự thảo dự án hiện đại. Phần này ,tôi sẽ đề cập đến cách nhìn của mình về việc làm thế nào một mô hình định giá phần mềm được cấu trúc để ước lượng tốt nhất một quá trình phần mềm hiện đại. Có nhiều mô hình và kỹ thuật trong ngành công nghiệp này có thể thoả mãn một tập con của cách tiếp cận này. Mô hình định giá của tôi được đưa ra hoàn toàn trên lý thuyết, tôi không có một bằng chứng thực tế nào mô tả rằng cách tiếp cận này sẽ chính xác hơn các mô hình hiện nay. Mặc dù hầu hết các phương pháp và kỹ thuật cần thiết cho cách quản lý hiện đại hiện nay đều có sẵn, vẫn chưa có một dự án nào đủ xác đáng và hoàn thiện để ủng hộ cho sự xác nhận của tôi với các bằng chứng cụ thể.

Thế hệ tiếp theo của mô hình định giá phải phân cách rõ ràng các công nghệ kiến trúc và các sản phẩm ứng dụng, giống như kiểu kiến trúc tiến trình đầu tiên. Phí tổn của các khâu thiết kế , sản xuất, kiểm thử và bảo trì đường cơ sở kiến trúc là một hàm của qui mô, chất lượng, kỹ thuật, tiến trình và các kỹ năng nhóm. Có thể tồn tại một trạng thái không kinh tế về qui mô (số mũ lớn hơn 1) trong kiến trúc mô hình định giá vì nó là một thuộc tính cố hữu định hướng bởi việc nghiên cứu và phát triển theo mô hình hướng đối tượng. Khi một tổ chức đạt đến một kiến trúc bền vững, chi phí sản xuất là một hàm mũ của các biến kích thước, chất lượng và độ phức tạp với một khoảng giới hạn bền vững hơn của tiến trình và ảnh hưởng nhân sự. Mô hình trạng thái sản xuất của hàm giá phải là kinh tế về qui mô (số mũ <1) tương tự như mô hình kinh tế học cổ điển của thương mại bán buôn. Hình 16-1 tổng kết một mô hình định giá giả thuyết cho một quá trình phát triển kiến trúc thứ nhất.

Mô hình định giá thế hệ tiếp theo có thể ước lượng các cấu trúc có quy mô lớn với sự kinh tế về qui mô. Điều này ngụ ý rằng các số mũ quá trình trong suốt giai đoạn sản xuất là <1. Lý do giải thích là trong một hệ thống lớn hơn thì có nhiều cơ hội để khai thác sự tự động hoá và tái sử dụng các tiến trình, bộ phận và cấu trúc.

Effort=F(T_{Arch}, S_{Arch}, Q_{Arch}, P_{Arch})+ F(T_{App}, S_{App}, Q_{App}, P_{App})
Time = F(P_{Arch}, $Effort_{Arch}$)+ F(P_{App}, $Effort_{App}$)
Trong đó:
Effort : Công sức
Time: thời gian
T: Thông số công nghệ (sự ủng hộ của môi trường tự động hoá)
S: Thông số qui mô (như là các trường hợp ứng dụng, các điểm chức năng, các dòng mã nguồn)
Q: Thông số chất lượng (vd như khả năng vận chuyển, độ tin cậy, hiệu năng)
P: Thông số tiến trình (vd như sự kỹ càng, kinh nghiệm phạm vi hoạt động)



Trong các quy trình cổ điển , cấp nhỏ nhất của kỹ thuật tự động hoá cung cấp các hoạt động tổng phí của việc lập kế hoạch, điều hành dự án, và thay đổi sự quản lý dẫn đến dòng cường độ lao động tăng, và sự phi kinh tế về qui mô. Thiếu tự động hoá trong quản lý là vấn đề tồn tại trong các đa dự án, các tổ chức công việc theo dòng cũng như trong các dự án riêng lẻ. Cơ sở hạ tầng và môi trường thế hệ tiếp theo đang chuyển sang tự động và chuẩn hoá nhiều

hoạt động quản lý, bằng cách ấy, nó sẽ đòi hỏi một tỷ lệ phần trăm thấp hơn về công sức của các hoạt động tổng phí khi quy mô được tăng lên.

Tái sử dụng các tiến trình thông dụng thông qua sự lặp nhiều lần một dự án đơn, phát hành nhiều lần một sản phẩm riêng lẻ, hoặc có nhiều dự án trong một tổ chức cũng làm giảm các tác nhân gây ra thiếu kinh tế về qui mô. Các nguồn gây ra mẩu thừa và các công việc phải làm lại được khử đi bằng cách ứng dụng các kinh nghiệm tiền lệ và các tiến trình chắc chắn. Taọ lập nên những kế hoạch có giá trị dựa trên các chỉ tiêu tin cậy về hiệu suất của dự án và việc sử dụng những bộ phận cấu thành đáng sẽ làm giảm các nguồn đầu thừa đuôi thẹo và các công việc phải làm lại. Trong khi phần lớn những sự tái sử dụng các thành phần đưa đến kết quả là làm giảm nỗ lực sản xuất thì việc sử dụng lại các quá trình , các công cụ và kinh nghiệm lại gây ảnh hưởng trực tiếp đến tính kinh tế về qui mô.

Một sự khác biệt nữa trong mô hình định giá là các kiến trúc và các ứng dụng lại có các đơn vị khối khác nhau (qui mô và kích cỡ) và được biểu diễn trong không gian các giải pháp. Qui mô có thể được đo trong phạm vi các yếu tố quan trọng về kiến trúc (lớp, thành phần, tiến trình, nút) và kích cỡ được đo bằng đơn vị SLOC hay megabytes các mã thực hiện. Các phép đo này khác với các phép đo trong các bài toán không gian như các yêu cầu rời rạc hay các trường hợp úng dụng. Sự mô tả bài toán không gian này dẫn tới định nghĩa về không gian các giải pháp. Tuy nhiên, với mỗi vấn đề đặt ra ta sẽ có rất nhiều giải pháp, như được minh hoạ trong hình 16-2, và mỗi giải pháp có một giá trị mệnh đề khác nhau.

Chi phí là một tiêu chuẩn phân biệt giữa các giải pháp có thể. Việc ước lượng chi phí một cách chính xác và đúng đắn hơn có thể tìm được từ các giải pháp chi tiết cho các vấn đề. Do đó, mô hình ước lượng giá phải được điều hành bởi các tham số cơ bản của một giải pháp ứng viên. Nếu không một giá trị mệnh đề nào là chấp nhận được với bài toán thì cần phải tìm ra các giải pháp ứng cử khác hoặc là thay đổi các điều kiện của bài toán

Cuộc tranh luận giữa những quan điểm gay gắt về hướng chức năng và hướng dòng mã nguồn là một chỉ báo tốt cho thấy sự cần thiết của các độ đo cả về qui mô và kích thước. Tôi nghĩ rằng việc phân rã theo các điểm chức năng là chính xác hơn trong việc xác định số lượng qui mô của kiến trúc được yêu cầu, trong khi SLOC chính xác hơn trong việc mô tả kích thước các thành phần tạo nên sự thực hiện tổng thể. Điều thú vị trong việc sử dụng SLOC là có thể dễ dàng tự động hoá ghép nối và dễ dàng đạt được độ chính xác cao. Tuy nhiên, độ chính xác của SLOC khi được sử dụng làm một phép đo kích thước thì khá mơ hồ và dễ bị nhầm lẫn trong thông dịch khi SLOC được sử dụng trong việc so sánh tuyệt đối giữa các dự án và tổ chức khác nhau. Ta thấy rõ điều này, đặc biệt trong pha đầu tiên của dự án khi SLOC được sử dụng để biểu diễn qui mô. Nhiều dự án đã sử dụng SLOC như một độ đo thành công để tính kích thước trong các pha sau trong chu trình xây dựng phần mềm, khi mà phép đo quan trọng nhất là những thay đổi có liên quan từ tháng này sang tháng khác khi dự án dần đạt tới các phiên bản có thể phát hành.

Giá trị của việc sử dụng mô hình điểm chức năng là chúng tốt hơn khi mô tả qui mô tổng thể của một giải pháp, một cách độc lập với kích thước thật sự và ngôn ngữ thực thi khi thực hiện. Các điểm chức năng không dễ được rút ra từ một qui cách biểu diễn chính xác nào, tuy vậy , sự tự động hoá và theo dõi sự thay đổi lại khó thực hiện hoặc không rõ ràng.

Một chú giải chính xác đối với các mẫu thiết kế là một điều kiện tiên quyết cần thiết để phát triển trong độ chính xác mà qui mô của một thiết kế có thể ước lượng được. Trong tương lai, tôi hi vọng sẽ có cơ hội để tự động hoá một phép đo mới về qui mô thu được trực tiếp từ cách biểu diễn thiết kế bằng UML.

Tôi cho rằng hai cải tiến chính trong các mô hình định giá phần mềm thế hệ tiếp theo sẽ là:

1. Sự phân biệt giữa giai đoạn thiết kế và giai đoạn sản xuất sẽ buộc các công cụ định giá phải phân biệt giữa qui mô kiến trúc và kích thước thực thi. Điều này sẽ cho phép sự chính xác cao và sự trung thực hơn trong việc ước lượng vòng đời.
2. Các ký hiệu để biểu diễn thiết kế chính xác như UML sẽ tạo ra một cơ hội để định nghĩa các đơn vị đo qui mô mà được tiêu chuẩn hoá và do đó có thể tự động hoá và theo dõi kiểm tra. Các phép đo này cũng có thể được chỉ ra một các trực tiếp trong giá của sản phẩm .

Định lượng qui mô của kiến trúc phần mềm trong giai đoạn thiết kế là một lĩnh vực cần nghiên cứu. Trong suốt thập kỷ tới, hai sự đột phá trong tiến trình phần mềm có thể sẽ xảy ra , cả hai đều thu được từ các tiến bộ kỹ thuật trong môi trường phụ. Sự đột phá thứ nhất có thể là sự ra đời của các công cụ tích hợp sẵn có để tự động hoá việc trung chuyển thông tin giữa pha yêu cầu, thiết kế, thực thi và sự triển khai các yếu tố. Các công cụ này sẽ cho phép áp dụng công nghệ khứ hồi bao hàm toàn diện giữa các mẫu công nghệ. Sự đột phá thứ hai sẽ chú trọng việc lật đổ 4 pha công nghệ cơ bản hiện nay bằng 3 pha với việc tự động hoá các hoạt động liên quan đến các mã nguồn do con người tạo ra, bằng cách ấy, loại trừ sự cần thiết của một pha thực thi riêng biệt. Tiến bộ công nghệ này, được minh hoạ trong hình 16-3 sẽ cho phép sản xuất các chương trình có thể thực thi được một cách trực tiếp từ các biểu diễn UML mà không cần đến sự can thiệp của con người. Công cụ mô hình hoá trực quan cung cấp các tập con mã từ các mô hình UML đã sẵn sàng, nhưng sản xuất ra các tập con cạnh tranh thì vẫn còn là một vấn đề của tương lai.

Trong khi sự đột phá thứ nhất là đầy mạo hiểm nhưng rõ ràng thì sự đột phá thứ hai tạo ra một sự dịch chuyển khuôn cảnh quan trọng. Khi một nhóm kỹ sư công nghệ phần mềm có thể tạo ra những mẫu thực thi và triển khai trong một môi trường tự động hoá, không bao giờ có lỗi, tiến trình phát triển phần mềm có thể thay đổi một cách đầy kịch tính giống như sự thay đổi vượt bậc đã xảy ra khi việc sản xuất chip chuyển sang một tiến trình in tự động hoá.

16.2. Kinh tế học phần mềm thế hệ tiếp theo

Chương 1 giới thiệu 10 tiêu chuẩn đánh giá phần mềm hàng đầu của Boehm [Boehm,1987] như là một đối tượng đại diện cho trạng thái hiện tại của thực tiễn quản lý phần mềm. Khuôn khổ này có thể được sử dụng để tổng kết một số chủ đề quan trọng trong ngữ cảnh kinh tế học và nghiên cứu xem một khuôn khổ quản lý phần mềm hiện đại có thể thực hiện như thế nào. ở đây không có đủ các dữ liệu dự án để chứng minh cho khẳng định của tôi, nhưng tôi tin rằng các sự thay đổi đang được chờ đợi sẽ cung cấp một sự miêu tả tốt xem một nhà quản lý tổ chức cần cố gắng làm gì trong giai đoạn chuyển đổi sang một tiến trình mới (đoạn trích dẫn được in nghiêng).

1. Việc tìm và sửa các lỗi của phần mềm sau khi phát hành tốn gấp 100 lần so với tìm và sửa chúng trong pha thiết kế đầu tiên.

Các quá trình mới, các công nghệ phát triển dựa trên các thành phần và các khuôn khổ kiến trúc định hướng rõ ràng tói sự tăng mối liên kết. Trong nhiều lĩnh vực, và trong lĩnh vực lỗi phần mềm trong một thành phần riêng rẽ , tiến bộ trong các công nghệ bao bọc có thể giảm ảnh hưởng của tài nguyên một cách rõ ràng, có thể bằng một thứ tự của cường độ. Tuy nhiên, một cách tiếp cận theo kiến trúc thứ nhất có thể sẽ tăng gấp 10 hoặc gấp 100 sự phát triển các giải pháp về lỗi kiến trúc. Hệ quả là, tiến trình lặp chiếm một giá tiền lớn trong kiến trúc bên trong và các hoạt động đương đầu với mạo hiểm.

2. Bạn có thể rút gọn thời hạn phát triển phần mềm 25 % trên danh nghĩa, nhưng không hơn.

Công thức này có thể còn hiệu lực cho giai đoạn công nghệ trong vòng đời, khi nội dung tri thức của hệ thống được rút ra. Tuy nhiên, nếu giai đoạn công nghệ thành công trong việc đạt được một đường cơ sở bền vững- bao gồm kiến trúc, kế hoạch xây dựng và yêu cầu- sự rút ngắn thời hạn trong giai đoạn sản xuất có thể trở nên linh hoạt hơn. Nếu một loạt các tổ chức kinh doanh đang khấu hao giai đoạn công nghệ thông qua các sự tăng bội ,có thể có một cơ hội nhiều hơn cho sự phát triển đồng thời.

3. Mỗi USD bạn dùng để phát triển , bạn sẽ tiêu 2 cho bảo trì.

Thật khó để giải thích về tiêu chuẩn này, bởi vì có nhiều mô hình bảo trì khác nhau. Sự so sánh trong một số tuyệt đối tạo ra một cảm giác nhỏ trừ khi các dự án là duy nhất cho một loại . Một cách tốt hơn để đo tỷ lệ này là hệ số nhân giữa phát triển và bảo trì. (Phụ lục C mô tả các phép đo bảo trì). Một ảnh hưởng thú vị của sự phát triển lặp là đường ranh giới giữa phát triển và bảo trì đã trở nên mờ hơn. Một tiến trình lặp chắc chắn và một kiến trúc tốt có thể giảm cấp các mẩu thừa và công việc làm lại một cách đáng kể. Cho một sự đồng đẳng toàn thể giữa hoạt động phát triển và bảo trì, ta thấy rằng phép đo này có thể chuyển thành một quan hệ 1-1, trong đó hệ số nhân phát triển sẽ tương tự như hệ số nhân bảo trì.

4. Giá của sự phát triển và bảo trì phần mềm trước hết là một hàm của số dòng của chương trình nguồn.

Mệnh đề nói trên cho biết kích thước của sản phẩm là người lái đầu tiên của giá cả, và đơn vị cơ sở của kích thước là một dòng mã nguồn. Trong khi điều này đã trở nên rõ ràng trong các thế hệ trước của kỹ thuật phần mềm, nó đang trở nên không rõ ràng trong các kỹ thuật dựa trên các thành phần hiện nay. Các thành phần thương mại, tái sử dụng và các bộ sinh mã tự động có thể làm sai khác nghiêm trọng ý nghĩa của một dòng mã nguồn. Giá thành xây dựng sẽ phụ thuộc vào độ phức tạp của các hoá đơn của vật liệu. Việc sử dụng nhiều thành tố ,nhiều loại thành tố ,nhiều nguồn thành tố và nhiều thành tố do người sử dụng thiết kế hơn sẽ đòi hỏi nhiều công sức tích hợp hơn và sẽ hướng tăng giá. Việc sử dụng ít thành tố, loại thành tố ,nguồn thành tố và nhiều trang bị công cụ mạnh về công nghiệp hơn sẽ hướng giảm giá. Không may là, công nghiệp thành phần vẫn còn quá chưa đủ phát triển để ủng hộ các tiêu chuẩn cho một hoá đơn các vật liệu để có thể tăng độ tin cậy của việc ước lượng giá của nó. Vì vậy, mô hình định giá thế hệ tiếp theo phải trở nên ít phụ thuộc hơn vào số các dòng mã nguồn và trở nên nhạy cảm hơn với số rời rạc các thành phần và sự dễ dạng tích hợp chúng.

5. Sự thay đổi trong sự quan tâm của mọi người đối với sự những điểm khác nhau lớn nhất của năng suất phần mềm.

Cho bất cứ một dự án công nghệ nào trong đó tính tri thức là sản phẩm thực sự, các thừa số nhân có ảnh hưởng quan trọng là kỹ năng của nhân lực, làm việc nhóm và sự thúc đẩy. Trong phạm vi có thể, một tiến trình hiện đại gói gọn các yêu cầu về lực đòn bẩy cao của con người trong giai đoạn công nghệ , khi mà nhóm có liên quan nhỏ. Giai đoạn sản xuất, khi mà các nhóm cụ thể lớn hơn nhiều, sau đó các hoạt động nên ít phụ thuộc vào các ý kiến chuyên môn hiếm có.

6. Tỷ lệ toàn thể giữa giá phần mềm và phần cứng đang tăng lên. Năm 1955, nó là 15:85. Năm 1985, nó là 85:15.

Tôi không chắc phép đo này như thế nào hiện nay. Sự thông dụng của máy tính cá nhân và sự khác nhau trong giá phần mềm sau 1985 và trước 1985, đặc biệt là công cụ phầm mềm cho máy tính cá nhân đã thay đổi mối quan hệ này một cách không nghi ngờ gì nữa. ảnh hưởng chính của phép đo nói trên trong kinh tế học phần mềm là phần cứng đang trở nên rẻ đi. Các vòng xử lý, sự lưu trữ và dải thông mạng tiếp tục cho phép nhiều cơ hội tự động hoá mới. Hệ quả là, môi trường phần mềm giữ một vai trò quan trọng hơn. Trong phối cảnh tiến trình hiện đại, tôi có thể thấy môi trường làm việc nhiều hơn trong công việc kế toán và các hoạt động phân tích mà trước đây chỉ được làm bởi con người. Việc điều khiển cấu hình và phân tích sự đảm bảo chất lượng đã được tự động hoá rộng rãi và lĩnh vực tiếp theo là tự động hoá sản xuất và tự động hoá kiểm thử.

7. Chỉ 15% công sức phát triển phần mềm là dành cho lập trình.

Trong 10 năm trước đây, người ta không còn chú trọng việc đầu tư vào các ngôn ngữ và trình dịch, trừ Java và Ada 95. Những sự đầu tư công nghệ mới đã chuyển vào các sự hoàn thiện tiến trình (ví dụ như SEI CMM), mô hình hoá trực quan (như UML), chất lượng phần mềm được tự động hoá (vd sự tự động hoá kiểm thử), các thành phần (như ActiveX, Java, CORBA), quản

lý cấu hình, phép đo, và các khía cạnh khác của công nghệ phần mềm. Lượng công việclập trình có trong một dự án phát triển phần mềm chỉ còn khoảng 15 %. Sự khác nhau là các dự án hiện đại đang lập trình với cấp trìu tượng lớn hơn nhiều. Một tháng với một bộ phận trung bình của việc lập trình vào những năm 1960 cung cấp khoảng 200 lệnh máy, 1000 lệnh máy vào những năm 70 và 80. Hiệu năng của người lập trình những năm 1990 có thể cung cấp 10 nghìn lệnh máy trong một tháng riêng lẻ, mặc dù chỉ khoảng vài trăm dòng mã nguồn do con người viết ra được sản xuất.

8. Các hệ thống phần mềm và các sản phẩm trọn gói thường tốn gấp 3 lần nhiều hơn cho mỗi SLOC so với các chương trình phần mềm riêng rẽ. Một hệ thống các hệ thống phần mềm hay một sản phẩm phần mềm hệ thống tốn gấp 9 lần.

Sự không kinh tế về qui mô có thể được giảm nhẹ rất nhiều với một tiến trình mới hoặc các kỹ thuật mới. Trong các hoàn cảnh cụ thể – ví dụ như một dòng phần mềm của công việc sản xuất rời rạc, các hệ thống chuyên biệt theo đặt hàng với một kiến trúc tự chọn, môi trường tự chọn, và tiến trình tự chọn - một sự tiết kiệm qui mô là có thể đạt được.

9. Việc duyệt qua có thể bắt được 60% lỗi.

Tôi nhấn mạnh rằng sự cần thiết của tiêu chuẩn này bị biến mất trong top 10. Sự duyệt của con người và duyệt qua sẽ không làm lộ ra các vấn đề tới hạn, nó chỉ giúp giải quyết vấn đề. Tiêu chuẩn này có thể thay thế như sau: Khi môi trường bắt hầu hết những sự không ổn định cấp 1 và lỗi, các vấn đề thật sự quan trọng về cấu trúc chỉ có thể bị chỉ ra bằng cách mô tả ,sớm kiểm thử và giải quyết lại dưới sự khảo sát chăm chú cảa con người.

10. *80% sự đóng góp là đến từ 20% những người cộng tác.*

Mối quan hệ là không bị ảnh hưởng theo thời gian và nó tạo nên triết lý nền được ứng dụng trong mọi quá trình lập kế hoạch và chỉ đạo một tiến trình quản lý phần mềm hiện đại.

Chương 17
Sự quá độ sang xử lí hiện đại

Quản lí phần mềm thành công là một công việc hết sưc khó khăn.Những cuộc đột phá kĩ thuật,đột phá xử lí,và các công cụ mới sẽ làm cho nó trở nên dễ dàng hơn,nhưng các qui tắc quản lí vẫn là điểm then chốt quyết định sự thành công của dự ánphần mềm .

Điểm chính:

σ Quá độ sang xử lí phần mềm hiện đại và sử dụng các kĩ thuật mới cần rất nhiều các sự thay đổi ở khía cạnh văn hoá mà không phải lúc nà cũng dễ dàng đạt được .

σ Các bài học trong các tổ chức đang thực hiện sự chuyển sang xử lí hiện đại đã phơi bày rất nhiều chủ đề được tái diễn dẫn đến thành công mà thể hiện những sự thay đổi văn hoá quan trọng so với việc thực hành truyền thống

σMột sự thay đổi đáng kể có thể được thực hiện trên một dự án có ý nghĩa .Trong các dự án thử nghiệm không nên công kích nhữn người xuất sắc ,và hhọ rất quan trọng đối với thành công của bất kì sự thay đổi có ý nghĩa nào.

Những tiến bộ kĩ thuật mới kéo theo nhiều cơ hội cho các phần mềm ứng dụng, chiều huớng mới của sự phức tạp, nhận thức mới của sự tự động hoá, những khách hàng mới cùng với các quyền ưu tiên khác nhau .Việc điều tiết những sự thay đổi này sẽ làm xáo trộn rất nhiều giá trị và quyền ưu tiên của phong cách quản lí phần mềm quen thuộc .Tuy nhiên, tác động vào sự cân bằng giữa yêu cầu, thiết kế ,và việc lập kế hoạch sẽ vẫn còn là những đối tượng cơ bản của những nỗ lực quản lí phần mềm trong tương lai,kể cả bây giờ.

Cơ cấu quản lí phần mềm mà tôi sẽ trình bày trong cuốn sách này không mang tính chất cách mạng, vô số các dự án được thực hiện dưới những qui tắc này trong những năm gần đây.Tuy nhiên, rất nhiều kĩ thuật và qui tắc đuợc nêu trong này sẽ đòi hỏi một sự thay đổi mô hình đáng kể.Vài trong số những thay đổi này sẽ bị hạn chế bởi một số cổ đông nhất định hoặc bởi những nhân tố nhất định trong phạm vi của một dự án hay một tổ chức. Không phải luôn luôn dễ dàng phân tách những cản trở văn hoá từ những cản trở khách quan. Chương này sẽ tổng kết một vài sự thay đổi văn hoá quan trọng để chuẩn bị cho việc ngăn chặn nhiều nhất có thể sự bất hoà trong việc chuyển sang thành công quá trình xử lí hiện đại.

17.1.Sự quá độ xét ở khía cạnh văn hoá

Rát nhiều sự thay đổi văn hoá phải được khắc phục để chuyển sang thành công quá trình xử lí phần mềm.Vì một vài sự điều chỉnh , sẽ rất khó khăn để phân biệt giữa những cản trở mới xuất hiện và những vấn đề cũ vẫn chưa được sữa chữa. Tuy vậy , có rất nhiều chỉ dẫn chung cho một sự chuyển tiếp thành công tới một nền văn hoá hiện đại. Phần này sẽ thảo luận rất nhiều về những chỉ dẫn thô để tìm kiếm nhằm mục đích phân biệt các dự án đã thực hiện một sự thay đổi văn hoá với các dự án mà chỉ đưa ra sự thay đổi bề ngoài. Rất nhiều chỉ dẫn được suy dẫn một cách trực tiếp từ trật tự , cơ cấu xử lí được mô tả trong những chương trước, những chỉ dẫn còn lại là những hiệu ứng thứ hai.

- **Những nhà quản lí mức thấp và mức trung bình là người thực hiện:**

Có lẽ không có những nhà quản lí thuần tuý trong một tổ chức hay một tổ chức con với không quá 25 người . Nhu cầu về những người quản lí thuần tuý chỉ tăng lên khi nguồn nhân lực vượt quá mức này. Kĩ năng quản lí thực hành thay đổi, nhưng các nhà quản lí thành thạo điển hình dành phần lớn thời gian của họ vào việc thực hiện ,đặc biệt là việc hiểu rõ dự án ngay lần đầu tiếp xúc và phát triển kế hoạch , đánh giá .Trên tất cả , người điều hành nên lập kế hoạch . Điều này không có nghĩa là người quản lí nên tán thành kế hoạch ; mà có nghĩa là ngườii quản lí nên tham gia vào việc phát triển nó.Trong việc đánh giá những dự án độc lập mà tôi đã từng thực hiện , một người chỉ đạo tốt là một nhà quản lí không phải là tác giả mà cũngkhông là người sở hữu nó. Các cổ đông bị tác động bởi chính quá trình chuyển tiếp này và bởi các nhà quản lí phần mềm .

- **Yêu cầu và cách thiết kế là mềm dẻo và không trừu tượng ,gắn bó với thực tiễn .**

Cách xử lí truyền thống tập trung quá nhiều vào cung cấp tài liệu mô tả sản phần mềm và tập trung quá ít vào việc đưa ra sự lớn lên,sự tiến bộ trông thấy của chính bản thân sản phẩm. Những cột mốc lớn được định nghĩa một cách duy nhất theo các tài liệu cụ thể. Những tổ chức phát triển những dự án theo hợp đồng lớn buộc phải cung cấp hàng tấn giấy để chỉ ra các cột mốc và nhận tiền thanh toán tăng dần lên , hơn là tiêu phí năng lượng của họ vào các công việc làm giảm rủi ro và cung cấp những phần mềm chất lượng . Một qui trình lặp lại yêu cầu xây dựng thực sự một dãy các hệ thống xử lí càng ngày càng dễ hiểu,dể nắm bắt hơn để thể hiện kiến trúc ,giúp cho các cuộc thương lượng khách quan về yêu cầu ,làm hợp lí việc tiếp cận kĩ thuật ,và hướng sự chú ý vào việc tìm ra các giải pháp giải quyết các rủi ro .Một cách lí tưởng,tất cả các cổ đông nên tập trung vào những cột mốc có thực này ,với phân phối lợi nhuận của chức năng có ích hơn là các mô tả trên giấy của sản phẩm .Sư chuyển sang môi trường ít sử dụng giấy tờ sẽ được chấp nhận bởi đội ngũ các kĩ sư,tuy nó có khả năng bị hạn chế bởi việc giám sát hợp đồng theo phong cách truyền thống.

- **Sự thể hiện đầy tham vọng được khuyến khích .**

Mục đích của cách thể hiện vòng đời trước đây là vạch trần nhược điểm của thiết kế , không phải là đưa ra khía cạnh bề ngoài của nó . Các cổ đông không được phản ứng quá dữ dội với các lỗi ban đầu ,với sự lạc đề,hay là với thiết kế chưa hoàn chỉnh.Phê bình ,đánh giá trong những lần phát hành đầu tiên là mục tiêu chứ không phải là yêu cầu .Nếu những chướng

ngại kĩ thuật bị nhấn mạnh ,các tổ chức phát triển sẽ bắt đầu thiết lập sự lặp lại mà có sự suy giảm tham vọng trong những lần tiếp theo .Mặt khác , các cổ đông không nên chịu đựng sự thiếu hụt của công đoạn cuối trong các vấn đề đang phải giải quyết .Nếu xu hướng tiêu cực không được chú ý mạnh mẽ , nó có thể gây ra chiều hướng lo sợ nguy hiểm .Quan tâm và chú ý đến các công đoạn cuối là rất quan trọng trong việc giải quyết các vấn đề .Đội ngũ các nhà quản lí rất có thể ngăn cản sự quá độ này (đặc biệt là khi mà dự án được bán),bởi vì nó sẽ vạch ra bất cứ vấn đề kĩ thuật hay xử lí mà thật dễ dàng che dấu việc sử dụng cách xủ lí truyền thống .Khách hàng ,người dùng và các nhà kĩ thuật sẽ chấp nhận sự quá độ này cho cùng một lí do .

• Việc thực hiện các dự án tốt hay xấu được thấy rõ ràng hơn ngay từ lúc ban đầu của vòng đời .

Trong một sự phát triển lặp lại ,thành công nối tiếp thành công ,và các thất bại ban đầu là cực kì rủi ro để chuyển hướng ngược lại .Kinh nghiệm trong cac dự án thế giới thực rất nhiều lần chỉ ra rằng những pha ban đầu quyết định sự thành công hay thất bại của dự án Do đó việc chắc chắn rằng những pha ban đầu(pha lập kế hoạch và pha kiến trúc)là do đội ngũ tốt nhất thực hiện là tối quan trọng .Nếu những pha này được làm tốt với đội ngũ giỏi ,dự án có thể được thực hiện thành công bởi một đội ngũ chỉ ở trình độ trung bình phát triển ứng dụng thành sản phẩm cuối cùng .Nếu pha lập kế hoạch và pha kiến trúc không được thực hiện một cách chính xác ,tất cả các chuyên gia lập trình và các chuyên viên kiểm tra trên thế giới đều có khả năng không đạt được thành công Không người nào nên ngăn cản việc khởi đầu công việc với một đội ngũ tốt.Hơn nữa ,hầu hết các tổ chức đều có tài nguyên khan hiếm cho những vai trò của vòng đời lúc ban đầu và họ do dự trong việc thực hiện phân phối nhân lực cần thiết.

• Những lợi nhuận ban đầu sẽ không chắc chắn.

Những cổ đông bên ngoài , chẳng hạn như khách hàng và người sử dụng ,không thể hi vọng những sự phân phối khởi đầu thực hiện tới mức chi tiết được hoàn thành , được tin cậy hoàn toàn , hay đạt chất lượng cao nhất , được thực hiện tốt nhất .Mặt khác , các tổ chức phát triển phải bị coi là chịu trách nhiệm ,phải chứng tỏ tiến bộ trông thấy bằng lãi suất.Xu hướng này sẽ đưa ra sự hội tụ đến chi tiết . Việc thay đổi , cố định , hay cập nhật định lượng khách quan sẽ chỉ ra chất lượng của việc xử lí ,và chỉ ra môi trường cho các hoạt động trong tương lai .Khách hàng và người dùng sẽ gặp khó khăn trong việc thừa nhận nhược điểm ở những lần phát hành ban đầu , mặc dù họ có thể bị gây ấn tượng bởi những lãi suất được đem lại sau này.Các nhà quản lí và đội ngũ phát triển sẽ chấp nhận sự chưa hoàn thiện như là một phần tự nhiên của quá trình xử lí

•Mẫu(artifact) là không quan trọng ban đầu ,nhưng sẽ quan trọng về sau này .

Thật là lãng phí thời gian khi cứ băn khoăn về chi tiết (khả năng truy nguyên , sự tỉ mỉ , sự hoàn thiện) của tập hợp các mẫu cho đến giới hạn cuối cùng đạt được .Đủ hữu ích và chắc chắn để biện hộ cho những phân tích tốn thời gian cho những nhân tố về mặt chất lượng này Trái lại ,một dự án sẽ phung phí những chu trình công nghệ ban đầu và các tài nguyên quí giá thêm vào nội dung và chất lượng các mẫu mà có thể nhanh chóngtrở thành lỗi thời Trong khi

các nhà phát triển sẽ chấp nhận sự thay đổi này một cách hoàn toàn, việc giám sát hợp đồng truyền thống sẽ hạn chế sự không nhấn mạnh vào tính hoàn chỉnh ban đầu.

•Các vấn đề thực được trải rộng ra và giải quyết một cách hệ thống hoá

Những dự án thành công cho thấy yêu cầu và cách thiết kế phát triển song song với nhau ,bởi các cuộc thương lượng liên tiếp , mua bán , đổi chác hướng tới giá trị tốt nhất ,hơn là bám chặt một cách mù quáng vào bản hợp đồng lưỡng nghĩa , mơ hồ .Trong một dự án về sức khoẻ đang tiến triển ,có vẻ dễ dàng phân biệt giữa các vấn đề thực và các vấn đề rõ ràng,hiển nhiên.Tuỳ thuộc vào tình huống này ,sự thay đổi văn hoá có thể tác động vào bất cứ đội ngũ nào .

- **Sự chắc chắn về mặt chất lượng là công việc của tất cả mọi người ,không phải là công việc riêng .**

Rất nhiều tổ chức có một nhóm tách ra gọi là để bảo hiểm chất lượng .Tôi nói chung chống lại khái niệm các hoạt động ,các đội ngũ ,hay các sản phẩm đảm bảo chất lượng một cách riêng rẽ như thế .Để bảo hiểm chất lượng nên liên kết tất cả mọi vai trò ,mọi hoạt động ,mọi sản phẩm .Sự bảo hiểm chất lượng đúng được đánh giá bởi sự tiến bộ trông thấy và dữ liệu khách quan chứ không phải bởi danh sách kiểm tra , các hội nghị ,các sự xem xét của con người .Các nhà điều hành dự án phần mềm hay các nhà thiết kế nên nghĩ rằng vai trò của sự chắc chắn rằng bảo hiểm chất lượng là sự tích hợp đặc biệt vào tiến trình .Việc giám sát theo kiểu truyền thống bằng một đội ngũ riêng rẽ các nhà kiểm tra được thay thế bởi đội ngũ làm việc tự giám sát của một tổ chức ,với sự xử lí chín chắn các đối tượng chung và sự khích lệ chung Các nhà quản lí truyền thống và các nhân viên sẽ hạn chế sự quá độ này .Các đội ngũ kĩ thuật sẽ chấp nhận nó .

- **Các vấn đề thực thi sẽ gia tăng trong giai đoạn ban đầu của vòng đời** .

Những vấn đề thực hiện ban đầu sẽ được trải rộng ra trên hầu hết các dự án thành công mà tôi biết .Những vấn đề này là bằng chứng cho sự thiết kế chưa hoàn hảo nhưng là quá trình xử lí chín chắn .Các cổ đông sẽ luôn luôn quan tâm đến vấn đề thực hiện ban đầu .Các nhà phát triển sẽ chấp nhận sự nhấn mạnh trong sự thực hiện ban đầu và khả năng ước lượng ,đánh giá sự cân bằng thực thi trong những lần phát hành sau .

- **Đầu tư vào tự động hoá là cần thiết .**

Bởi vì các dự án phát triển lặp đi lặp lại sự tự động trên phạm vi rộng, thật là quan trọng đầu tư vào môi trường lợi nhuận .Việc nhận được môi trường tích hợp cũng rất quan trọng đối với các cổ đông , nó cho phép tham gia có hiệu quả trong sự phát triển lặp lại.Ngoài ra ,tương tác với các tổ chức phát triển sẽ làm giảm đi sự trao đổi giấy tờ và nhiều thứ khác của qúa trình xử lí truyền thống .Những đầu tư này có thể bị phản đối bởi người quản lí tổ chức mà tập trung thái quá vào lợi nhuận ngắn hạn hay bởi các nhân viên dự án ,người ủng hộ của các dự án cá nhân hơn là các giải pháp toàn bộ mà phục vụ cho cả dự án lẫn mục đích của tổ chức .

- **Các tổ chức phần mềm tốt nên đem lại nhiều lợi nhuận hơn.**

Trong lĩnh vực thương mại phần mềm ,đây không phải là một vấn đề . Nhưng trong hầu hết phần mềm hợp đồng , đặc biệt là hợp đồng của chính phủ, nó lại được định nghĩa là một vấn đề . Như là một phần của bản chất chống đối của sự thu thập và xử lí theo hợp đồng , có rất nhiều tiêu điểm có thể xem xét trong việc bảo đảm rằng lợi nhuận của nhà thầu là trong phạm vi một mức có thể chấp nhận được nhất định (điển hình từ 5% đến 15%).Thường thườhg một công ti thầu xuất sắc,một kết quả kĩ thuật tốt, hay kết quả sử dụng lại đáng chú ý trong lãi suất tiềm tàng trong sự quá mức trong việc trả giá ban đầu có thể chấp nhận được.. Ngay khi một khách hàng (hoặc người dùng ,hoặc các giám sát kĩ thuật) nhận biết được xu hướng như thế ,thật không thể tin được các áp lực đáng kể có thể được sử dụng để đưa ra những tài nguyên quá mứcđến các thay đổi vượt quá phạm vi cho đến khi sự chênh lệch giữa giá thành và giá bán trở lại mức có thể chấp nhận được

Kết quả là ,động cơ lợi nhuận đơn giản nhất tạo nên cơ sở của giao dịch thương mại và khích lệ hiệu quả được thay thế bởi sự khích lệ bằng hợp đồng phức tạp (và các mâu thuẫn giữa người tiêu dùng và người cung cấp) mà thường xuyên chưa được tối ưu.Rất thường xuyên ,nhà thầu không thấy sự khích lệ kinh tế nào khi thực hiện sự giảm giá ,và tất nhiên ,sẽ có rất ít sự khích lệ khi chấp nhận rủi ro để có thể mang lại nhiềutiền lãi .Mặt khác ,các nhà thầu có thể dễ dàng bỏ ra một khoản tiền lớn thường thường tại các điểm lãi suất sít sao.

Để thành công trong công nghiệp phần mềm , nhà thầu giỏi nên được trao thưởng (tăng lợi nhuận),nhà thầu kém nên bị phạt bằng cách giảm lợi nhuận .Khách hàng nhận được một sản phẩm tốt với một giá hợp lí nên hạnh phúc rằng nhà thầu đã làm một công việc tốt.Cho các nhà thầu đã làm việc không tốt tiếp tục làm việc thì sẽ không tốt đối với tất cả mọi ngươì . Đây là nơi mà lĩnh vực thương mại không hiệu quả bằng lĩnh vực hợp đồng chính phủ .

17.2 Đoạn kết

Tóm tắt lại ,việc xử lí phần mềm truyền thống có đặc trưng như sau :

- Chuyển dịch tuần tự từ yêu cầu tới thiết kế để mã hoá nhằm mục đích kiểm tra .
- Đạt được 100% sự hoàn chỉnh của mỗi sản phẩm tại mỗi giai đoạn của vòng đời
- Xử lí tất cả các yêu cầu ,sản phẩm ,các thành phần ,v...v
- Đạt được khả năng lần theo dấu vết với độ trung thực cao giữa các sản phẩm tại mỗi giai đoạn trong vòng đời

Một cơ cấu xử lí phát triển lặp lại hiện đại có đặc trưng như sau

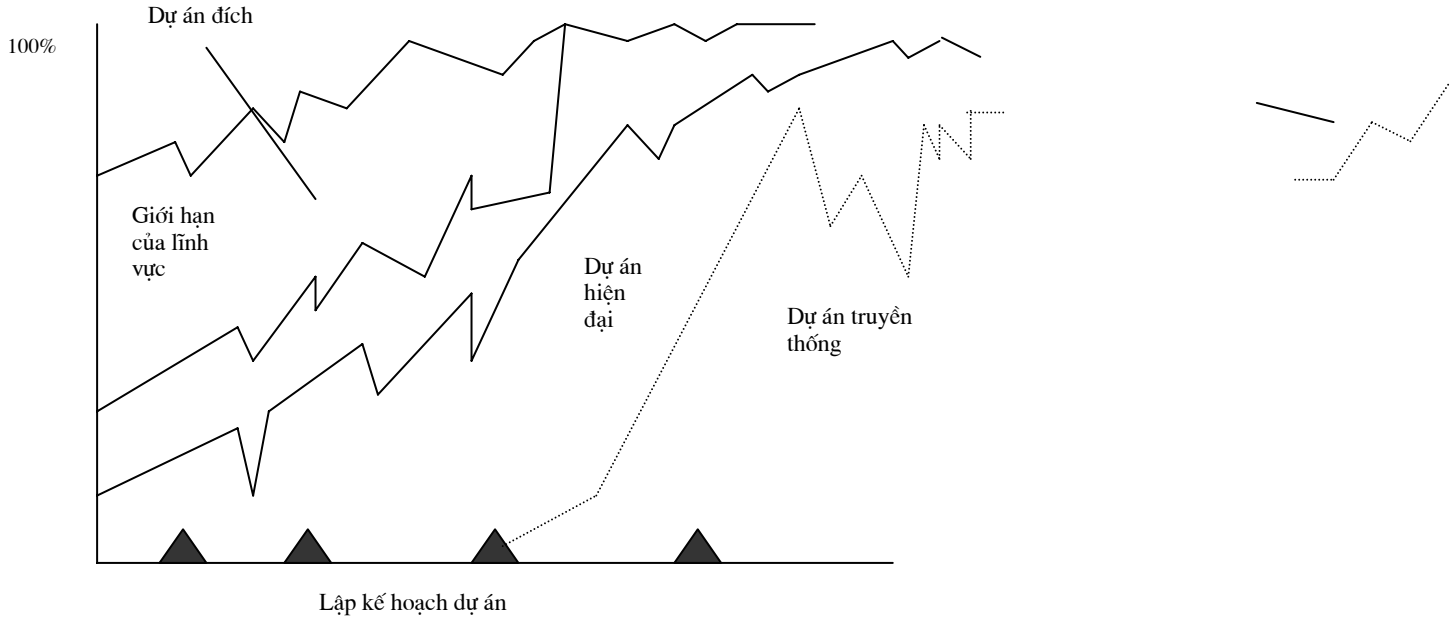
- Công nghệ quay vòng liên tục từ yêu cầu để kiểm tra tại bất cứ mức phát triển nào của dự án .
- Đạt được sự hiểu biết với độ trung thực cao của bộ diều khiển (20%) ngay từ khi mới thực hành ban đầu.
- Phát triển mẫu theo chiều rộng và chiều sâu dựa trên quyền quản lí mạo hiểm
- Hoãn xem xét sự hoàn chỉnh và sự phân tích nhất quán cho đến cuối vòng đời

Một cơ cấu xử lí hiện đại tấn công các nguồn ban đầu của sự cân bằng ------

cố hữu trong phong cách xử lí phần mềm truyền thống trong việc xử lí phần mềm truyền thống .

Hình 17.1minh họa thế hệ tiếp theo của việc thực hiện dự án phần mềm bằng việc thể hiện tiến độ phát triển theo thời gian , khi tiến độ thực hiện công việc được tính theo phần trăm (được trông thấy ở trục tung).

Mục đích của tôi trong cuốn sách này là giải thích cách di chuyển lên vùng tối phía trên với một quá trình xử lí hiện đại được hỗ trợ bởi môi trường tiến bộ ,tích hợp đầy đủ và một kiến trúc dựa trên các thành phần .Các tổ chức mà thành công có khả năng triển khai việc sản xuất phần mềm mà được xây dựng rộng rãi từ các thành phần đã tồn tại với thời gian ít hơn 50%,với tài nguyên nhiều hơn 50%,và được duy trì bởi đội ngũ chỉ bằng một nữa những đội ngũ được yêu cầu bởi hệ thống ngày nay



Hình 17-1. Thực hiện dự án ở thế hệ tiếp theo

Khi một tổ chức chuyển sang sử dụng công nghệ ,kĩ thuật mới ,luôn có sự lo lắng ,băn khoăn về việc có thể gặp thất bại trong tương lai.Duy trì trạng thái quo và tin tưởng vào các phương pháp đang tồn tại thường được coi là con đường an toàn nhất .Trong công nghiệp phần

mềm ,khi hầu hết các tổ chức thành công chỉ trong một số phần trăm nhỏ của các dự án của họ ,dyu trì trạng thí quo không phải luôn an toàn .Khi một tổ chức quyết định thực hiện một sự thay đổi ,hai phần của sự suy xét truyền thống thường được đề nghị bởi các nhà vô địch bên trong cũng như là các tác nhân thay đổi bên ngoài

Mở đầu sử dụng bất cứ kĩ thuật mới nào trong một chương trình thí điểm nhỏ

Chuẩn bị sử dụng một lượng lớn hơn tài nguyên ,tiền bạc và thời gian trên dự án đầu tiên thực hiện sự thay đổi của bạn .

Tôi thấy hai đề nghị này như là phản tác dụng

Các chương trình thử ngiệm không thuộc xu hướng chủ đạo có chổ đứng của nó ,nhưng nó hiếm khi đạt được bất cứ sự thay đổi quan trọng nào đến kết quả .Thử một kĩ thuật,công cụ hoặc phương pháp mới ,nhỏ trên một dự án nhỏ ,thời gian thực hiện nhanh (chưa đến ba tháng), và chỉ một ít người có thể đưa ra kết quả tốt , đà trước tiên , bằng chứng của khái niệm . Vấn đề với các dự án thử nghiệm đó là chúng hầu như không bao giờ bị phê bình với tổ chức .Kết quả là những dự án đó không xứng đáng với những người thực hiện loại A, với những tài nguyên thích đáng ,với sự quan tâm của tổ chức .

Những sự thay đổi tối quan trọng của các tổ chức thành công nhất tôi đã từng thấy có hoàn cảnh tương tự những nét sau : Các tổ chức đưa các dự án bị chỉ trích nhiều nhất cho các nhân viên có năng lực , phẩm chất cao nhất ,đưa cho họ những tài nguyên thích đáng ,và yêu cầu kết quả tốt hơn .Nếu ,mặt khác ,một tổ chức mong đợi một phương pháp mới ,một công cụ mới ,hoặc công nghệ mới để có một tác động bất lợi trên kết quả của dự án tiên phong ,mà sự mong đợi là hầu như trở thành sự thật .Tại sao? Bởi vì không có nhà quản lí tổ chức nào chủ tâm gây ra một tác động bất lợi trên các dự án quan trọng nhất ,và đó là nơi mà những nhân viên tốt nhất của tổ chức được ấn định Do đó ,các dự án tiên phong sẽ là các dự án không bị phê phán được thực hiện bởi các nhân viên không bị phê phán,bị mong đợi quá nhiều .Sự mong đợi ít này thường là được hoàn thành .

Một cách tốt hơn để để chuyển sang việc xử lí lặp lại chín chắn hơn mà hổ trợ kĩ thuật tự động hoá và các kiến trúc hiện đại là thực hiện những bước sau

- Sẵn sàng . Làm bài tập của bạn . Phân tích các cách tiếp cận hiện đại ,và các kĩ thuật hiện đại .Xác định (hoặc là phát triển ,tối ưu) quá trình xử lí của bạn .Hổ trợ nó với môi trường hoàn thiện , các công cụ ,các thành phần mới .Lập kế hoạch một cách cẩn thận
- Mục đích .Lựa chọn một dự án bị chỉ trích .Phân bố nhân lực tốt cho nó ,cung cấp tài nguyên đầy đủ và đòi hỏi kết quả tiến bộ hơn.
- Gây hứng thú .Xử lí việc lập kế hoạch tổ chức và lập kế hoạch cho các giai đoạn của dự án với sự nổ lực lớn và làm đến cùng .Giải quyết nó với những tài nguyên bổ sung nhau và yêu cầu nâng cao kết quả .

Phụ lục A thực trạng trong quản lí phần mềm.

Phụ lục B mô hình ước lượnG giá Cocomo.

Phụ lục C độ đo ngẫu nhiên.

Phụ lục D nghiên cứu dự án ccpds-r

Phụ lục E tiến trình xử lí và ánh xạ tới cmm.

Các tư liệu đã trình bày trong bốn phần đầu của cuốn sách này dựa trên sự nỗ lực của nhiều người , các dự án và các tổ chức. Phần lớn các đánh giá và giới thiệu của tác giả được dựa trên các bài học rút ra từ việc ứng dụng các dự án. Phần này bao gồm các phụ lục chi tiết ,cung cấp các bối cảnh lịch sử và các giải thích chi tiết các thực tế có liên quan.

- Phụ lục A tóm lược ba bối cảnh khác nhau của tình hình quản lí phần mềm trong thực tế.
- Phụ lục B tóm tắt sự phát triển của mô hình COCOMO từ khởi đầu đến phiên bản hiện tại là COCOMO II. Chủ đề này cung cấp phạm vi các quan điểm của tác giả về sự phát triển của nền kinh tế phần mềm.
- Phụ lục cung cấp chi tiết hơn về nguồn gốc và các nhân tố cơ bản sau độ đo ngẫu nhiên đã trình bày ở chương 13.
- Phụ lục D là sự nghiên cứu tỉ mỉ trên đối tượng cụ thể, một dụ án phức tạp qui mô lớn đã thành công.Đó là nguồn ví dụ phong phú có thể cung cấp như là chuẩn thực hiện cho các dự án khác.Trong khi bất kì sự nghiên cứu trên một dự án cụ thể sẽ biểu diễn các chủ đề mà chỉ liên quan tới nhóm hẹp, thì rất nhều chủ đề cung cấp một cách toàn diện các kết quả khá thú vị. Sự mở rộng , sự tiếp cận và kết quả của dự án này là mô hình của một tiến trình phần mềm hoàn thiện và được quản lí tốt.
- Phụ lục E định giá tiến trình đã biểu diễn trong cuốn sách này sử dụng bảng câu hỏi điều tra theo kì hạn của viện công nghệ phần mềm. Nó cung cấp một sự miêu tả sơ lược về sự trưởng thành, tính vững chắc và tính toàn vẹn của một cơ cấu tổ chức tiến trình tương phản với một sự định giá tiến trình chuẩn đã được thừa nhận.

Phụ lục A

Thực trạng trong quản lí phần mềm

Những nội dung chính :

- Việc quản lí công nghiệp phần mềm thực hiện trong những năm 1990 vẫn còn phản ánh các tiến trình chưa chín muồi với đặc điểm phân mảnh và lặp lại quá mức.
- Khoảng 10% các dự án truyền thống là thành công ,trong đó sự thành công được định nghĩa là sự thoã mãn sự mong đợi của khách hàng về giá cả, thời gian, chất lượng và tập hợp các đặc trưng đặc biệt và tạo ra lợi nhuận.
- Những nhà quản lí phần mềm là những người đầu tiên nhận ra sự thành công và thất bại của dự án.

Ba phân tích quan trọng được thực hiện vào giữa những năm 1990 đã chỉ rõ tình trạng của nền công nghiệp công nghệ phần mềm. Chúng đưa ra kết luận là tỉ lệ thành công của các dự án phần mềm là rất thấp. Phụ lục này sẽ tóm tắt các kết quả của các phân tích đó .

Mô hình của hệ thống phần mềm thât bại và thành công

Cuốn sách này [Jones,1996] là sự trình bày đầy đủ tình trạng của nền công nghiệp phần mềm. Tác giả đã phân tích kết quả hàng nghìn dự án được chia thành 6 nghành công nghiệp nhỏ: phần mềm hệ thống, hệ thống thông tin, phần mềm thương mại , phần mềm không nguồn, phần mềm quân sự, phần mềm cho người dùng thuần tuý. Bảng A-1 tóm tắt toàn bộ nguyên nhân căn bản của sự thất bai và thành công của các dự án phần mềm.

Tác giả đã đưa ra một nhận xét thú vị về bảng này là:

> *Điều thú vị và có ý nghĩa là sáu trong số mười sáu yếu tố kĩ thuật đầu tiên có liên quan tới các thảm hoạ phần mềm là các thất bại cụ thể trong lĩnh vực quản lí dự án , và ba trong số những sự thiếu sót về công nghệ có thể được gán trực tiếp cho thực tế quản lí nghèo nàn.*

Tác giả cũng đã nhận ra các yếu tố văn hoá và con người phân biệt sự giữa các dự án thành công và thất bại. Những yếu tố này được trình bày trong bảng A-2

Mô hình của sự thành công và thất bại được ước lượng từ nhiều bối cảnh khác nhau. Sự khác nhau giữa 6 ngành công nghiệp nhỏ và tỉ lệ khác nhau của các dự án được tác giả mô tả một cách chi tiết. Một thông điệp nổi bật chung cho các nhân tố thông qua tất cả các lĩnh vực được tác giả đưa ra : *" Tôi tán thành với hầu hết các kế luận được đưa ra trong hai bảng trên nhưng tôi phản đối việc so sánh tầm quan trọng giữa các yếu tố khác nhau và việc thực hiện các chi tiết có liên quan việc ứng dụng thành công một số công nghệ.Ví dụ ba yếu tố đầu trong bảng A-1 có thể là những đặc điểm chung nhất song tôi không nghĩ là chúng sự phân biệt quan trọng nhất giữa sự thất bại và thành công.Quan điểm của tôi được trình bày ở chương 4."*

Bảng A-1 Các *công nghệ đã sự dụng trong các dự án phần mềm*

Các công nghệ trong các dự án không thành công.	Các công nghệ trong các dự án thành công.
Không có dữ liệu phép đo phần mềm gốc	Phép đo phần mềm chính xác.
Không sử dụng công cụ ước lượng tự động	Sử dụng công cụ ước lượng từ đầu.
Không sử dụng công cụ lập kế hoạch tự động	Liên tục sử dụng các công cụ lập kế hoạch
Không giám sát sự tiến triển và các mốc quan trọng.	Đều đặn kiểm tra sự tiến triển.
Không sử dụng kiểu kiến trúc có hiệu quả .	Đều đặn dự tính các kiến trúc.
Không sử dụng các cách phát triển có hiệu quả.	Phương pháp phát triển hợp thức.
Không xem xét lại thiết kế.	Đều đặn xem xét lại thiết kế.
Không sử dụng sự kiểm tra điều luật	Đều đặn kiểm tra điều lệ.
Không tính đến sự quảnlí rủi ro thông thường .	Quản lí các rủi ro thông thường .
Kiểm tra không đầy đủ và đúng qui định	Phương pháp kiểm tra theo đúng qui định.
Thiết kế và đặc tả bằng tay.	Thiết kế và đặc tả tự động.
Không sử dụng sự điều khiển cấu hình hợp thức	Tự động điều khiển cấu hình .
Hơn 30% tăng dần theo nhu cầu của người sử dụng	Hơn 10% tăng dần theo nhu cầu của người sử dụng.
Không tương thích với việc sử dụng 4 GLs	Sử dụng ngôn ngữ thích hợp.
Qúa phức tạp và không thể đo được độ phức tạp.	Sự phức tạp điều khiển và đo được.
Rất ít hoặc không có sự dùng lại các tài liệu đã được kiểm chứng .	Dùng lại có hiệu quả các tài liệu đã được kiểm chứng.
Không định nghĩa các thành phần cơ sở dữ liệu	Đều đặn dự tính các cơ sơ dữ liệu.

Bảng a-2 *Các yếu tố xã hội đực nhận xét từ các dự án*

Các dự án không thành công	Các dự án thành công
Lập kế hoạch quá vội vã	Sự dư tính kế hoạch làm việc mang tính thực tế
Bỏ qua việc ước lượng thực tế	Sự am hiểu về ước lượng có tính thực tế
Xung đột gay gắt với khách hàng	Cộng tác tốt với khách hàng
Chính sách tập thể gây chia rẽ	Mục tiêu quản lý phù hợp
Sự giao tiếp hợp tác nghèo nàn	Sự giao tiếp hợp tác xuất sắc
Các ỷu viên ban quản trị chưa có kinh nghiệm	Các uỷ viên ban quản trị có kinh nghiệm
Ban uản lý dự án lạm dụng công quĩ	Ban quản lý dự án có năng lực
Các nhân viên kỹ thuật không đủ năng lực	Các nhân viên kỹ thuật có năng lực
Không có các chuyên viên sử dụng cho nhiệm vụ đánh giá: tính hoàn thiện, tính chắc chắn, việc kiểm tra, lập kế hoạch và định giá.	Có các chuyên viên được sử dụng cho nhiệm vụ đánh giá: tính hoàn thiện, tính chắc chắn, việc kiểm tra, lập kế hoạch và định giá.

Bản báo cáo "CHAO"[Standish Group,1995]

"CHAO" là một báo cáo tập trung về ngành công nghiệp phần mềm thương mại đã đưa ra những kết luận sau :

- Các công ty Mỹ đã phải chi phí 81 tỉ đô la cho các dự án phần mềm bị huỷ bỏ trong năm 1995.
- 31% các dự án phần mềm đã nghiên cứu bị huỷ bỏ trước khi chúng hoàn thành.
- 53% các dự án phần mềm bị huỷ bỏ hơn 50%.
- Chỉ 9% các dự án phần mềm cho các công ty lớn được giao đúng thời hạn và trọn gói, đối với các công ty vừa và nhỏ con số này tăng lên là 16% và 28%.

Bản báo các mô tả đặc điểm của 10 nguyên nhân thành công và 10 nguyên nhân làm cho các dự án gặp rủi ro ,quan trọng nhất (Các dự án gặp rủi ro được gọi là "challenged"). Các yếu tố này được tóm tắt trong bảng A-3. Hầu như toàn bộ báo cáo "Chào" có liên quan tới các vấn đề và các trở ngại mà các nhà quản lí hệ thống thông tin cấp công ti gặp phải. Mặc dù chỉ có một sự luận giải rất nhỏ cho các giải pháp có thể, bản báo cáo đã giới thiệu một phương pháp khắc phục là cách tiếp cận theo hướng xử lí cao hơn chứ không chỉ là giải quyết các vấn đề đã đưa ra.

Bản báo cáo kết luận:

Khung thời gian nhỏ cùng với các sự phát sinh các thành phần phần mềm sớm và thường xuyên sẽ làm tăng tỉ lệ thành công.Khung thời gian ngắn hơn sẽ dẫn tới một tiến trình lặp đi lặp lại của việc thiết kế, làm bản mẫu, phát triển, kiểm tra và sự khai triển thành các thành phần nhỏ. Tiến trình này được biết đến như là sự gia tăng phần mềm(growing softwear) khác với khái niệm cũ là phát triển phần mềm (developing software)

Sự gia tăng phần mềm(growing software) thu hút người sử dụng sớm hơn,mỗi thành phần có một chủ sở hữu hoặc một tập hợp nhỏ những chủ sở hữu, và những sự dự tính thực sự được đặt ra. Thêm vào đó mỗi thành phần phần mềm có một sự trình bày và tập hợp các đối tượng rõ ràng chính xác .Các thành phần mềm và các dự án nhỏ có xu hướng ít phức tạp hơn. Tạo ra các dự án đơn giản hơn là một nỗ lực đáng giá , bởi vì sự phức tạp chỉ tạo ra sự hỗn độn và làm tăng chi phí.

Bản báo cáo "Chaos" phản ánh sự tin tưởng nổi trội trong quản lí phần mềm là: lí do chính dẫn đến sự thất bại và thành công tập trung vào các tiến trình quản lí các yêu cầu.

Điều này hàm ý rằng từ các cấu trúc suy ra chúng xây dựng cái gì (yêu cầu) và chúng xây dựng như thế nào(tiến trình) không phải là một vấn đề lớn. Nhưng nó trái với sự thật là hoạt động quản lí các yêu cầu thường chỉ tiêu thụ khoảng 10% tài nguyên của một vòng đời (life-cycle), 90% còn lại cũng phải thực hiện thành công. Bởi vì hoạt động quản lí các yêu cầu chiếm ưu thế trong các vòng đời ban đầu nên chúng rất dễ bung xung. Tương phản với hàm ý đó ,trong bản báo cáo này sự giới thiệu về cách làm cho vấn đề nhỏ hơn là khá sâu sắc và thích hợp với một tiến trình lặp lại hiện đại.

Bảng A-*3 Những nhân tố ảnh hưởng tới sự thành công của các dự án phần mềm.*

Các dự án thành công	%	Các dự án rủi ro	%
Sự liên quan tới người sử dụng	15,9	Thiếu sự cung cấp của người sử dụng	*12,8*
Sự ủng hộ của người quản lí	*13,9*	Không hoàn thành các yêu cầu	*12,3*
Sự trình bày rõ các yêu cầu	*13,0*	Thay đổi các yêu cầu	*11,8*
Các kế hoạch đúng đắn	*9,6*	Thiếu sự hỗ trợ hành chính	*7,5*
Dự tính có tính thực tế	*8,2*	Thiếu trình độ kỹ thuật	*7,0*
Các mốc tính dự án nhỏ hơn	*7,7*	Thiếu tài nguyên	*6,4*
Các nhân viên tham gia	*7,2*	Dự tính không có thực	*5,9*
Quyền sở hữu	*5,3*	Các đối tượng không rõ ràng	*5,3*
Đối tượng và tầm nhìn rõ ràng	*2,9*	Khung thời gian không có thực	*4,3*
Các nhân viên chăm chỉ và đoàn kết	*2,4*	Kỹ thuật mới	*3,7*
Các yếu tố khác	*13,9*	Các yếu tố khác	*23,0*

Bản báo cáo của Defense Sience Board Task Force dựa trên kinh nghiệm của phần mền bảo vệ trên phương diện thương mại .

Bản báo cáo này trình bày các nội dung sau :

- Chuyên ngành bảo vệ thực tế không phù hợp với thực tế kinh doanh thương mại .
- Ban quản lí chương trình DOD hầu như không quan tâm tới thực tế thương mại.
- Có sự thiếu hụt các nhân viên phần mềm có đủ trình độ thích hợp tại tất cả các mức của DOD.
- DOD không nhận biết một cách đầy đủ về những lí luận tán thành và phản đối của việc sử dụng các thành phần thương mại .
- DOD không làm nổi bật được kiến trúc.
- DOD không xúc tiến một cách tương xứng việc trao đổi công nghệ với thị trường thương mại.

Bản báo cáo phát biểu rằng mặc dù DOD đã thực hiện nhiều nhiều nghiên cứu về các dự án phần mềm (18 dự án được liệt kê), nhưng phần lớn các đề nghị đưa ra từ những nghiên cứu này lại không được thực hiện đầy đủ .

Những nguyên nhân chính làm cho dự án phần mềm DOD rơi vào tình trạng rối loạn được nhận biết như sau:

- Sự xác định yêu cầu quá nghèo nàn.
- Sự quản lí tiến trình phần mềm không đầy đủ .
- Thiếu các nhóm sản phẩm tích hợp.
- Sự quản lí các thầu phụ không có hiệu quả.
- Thiếu sự chú ý thích hợp tới các tiến trình.
- Quá ít sự quan tâm tới các kiến trúc phần mềm.
- Định nghĩa một các nghèo nàn không thích đáng các giao diện điều khiển.
- Sự nâng cấp phần mềm tạo ra sự thiếu phần cứng.
- Tập trung vào sự đổi mới hơn giá cả và sự rủi ro.
- Bị giới hạn bởi tiêu chuẩn quân sự.

Những đề nghị chính được đưa ra là :

- Khai thác các thực tế thương mại (ví dụ sự phát triển lặp lại và các tiến trình kiến trúc ban đầu).
- Khai thác các thành phần thương mại và các công nghệ.
- Đầu tư hơn nữa vào việc đào tạo phần mềm cho những người tham gia dự án DOD.

Bản báo cáo thảo luận cách giải quyết các rủi ro mà nó đưa ra . Nó không quá cường điệu sự cần thiết phải định nghĩa và điều khiển yêu cầu tốt hơn, như là các nghiên cứu trước của DOD đã làm . Chủ đề này được đề cập và thảo luận với sự nhấn mạnh thích hợp có cân nhắc với các yếu tố quan trọng khác.

Báo cáo "CHOAS" cho rằng phần lớn lỗi dẫn đến các dự án không thành công là sự thiếu hụt trong quản lí các yêu cầu. Chuyên ngành bảo vệ (The Department of Defens) có lẽ sẽ tán thành sau những năm 1980 tuy nhiên dường như đã hoàn thiện một sự tự đánh giá có cân nhắc hơn và sự am hiểu cả về biểu hiện và thực chất bên trong.

Mô hình ước lượng chi phí COCOMO

Các chủ điểm chính:

- *Lịch sử của mô hình COCOMO đưa ra việc thấu hiểu bên trong tới sự tiến triển của thứ tự ưu tiên của nền kinh tế phần mềm.*
- Mô hình COCOMO ban đầu đã khá phù hợp đối với sự ước lượng chi phí dự án phần mềm truyền thống trong thập niên 80.
- Mô hình Ada COCOMO đã cải tiến phiên bản đầu tiên, đặc biệt qua một tham số mũ mà đã phản ánhsự cải tiến tiến trình hiện đại và tác động của chúng vào nấc thang kinh tế.
- Mô hình COCOMO II đã phù hợp hơn đối với việc ước lượng các dự án phát triển phần mềm hiện đại. Nó cung cấp nhiều sự hỗ trợ tự nhiên hơn cho các tiến trình và công nghệ hiện đại và nhiều cập nhật cơ bản của kinh nghiệm dự án.

Ngày nay có vài mô hình ước lượng chi phí đang được sử dụng. Một mô hình ước lượng chi phí phần mềm có tính mở, thông dụng là mô hình chi phí có tính kết cấu (COCOMO) được phát triển bởi Barry Boehm. Sự ra đời của COCOMO cung cấp một khung nhìn thích hợp cho việc quan sát sự phát triển của kinh tế công nghệ phần mềm trong những năm qua.

Biểu thức ước lượng COCOMO được cho dưới dạng sau:

Công sức= C1 EAF (Kích Thước)^{P1}

Thời gian= C2 (Công sức)^{P2}

Với Công sức= số tháng làm việc thực sự

C1= hệ số tỷ lệ không đổi của công sức

EAF= nhân tố điều chỉnh công sức mà mang tính chất lĩnh vực, tính cá nhân, tính môi trường, và công cụ sử dụng để sản xuất các mẫu tiến trình

Kích thước = kích cỡ của sản phẩm cuối (trong mã nguồn do người phát sinh ra, đo bởi số chỉ thị nguồn phân phát cần thiết để phát triển những chức năng đòi hỏi.

P1= số mũ mà mang tính chất kinh tế của tỷ lệ cố hữu trong tiến trình sử dụng để sản xuất ra sản phẩm cuối, đặc biệt là khả năng của tiến trình tránh các thao tác cộng thêm không giá trị (làm lại, trễ nải do thủ tục hành chính, truyền thông bên trên)

Thời gian= tổng số tháng làm việc

C2= hệ số tỷ lệ hằng số cho việc lập lịch trình

P2= số mũ mà có đặc điểm mang tính trơ và tương đồng cố hữu trong quản lý công sức phát triển phần mềm.

B.1 COCOMO

Mô hình COCOMO khởi thuỷ [Boehm 1981] là một trong những đột phá của kỹ thuật phần mềm trong đầu những năm 1980. Nó chỉ là một đột phá thành phần bởi sự đóng góp công nghệ cố hữu nhưng cơ bản bởi nó cung cấp một cơ cấu tổ chức được định nghĩa tốt cho việc truyền thông của việc cân bằng các nhân tố và mức đặc quyền cùng với chi phí phần mềm và quản lý việc lên kế hoạch. Vì là một sinh viên sau đại học ở UCLA năm 1980, tôi đã được tiếp cận lần đầu với mô

hình COCOMO như một đề tài cho khoá học cho sinh viên sau đại học của Boehm. Cùng lúc đó, tôi đang làm việc tại TRW với vai trò một người thiết kế chính về kế hoạch mở rộng phần mềm cho những điều mà chúng tôi cần lên kế hoạch và bảo vệ chi phí phần mềm và ước lượng lên kế hoạch. Một ưu điểm nổi bật của của mô hình COCOMO là khả năng tạo nền tảng tham khảo, ước lượng cho nó, nguyên nhân về ưu và nhược điểm của nó, và thương lượng với người giữ chi phí đặt trước. Từ đó, tôi đã dùng COCOMO để hợp lý hoá khi bổ sung các kỹ thuật, phát triển các tiến trình, thay đổi kiến trúc của dự án và tiến tới cách quản lý mới. Trong những hoạt động này, tôi đã có kinh nghiệm với ưu và nhược điểm của nó, cũng như những khả năng trong sử dụng và không sử dụng của nó.

Mô hình COCOMO ban đầu dựa trên một cơ sở dữ liệu của 56 dự án. 3 chế độ của nó phản ánh sự khác nhau trong tiến trình thông qua một phạm vi các lĩnh vực phần mềm. Những chế đô này được xác định gồm có chế độ bộ phận, chế độ bán phụ thuộc và chế độ nhúng. Chế độ bộ phận có đặc điểm là nhỏ, những phát triển ít phức tạp mà có tiến trình linh động. Các tính năng, chất lượng, chi phí, và lập lịch có thể tính tự do với sự tối thiểu ở trên. Các hệ thống có chế độ nhúng biểu diễn các dự án cộng đồng bảo vệ thông thường: tính phức tạp, độ tin cậy và các vấn đề thời gian thực có tính trội hơn, và tính chất hợp đồng của kinh doanh có kết quả trong một tiến trình nghiêm khắc. Các tính năng, chi phí chất lượng, các kế hoạch được kiểm soát chặt chẽ và những thay đổi đòi hỏi sự đồng ý của những người giữ chi phí đặt trước. Còn chế độ bán phụ thuộc đại diện cho lớp trung gian.

Biểu thức công sức cơ bản và công thức ước lượng lập lịch trình

Có những mối liên hệ trong ước lượng chi phí COCOMO ban đầu:

Chế độ bộ phận: Công sức= 3.2 EAF (Kích thước)$^{1.05}$

Thời gian (theo tháng)= 2.5 (Công sức)$^{0.38}$

Chế độ bán phụ thuộc: Công sức= 3.0 EAF (Kích thước)$^{1.12}$

Thời gian(theo tháng)= 2.5 (Công sức)$^{0.35}$

Chế độ nhúng: Công sức= 2.8 EAF (Kích thước)$^{1.2}$

Thời gian(theo tháng)= 2.5 (Công sức)$^{0.32}$

với Công sức = số tháng làm việc

EAF = kết quả của 15 nhân tố điều chỉnh công sức (bảngB-1)

Kích thước=số câu lệnh nguồn được trả lại (theo đơn vị hàng nghìn dòng mã)

Bộ kết hợp nhân tố điều chỉnh công sức (EAF- Effort Adjustment Factor) biểu diễn cho các hiệu quả kết hợp của các tham số nhân. Những tham số này cho phép dự án được vạch đặc điểm và bình thương hoá đối lại các dự án trong cơ sở dữ liệu COCOMO. Mỗi tham số được định giá là rất thấp, thấp, bình thường, cao hoặc rất cao. Sự thiết lập ảnh hưởng của mỗi tham số là bộ nhân mà có phạm vi thông thường từ 0.5 đến 1.5. Kết quả của 15 ảnh hưởng này được sử dụng như hệ số trong biểu thức chi phí.

Bảng B-1 *Các tham số đặc tính dự án COCOMO*

Định danh	Nhân tố điều chỉnh công sức	Phạm vi tham số	Tác động tiềm tàng
RELY	Yêu cầu tin cậy	0.75-1.40	1.87
DATA	Cỡ cơ sở dữ liệu	0.94-1.16	1.23
CPLX	Mức phức tạp sản phẩm	0.70-1.65	2.36
TIME	Rằng buộc thời gian thực hiện	1.00-1.66	1.66
STOR	Rằng buộc bộ nhớ chính	1.00-1.56	1.56
VIRT	Tính thay đổi máy ảo	0.87-1.30	1.49
TURN	Thời gian xoay vòng máy tính	0.87-1.45	1.32
ACAP	Khả năng phân tích	1.46-0.71	2.06
AEXP	Kinh nghiệm ứng dụng	1.29-082	1.57
PCAP	Khả năng người lập trình	1.42-0.70	2.03
VEXP	Kinh nghiệm máy ảo	1.21-0.90	1.34
LEXP	Kinh nghiệm ngôn ngữ	1.14-0.95	1.20
MODP	Thực hành lập trình hiện đại	1.24-0.82	1.51
TOOL	Sử dụng công cụ phần mềm	1.24-083	1.49
SCED	Lập lịch phát triển yêu cầu	1.23-1.10	1.23

Giả thiết

Có vài giả thuyết mang tính cố hữu trong biểu thức COCOMO. Các chỉ thị nguồn phân phối bao gồm tất cả các dòng mã xử lý máy tính (không kể chú thích). Vòng đời phát triển bắt đầu tại điểm đầu của khâu thiết kế sản phẩm và kết thúc với sự kiểm tra chấp nhận tại khâu kết luận của pha phân tích và kiểm tra. (Công sức phân tích các yêu cầu và lập lịch được ước lượng độc lập như mức phần trăm thêm vào của ước lượng phát triển). Các hoạt động chỉ bao gồm công sức dự án tính trực tiếp, và loại ra các hoạt động bên ngoài thông thường như hỗ trợ hành chính, chức năng, và thiết bị tài chính. Một tháng làm việc bao gồm 152 giờ. Dự án sẽ được quản lý tốt. Dự án sẽ như những yêu cầu ổn định.

Mô tả vòng đời

Vòng đời COCOMO bao gồm 5 pha cơ bản: kế hoạch và yêu cầu, thiết kế sản phẩm, thiết kế chi tiết, mã hoá và kiểm tra đơn vị, kết luận và kiểm tra. COCOMO cung cấp các nhắc nhở cho việc phân bổ công sức và lên kế hoạch qua những pha cơ bản của mô hình thác nước thông thường. Những nhắc nhở này thay đổi với chế độ và tỷ lệ; bảng B-2 cung cấp phác thảo thông thường cho dự án theo chế độ nhúng và lớn. COCOMO ước lượng công sức và lập lịch để phát

triển giải pháp (sản phẩm thiết kế thông qua pha kết luận và kiểm tra). Việc làm thành công thức vấn đề (kế hoạch và yêu cầu) được ước lượng như phần trăm phụ thêm dựa trên công sức phát triển và lập lịch.

Cấu trúc chia nhỏ sản phẩm phần mềm (WBS)

Phần lớn các cấu trúc chia nhỏ sản phẩm thông thường được tổ chức quanh các hệ thống phụ của sản phẩm tại cấp cao hơn và quanh các hoạt động chi tiết tại cấp thấp hơn. Các hoạt động

Bảng B-2. *Sự phân chia công sức và lịch trình qua pha vòng đời truyền thống*

Hoạt động	Công sức (%)	Lịch trình (%)
Các kế hoạch và yêu cầu	(+8)	(+36)
Thiết kế sản phẩm	18	36
Thiết kế chi tiết	25	18
Lập trình và khối kiểm tra	26	18
Tích hợp và kiểm thử	31	28

Bảng 3-2. *Sự phân chia công sức và lịch trình qua pha vòng đời truyền thống*

Hoạt động	Ngân sách (%)
Phân tích các yêu cầu	(+36)
Thiết kế sản phẩm	36
Lập trình	18
Lập kế hoạch kiểm tra	18
Tích hợp và kiểm thử	28

chuẩn ước lượng bởi COCOMO và bao gồm trong phần mềm WBS là phân tích yêu cầu, thiết kế sản phẩm, lập trình, kế hoạch kiểm tra, việc thẩm tra và làm cho có hiệu lực, các chức năng văn phòng dự án (quản lý và kiểm tra), quản lý cấu hình, bảo đảm chất lượng và hướng dẫn. COCOMO cũng đồng thời nhắc nhở sự phân phối mức đỉnh của công sức qua các hoạt động của WBS. Trái lại, những sơ thảo này độc lập với chế độ và tỷ lệ. Bảng B-3 xác định hiện trạng sự tiêu dùng mong đợi trong các hoạt động trong WBS cho các dự án theo chế độ nhúng, lớn. Một khuyến cáo quan trọng là trong COCOMO hoạt động lập trình bao gồm thiết kế chi tiết,mã hoá, kiểm tra đơn vị và kết luận.

Sơ thảo dự án COCOMO thông thường

Việc thảo luận tiếp theo đây sẽ tập trung vào một dự án ví dụ cụ thể nhằm mục đích minh hoạ các liên quan lập kế hoạch cho dự án.Giả sử một hệ thống có chức năng phê bình

lớn,100.000 dòng mã nguồn (ví dụ kiểm soát của một kế hoạch lớn) xây dựng dưới hợp đồng cho một tổ chức chính phủ. Hình B-1 minh hoạ cho COCOMO ước lượng sơ thảo cho dự án. COCOMO sẽ ước lượng 900 tháng làm việc cho việc phát triển cộng với 72 tháng làm việc cho các đặc tả đòi hỏi cho dự án. Yêu cầu lập lịch sẽ là 22 tháng làm việc kể từ sự khởi tạo của thiết kế qua kiểm tra, cộng với 8 tháng cho các yêu cầu.

B.2 Mô hình Ada COCOMO

Vào giữa những năm 80, TRW giáp mặt với thách thức truyền 1 số dự án tới Ada. Trong một số trường hợp, những kỹ sư dự án tin rằng công nghệ Ada là giành trước cạnh tranh và hiệu suất thành công. (Những dự án này dường như sắp thành công). Tôi đã hình thành dự án vào năm 1985. Mục đích của công sức này là cung cấp một khung chuẩn cho việc quản lý TRW tin cậy và một khách hàng rằng lợi nhuận chi phí của việc sử dụng Ada trên dự án có phạm vi lớn xác định là quan trọng, và rằng áp dụng Ada trên dự án nàylà một lợi thế trong chiến lược cạnh tranh. nó cũng đồng thời phần mềm có ít sai sót nhất đạt được cho phân phối phần mềm trên kinh phí và trên lập lịch với chất lượng yêu cầu. (Dự án đó là CCPDS-R, hệ thống cảnh báo tên lửa thế hệ kế tiếp được giới thiệu trong phụ lục D).

Sự phát triển khởi đầu của Ada COCOMO chỉ là một trong các hoạt động trong một cách tiếp cận có 3 ngạnh:

1. *Phát triển bộ các thành phần nền tảng cấu trúc trong Ada để đo hiệu suất biên dịch và cung cấp một bộ tái sử dụng các thành phần nền tảng và hệ thống điều khiển như CCPDS-R.*
2. *Phát triển sự mô tả tiến trình thế hệ tiếp theo để khai thác những kỹ thuật phát triển tương tác và một sự tiếp cận với nền tảng biểu hiện và kiến trúc thượng hạng. Mẫu xử lý Ada này là một bước chính trong xử lý hiện đại để dùng trong các dự án trong lĩnh vực phòng thủ.*

Ví dụ: Dự án 100.000 SLOC (số dòng mã lệnh) mà đòi hỏi 972 tháng công sức làm việc và 30 tháng cho lập lịch.

Công sức

$=2.8\ EAF\ (KDSI)^{1.2}$

$=2.8(1.28)\ (100)^{1.2}$

=900 tháng làm việc của công sức phát triển

+ 72 tháng cho lên kế hoạch, yêu cầu

=972 tháng công sức tổng cộng

Tổng EAF (nhân tố điều chỉnh công sức)(1.28 trong ví dụ này) là kết quả của tất cả ảnh hưởng khiển giải chi phí cá nhân.

Thời gian

$= 2.5\ (Công\ sức)^{0.32}$

$= 2.5\ (900)^{0.32}$

=22 tháng cho lập lịch phát triển

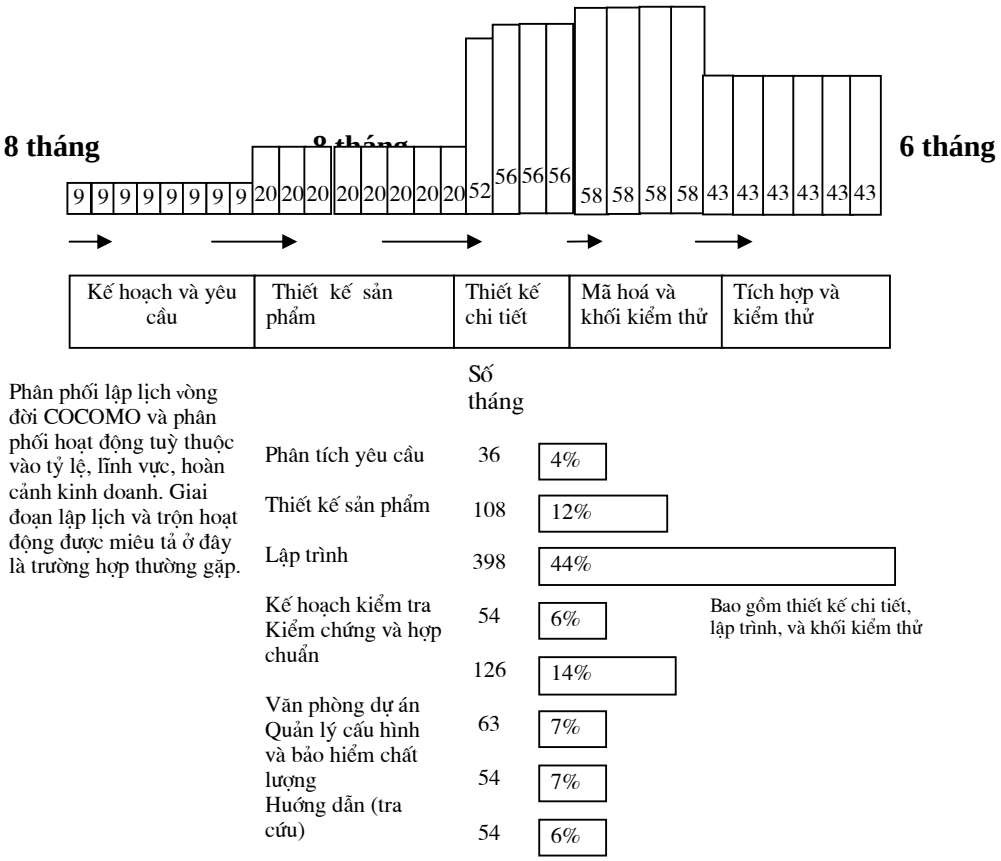
+ 8 tháng cho kế hoạch, yêu cầu

=30 tháng

		Ảnh
Kinh nghiệm ngôn ngữ	Bình thường	1.0
Rằng buộc lập lịch	Bình thường	1.0
Cỡ dữ liệu	Bình thường	1.0
Thời gian xoay vòng máy	Bình thường	1.0
Kinh nghiệm máy ảo	Bình thường	1.0
Tính thay đổi máy ảo	Bình thường	1.0
Sử dụng công cụ phần mềm	Cao	0.88
Thực hành lập trình hiện đại	Bình thường	1.0
Rằng buộc lưu trữ	Bình thường	1.0
Kinh nghiệm ứng dụng	Thấp	1.10
Ràng buộc thời gian	Bình thường	1.0
Tính tin cậy các yêu cầu	Cao	1.15
Mức phức tạp sản phẩm	Cao	1.15
Khả năng cá nhân/đồng đội	Bình thường	1.0

Nhân tố điều chỉnh công sức = 1.28

Sơ thảo làm việc và kết hợp hoạt động



HÌNH B-1. *Phác thảo một dự án*

3. Phát triển 1 phiên bản Ada của COCOMO để miêu tả chi phí và lợi nhuận lập lịch của tiến trình và công nghệ mới này.

Kết quả công sức này là việc TRW đưa đến CCPDS-R, và sự ra đời của Ada COCOMO là chìa khoá để giải quyết toàn bộ vấn đề cho cả bên quản lý cũng như khách hàng. Phiên bản đầu tiên khi đó đã được chính thức hoá trong TRW dưới sự lãnh đạo của Boehm. Một vài kinh nghiệm dự án khác được liên kết, các tham số được trả lại và trọng tâm xử lý được mở rộng bằng cách giới thiệu khái niệm về số mũ tham biến.

Sự tiến triển ban đầu trong Ada COCOMO là loại bỏ 3 chế dộ của COCOMO (chế độ bộ phận, chế độ bán phụ thuộc và chế đô nhúng) và cho phép hệ số mũ được đặt tham số để phản ánh tính kinh tế của những đóng góp tỷ lệ cố hữu trong xử lý phát triển tương tác hiên đại. Một vài sửa đổi chính đã thiết đặt những tham số khác cho những lợi thế công nghệ cố hữu trong môi trường Ada.

Quan hệ ước lượng chi phí Ada COCOMO được cho dưới dạng sau:

Công sức = 2.8 EAF(Kích thước)P

Thời gian = 2.5 (Công sức)0.32

Trong đó:

Công sức =số tháng làm việc

EAF = sản phẩm của 19 nhân tố điều chỉnh công sức (bảng B40)

Kích thước=số chỉ thị nguồn phân phối (theo đơn vị hàng nghìn dòng mã lệnh)

P = số mũ xử lý

Thời gian= tổng tháng làm việc

Bộ nhân EAF lại đại diện cho sự kết hợp các ảnh hưởng của tham số nhân. Trong Ada COCOMO, tuy nhiên, còn một số thay đổi mang lại sự tiến triển thông thường cho COCOMO, các ảnh hưởng xác định Ada, và các ảnh hưởng của một sự xử lý tương tác. Sự điều chỉnh này gây kết quả ra 2 trình điều khiển chi phí mới (RUSE và SECU), một trình điều khiển chi phí được chia (VIRT được chia làm 2 trình điều khiển chi phí mới :VMVH và VMVT) và một số mức độ mới hoặc thay đổi tới ảnh hưởng dưới của trình điều khiển chi phí.

Một trong những nền tảng của Ada COCOMO là việc sử dụng mẫu xử lý Ada. Không cần thiết để sử dụng ngôn ngữ Ada để sử dụng kỹ thuật cơ bản của mẫu xử lý này. Tuy nhiên khi nó mới được phát triển, có quá nhiều Ada underhype và overhype trong thị trường phần mềm bảo vệ đến mức mà sự mô tả xử lý cũng được gắn với sự xử dụng của Ada. Nhìn lại mô hình xủ lý Ada có thể xem như một trạng thái tức thì giữa xử lý thông thường và khung xử lý hiện đại được mô tả trong sách này. Các chiến thuật mô hình xử lý Ada tóm tắt ở đây cung cấp sự hiểu biết về tiến trình tham số hoá của hệ số mũ Ada COCOMO

Bảng B-4. *Sự cải tiến Ada COCOMO thành các nhân tố điều khiển công sức*

Định danh	Nhân tố điều chỉnh công sức	Sự nhiễu loạn Ada COCOMO
RELY	Yêu cầu tin cậy	Các thay đổi dưới ảnh hưởng (tác động dương)
DATA	Cỡ dữ liệu	Không thay đổi
CPLX	Mức phức tạp sản phẩm	Các thay đổi dưới ảnh hưởng (tác động âm)
RUSE	Cấp tái sử dụng đòi hỏi	ảnh hưởng mới cho tính phức tạp của cấc thanh phần tái sử dụng
SECU	Rằng buộc an toàn	ảnh hưởng mới cho dự án phân loại
TIME	Rằng buộc thời gian thực hiện	Không thay dổi
STOR	Rằng buộc bộ nhớ chính	Không thay đổi
VIRT	Tính thay đổi máy ảo	Bỏ đi (chia ra thành 2 nhân tố mới)
VMVH	Tính thay đổi máy chủ VM	ảnh hưởng mới cùng với các khía cạnh máy chủ của VIRT
VMVT	Tính thay đổi mục tiêu VM	ảnh hưởng mới cùng với các khía cạnh mục tiêu của VIRT

TURN	Thời gian xoay vòng máy	Cấp độ mới cho đáp ứng tương tác (tác động dương)
ACAP	Khả năng phân tích	Thay đổi tới các ảnh hưởng dưới (nhiều tác động hơn)
AEXP	Kinh nghiệm ứng dụng	Không thay đổi
PCAP	Khả năng người lập trình	Thay đổi tới các ảnh hưởng dưới(ít tác động hơn)
VEXP	Kinh nghiệm máy ảo	Không thay đổi
LEXP	Kinh nghiệm ngôn ngữ	Thay đổi tới các ảnh hưởng dưới (nhiều tác động hơn)
MODP	Thực hành lập trình hiện đại	Thay đổi tới các ảnh hưởng dưới (nhiều tác động hơn)
TOOL	Sử dụng công cụ phần mềm	Cấp độ mới cho hỗ trợ tự động
SCED	Lập lịch phát triển yêu cầu	Thay đổi tới các ảnh hưởng dưới (ít tác động hơn)

Một chiến thuật đặc biệt của mô hình xử lý Ada COCOMO là nhấn mạnh cột mốc xét lại thiết kế mở đầu, mà cần được yêu cầu bởi chuăn mực quân sự ứng dụng, như một sự xét lại có tính cấu trúc được cung cấp bởi một thuyết minh có khả năng thực hiện của các khả năng. Mục đích mày dẫn đến một số chiến thuật mà khai thác công nghệ, công cụ, và kỹ thuật của môi trường Ada:

- Một nhóm thiết kế hạt nhân nhỏ với kinh nghiệm trong kiến trúc phần mềm và lĩnh vực ứng dụng
- Một trọng tâm sớm trên bộ xương kiến trúc có thể thực hiện cho thuyết minh các thành phần và cảnh báo cấp hệ thống cho rủi ro trước mắt.
- Thiết kế chi tiết độc lập và gia tăng đi qua các thành phần và các tạo lập hơn là một thiết kế duy nhất cho toàn hệ thống.
- Tích hợp liên tiếp qua biên dịch Ada và phát triển tiền cấu trúc
- Kiểm tra chương trình và xác minh đòi hỏi tập trung vào kiểm tra chuỗi công nghệ (nay gọi là trường hợp sử dụng) và kiểm tra thành phần đứng độc lập
- Mã Ada tự chế bản và mô tả toàn cảnh thay vì tài liệu thiết kế chi tiết, lớn mà mô tả những thiết kế mới xây dựng
- Các độ do tự động bắt nguồn từ dòng mã lệnh cơ bản.

Hệ số mũ xử lý Ada COCOMO có phạm vi từ 1.04 đến 1.24 và được định nghĩa từ các ảnh hưởng kết hợp của 4 thông số sau:

1. Kinh nghiệm mô hình xử lý Ada. Tỷ lệ hạn định xử lý có phạm vi từ "không quen thuộc" với xử lý cho đến sử dụng thành công trên các dự án nhiều bộ phận.

2. Thiết kế chu tất tại PDR. Tham số này tạo đặc điểm cấp độ của thiết kế chi tiết cố hữu trong thiết kế cơ bản chứng tỏ tại PDR. Nó có phạm vi từ "ít chu tất (20%)" tới "chu tất hoàn toàn (100%)"
3. Các sự cố loại bỏ tại PDR. Tham số này định cấp độ của tính cố hữu chưa chắc chắn trong dự án tại PDR, sau sự phát triển tổng thể được khởi tạo.Nó có phạm vi từ "giải pháp ít rắc rối" tới "giải pháp hoàn toàn"
4. Sự thay đổi yêu cầu trong quá trình phát triển.Tham số này có phạm vi từ "thay đổi rất lớn" đến "không thay đổi", tạo đặc điểm số biến động xử lý giới hạn bởi dự án.

*Số mũ thực tế cho Ada COCOMO đượcc xác định bởi tổng các tỷ lệ cho mỗi tham số có phạm vi từ 0.00 đến 0.05. Hệ số mũ chế độ nhúng (1.20) từ mô hình COCOMO ban đầu có liên quan tới xử lý COCOMO với tỷ lệ 0.04 trên mỗi thông số xử lý [1.04 + (4*0.04)].Theo các tham số vừa mô tả,. điều này tương ứng với 1 là ít quen thuộc với mẫu xử lý Ada, với 2 là thiết kế chu tất 40%, với 3 là loại bỏ rắc rối tại PDR 40% và với 4 là thường xuyên nhưng vừa phải với những thay đổi yêu cầu.*

Bốn tham số này hỗ trợ cho tạo đặc điểm xử lý khả năng của nó làm giảm nhẹ tính phi kinh tế của tỷ lệ cố hữu trong xử lý thông thường. Bằng cách giữ nhóm thiết kế nhỏ hơn và thiết lập một sự mô tả kiến trúc linh động hơn bởi PDR cách xử lý đang cố gắng để tối ưu hoá truyền thông bên trong, tránh làm lại luồng truyền xuống muộn, và khuyến khích tập trung yêu cầu sớm hơn.

B-3. COCOMO II

Dự án COCOMO II là một nỗ nực được thực hiện bởi Trung tâm USC cho ccông nghệ phần mềm, với sự hỗ trợ về tại chính và kỹ thuật của nhiều tập đoàn công nghiệp lớn. Các đối tượng của dự án này gồm 3 phần:

5. Để phát triển một mô hình ước lượng lập lịch và chi phí phần mềm cho các thực hiên chu trình trong những năm 1990 và 2000.
6. Để phát triển một cơ sở dữ liệu chi phí phần mềm và hỗ trợ công cụ cho sự phát triển của mẫu chi phí.
7. Để cung cấp một khung phân tích số lượng cho công nghệ định giá trị phần mềm và các ảnh hưởng kinh tế của chúng.

USC ước tính rằng thị trường phần mềm 2000 sẽ bao gồm 5 cộng đồng khác biệt như sau:

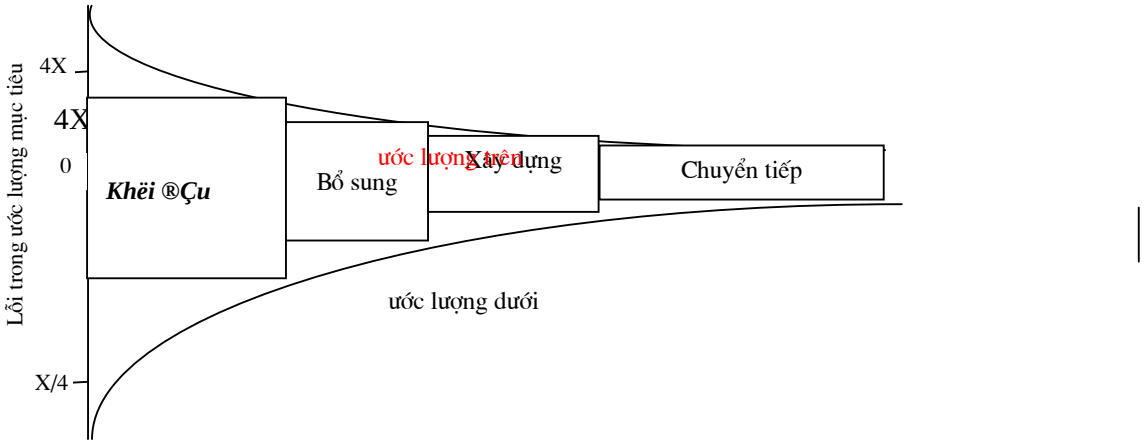
8. Những người lập trình sử dụng cuối (5.5 triệu) tạo nên các bảng tính hoặc các truy vấn dữ liệu.
9. Những người phát triển thành phần (600.000) tạo nên các ứng dụng sử dụng cuối và các hỗ trợ thành phần.
10. Người tích hợp thành phần (700.000) xây dựng những ứng dụng một cách nhanh chóng từ những người tạo lập GUI đã tồn tại, những người quản lý dữ liệu/đối tượng, trung gian, các thành phần đặc trưng lĩnh vực.
11. Người tích hợp hệ thống (700.000) đảm nhiệm những hệ thống có quy mô lớn, những hệ thống chưa từng thấy, các loại ứng dụng, các hệ thống nhúng đòi hỏi tiền kiến trúc và những sự phát triển phần mềm lựa chọn.
12. Người phát triển cơ sở hạ tầng (750.000) phát triển các thành phần độc lập theo lĩnh vực như hệ điều hành, hệ thống quản lý cơ sở dữ liệu, mạng, và khung giao diện người dùng.

Những người dùng cuối không phải là đối tượng của COCOMO II bởi vì chúng có khuynh hướng thực hiện những công sức có quy mô nhỏ, nhanh, chỉ tương ứng với những ước lượng có quy mô nhỏ hoạt động đơn giản. Các dự án mà đòi hỏi nhóm những người làm việc hàng tháng trời thậm chí hàng năm mới là thị trường chính của các mô hình ước tính chi phí phần mềm.

Chiến thuật của COCOMO II là gìn giữ tính mở của mô hình COCOMO nguyên thuỷ, thiết đặt nó trong thị trường vừa mô tả, dẫn đầu vào và đầu ra tới cấp thông tin sẵn có, và kích hoạt mô hình để đượoc thiết lập cho các chiến thuật xử lý dự án khác nhau. Đặc biệt, thế hệ COCOMO này cung cấp các ước lượng có phạm vi lớn hơn so với ước lượng điểm. Những ước lượng này thay đổi chu trình từ rât sớm,những đầu vào thô, và sau đó là các ước lượng có quy mô lớn, các đầu vào tinh, và ước lượng chi tiết hơn. Hình B-2 minh hoạ tính chính xác ước lượng qua một chu trình.

Để hỗ trợ chiến thuật này, COCOMO II định nghĩa 3 mô hình cho ước lượng chi phí. Hình B3 ánh xạ những mô hình này cho các pha của vòng đời tương tác. Các mô hình đại diện cho cấp chính xác và xấp xỉ bất định cho pha hiện tại cảu vòng đời. Mô hình thiết kế hậu tương ứng mô thình COCOMO nguyên thuỷ, khi nó được giả định dự án có yêu cầu cố định, kế hoạch, và kiến trúc đề bạt tại tập ra. Dự án sau đó sẽ kế được tiếp tục bởi một xử lý dạng mô hình thác nước trong suốt quá trình chuyển giao với các thay đổi yêu cầu nhỏ. Mô hình hậu kiến trúc cung cấp cho ước lượng của dự án khi nó có những cơ sở yêu cầu, cơ sở kiến trúc, và một kế hoạch cho giai đoạn kiến trúc. Mô hình thiết kế sớm cung cấp cho ước lượng trong giai đoạn bổ sung của chu trình, và mô hình thành phần ứng dụng cho phép ước lượng trong quá trình khởi tạo của dự án.

Mô hình thành phần ứng dụng tương ứng công việc khám phá được làm trong nỗ nực lấy mẫu và phân tích khả năng có thể tin cậy được. Biểu thức ước lượng là quan hệ tuyến tính đơn giản của các điểm đối tượng và tính phức tạp lĩnh vực.



Hình b-2. *ước lượng phần mềm qua một vòng đời dự án.*

Mô hình thiết kế sớm tương ứng cấp độ của sự có sẵn trong giai đoạn kiến trúc của dự án, trong khi kiến trúc, yêu cầu và kế hoạch đang được tổng hợp.

Biểu thức ước lượng chi phí tổng thể như sau:

Công sức = 2.54 EArch (kích thước)^{P}

Trong đó:

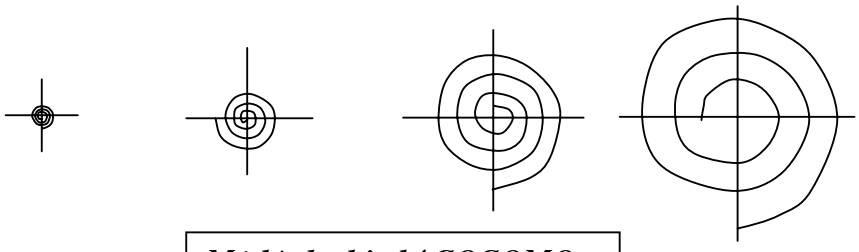
Công sức=số tháng làm việc

Earch = sản phẩm của 7 nhân tố điều chỉnh công sức thiết kế sớm.

Kích thước = số điểm chức năng hoặc KSLOC

P = mũ xử lý

Khởi đầu	*Bổ sung*	**Xây dựng**	**Chuyển tiếp**



Mô hình chi phí COCOMO

Mô hình lấy	Mô hình thiết kế	Mô hình hậu kiến trúc
Đầu vào thô Ước lượng chính xác dưới Yêu cầu thô Khái niệm kiến trúc	Dự án thấu hiểu Ước lượng chính xác vừa Yêu cầu thấu hiểu Kiến trúc thấu hiểu	Đặc điểm dự án chi tiết ước lượng chính xác cao Cơ sở đòi hỏi cố định Cơ sở kiến trúc cố định

Hình B-3: *ước lượng COCOMOII trên một vòng đời dự án*

Các tham số giai đoạn thiết kế sớm là các thành phần của các tham số hậu kiến trúc. Chúng cung cấp một mô hình ước lượng đơn giản hơn cho những vòng đời sớm khi có nhiều biến chưa biết.

Biểu thức ước lượng chi phí hậu kiến trúc được cho như sau:

Công sức = 2.45 EApp(Kích thước) ^{P}

Trong đó:

Công sức = số tháng làm việc

EApp= snả phẩm của 17 nhân tố điều chỉnh công sức hậu kiến trúc

Kích thước = số KSLOC hoặc điểm chức năng

P = số mũ xử lý

Bảng B-5. *Nhân tố điều chỉnh công sức mô hình thiết kế sớm*

Định danh	Nhân tố điều chỉnh công sức thành phần
Độ phức tạp sản phẩm	RELY-DATA-CPLX-DOCU
Tái sử dụng yêu cầu	RUSE
Độ khó nền	TIME-STOR-PVOL
Kinh nghiệm cá nhân	AEXP-PEXP-LTEX

Khả năng cá nhân	ACAP-PCAP-PCON
Tiện ích	TOOL- SITE
Lập lịch	SCED

Bảng B-6. *Mô hình hậu kiến trúc COCOMO II nâng cấp từ Ada COCOMO và COCOMO*

Định danh	Nhân tố điều chỉnh công sức	Sự nhiễu loạn COCOMO II
RELY	Yêu cầu tin cậy	Không thay đổi từ COCOMO
DATA	Cỡ dữ liệu	Không thay đổi từ COCOMO
CPLX	Mức phức tạp sản phẩm	Không thay đổi từ COCOMO
RUSE	Cấp tái sử dụng đòi hỏi	Không thay đổi từ Ada COCOMO
DOCU	Chế bản	Được thêm, tính phù hợp của chế bản cho nhu cầu vòng đời
TIME	Rằng buộc thời gian thực hiện	Không thay đổi từ COCOMO
STOR	Rằng buộc bộ nhớ chính	Không thay đổi từ COCOMO
PVOL	Tính phức tạp nền	Kết hợp tham số VMVH và VMVT của Ada COCOMO thành tính phức tạp nền
ACAP	Khả năng phân tích	Không thay đổi từ COCOMO
AEXP	Kinh nghiệm ứng dụng	Không thay đổi từ COCOMO
PCAP	Khả năng người lập trình	Không thay đổi từ COCOMO
PCON	Sự nối tiếp cá nhân	Tham số mới
LTEX	Kinh nghiệm ngôn ngữ/công cụ	Thay đổi để bao gồm cả kinh nghiệm ngôn ngữ và kinh nghiệm công cụ
SITE	Phát triển đa mặt Liên kết nhóm	Tham số mới cho cấp sắp xếp vào một chỗ và cấp tự động trong nhóm
TOOL	Sử dụng công cụ phần mềm	Không thay đổi từ COCOMO
SCED	Lập lịch phát triển yêu cầu	Không thay đổi từ COCOMO

Các hệ số E biểu diễn các ảnh hưởng kết hợp cảu nhiều tham số. Mô hình hậu kiến trúc dùng các tham số giống các tham số được sử dụng bởi mô hình COCOMO nguyên thuỷ. Những tham số này cho

phép môi trường phát triển có đặc điểm và bình thường hoá với các tham số trong cơ sở dữ liệu dự án COCOMO II (83 dự án). ảnh hưởng của mỗi thiết lập tham số (rất thấp, thấp, bình thường,cao, rất cao) là bộ nhân mà có phạm vi từ 0.5 đến 1.5. Sản phẩm của 17 ảnh hưởng này được dùng để tính toán công sức trong biểu thức ước lượng.

Tên của mô hình hậu kiến trúc diễn tả kết quả của pha thiết kế sớm, thường là kiến trúc. Việc sử dụng các điểm chức năng được khuyến cáo để định lượng kích thước cho pha tiền thiết kế, bởi vì các điểm chức năng phù hợp hơn với các pha sớm khi cấu trúc (và vì thế ước lượng SLOC) cho giải pháp phần mềm là không thể biết. Việc sử dụng SLOC được nhắc nhở để định lượng kích thước cho mô hình hậu kiến trúc. Điều này dường như trở thành một sự gắn kết kỹ thuật tốt giữa phe SLOC và phe điểm chức năng

COCOMO II dùng hệ số mũ cho pha thiết kế sớm và những mô hình hậu kiến trúc. Hệ số mũ xử lý có phạm vi từ 1.01 đến 1.26 và nó được định nghĩa như là ảnh hưởng kết hợp của 5 tham số sau:

1. Các tiêu chuẩn ứng dụng: cấp kinh nghiệm lĩnh vực chính của cơ quan phát triển.
2. Tính phức tạp xử lý: cấp độ đối lập, nghi thức, các hoạt động chu trình, và truyền thông những người nắm chi phí trước.
3. Phân tích tính rắc rối kiến trúc: cấp độ tính tin cậy kỹ thuật trước khi đưa tới sản xuất tổng thể.
4. Gắn kết nhóm: cấp độ phối hợp à chia xẻ giữa mọi người (người mua, người phát triển, người sử dụng và người bảo trì, trong những người khác.
5. Sự hoàn thành xử lý: cấp độ hoàn thành cuả cơ quan phát triển, như đã định nghĩa trong mô hình hoàn thành khả năng của viện công nghệ phần mềm.

Hệ số mũ COCOMO II là sự tiến bộ của Ada COCOMO đạt được với nền tảng rắn chắc hơn.Bảng B7 tóm tắt các tỷ lệ tham số. Số mũ thực tế cho COCOMO II được quyết định bởi tổng các ảnh hưởng cho mỗi tham số. Tác động kết hợp của những tham số xử lý này có thể rất cao. Nhóm COCOMO II vẫn chưa thừa nhận một tính kinh tế thực tế của lỷ lệ được thu (đó là gía trị của P không bao giờ nhỏ hơn 1.0). Họ tin rằng tính kinh tế của tỷ lệ được thu qua sự rút bớt kích thước gây kết quả từ việc dùng các thành phần thương mại. các thành phần tái sử dụng.

Bảng B-7. *các tham số mũ xử lý COCOMO II*

Tham số	Rất thấp (0.00)	Thấp (0.01)	Bình thường (0.02)	Cao (0.03)	Rất cao (0.04)	Thực sự cao(0.05)
Tính tiêu chuẩn	Hoàn toàn chưa tiêu chuẩn	Chưa tiêu chuẩn nhiều	Chưa tiêu chuẩn ít	Quen thuộc thông thường	Quen thuộc rộng	Hoàn toàn quen thuộc
Tính phức tạp phát	Thô	Buông lỏng tạm thời	Hơi buông lỏng	Phù hợp thông	Hơi phù hợp	Mục tiêu tổng

triển				thường		thể
Độ phân giải rủi ro kiến trúc	ít (20%)	Một vài (20%)	Thường xuyên (60%)	Rộng (75%)	Phần lớn (90%)	Hoàn toàn (100%)
Tính gắn kết tập thể	Rất khó tương tác	Hơi khó tương tác	Kết hợp nền tảng	Kết hợp rộng	Kết hợp cấp cao	Kết hợp chặt chẽ
Hoàn thành tiến trình	Cấp 1	Cấp 2	Cấp 2+	Cấp 3	Cấp 4	Cấp 5

Một điều phát triển thú vị nữa của COCOMO II là biểu thức ước lượng lập lịch, mà là chức năng của cả ướclượng công sức và tham số xử lý.Tác động thu đượccảu xử lý tôt hơn là giảm cả công sức lẫn việc lập lịch trình. Tóm tại COCOMO II là sự tiến triển tốt trên các mô hình chi phí, mà rất nhiều trong số chúng đã lỗi thời. Nó là một kết nối tốt cho phát triển tương tác, công nghệ hiện đại, và quản lý xử lý trong sách này. Tuy nhiên nó cũng chỉ là chưa chín muồi, và cơ sở dữ liệu dự án của nó còn đòi hỏi các dự án khác nhau từ nhiều tổ chức. Khó mà tin được rằng nó còn tin cậy hơn so với mô hình COCOMO ban đầu.

Ma trận thay Đổi

	Những điểm mấu chốt
Tiến độ và chất lượng đo lường phần mềm là một công việc cực kỳ phức tạp bởi vì số lượng sản phẩm ,dự án , và các tham số toàn cục lớn mà nó có sự tác động tới những nỗ lực phát triển phần mềm. có lẽ không thể định rõ một tập các định nghĩa chính xác về đo lường phần mềm mà chúng sẽ đáp ứng được hầu hết các dự án phần mềm.	+ Một trong những đặc tính quan trọng nhất của một phần mềm tốt là sự thay đổi thuận tiện của nó . + Nỗ lực đo lường và đánh giá một cách manh mún và tái tạo trong một chuỗi các đường biên phần mềm cung cấp sự hiểu biết sâu sắc hữu ích đối với sự hội tụ hoặc phân rẽ có thể chấp nhận được về mặt chất lượng và tiến độ. + Các ma trận được trích trực tiếp từ các vật dụng kĩ thuật mà ra cung cấp một dẫn chứng cho việc dàn dựng tiến trình mà nó có thể điều khiển nmhất quán , chính xác và điều khiển dự án một cách chi tiết.

Nét đặc trưng quan trọng nhất của phần mềm đó là "mềm" : sự thuận tiện của phần mềm là khả năng thay đổi , thuận lợi cho việc lưu trữ lại nhằm đạt được những nét đặc trưng yêu cầu khác .

Vì vậy các ma trận trung tâm được tập trung cho những đo lường các xu hướng thay đổi phần mềm (chia nhỏ , tái tạo) trong các sản phẩm phần mềm ở khắp nơi, xuyên suốt trong toàn bộ vòng đời .Để cố gắng quản lý phần mềm hệ trọng, quản lý dự án phần mềm càn có một số phạm vi ma trận độc lập (cho sự so sánh với các dự tính chung) và một số ma trận độc lập có giới hạn .

Tôi đã phat triển rất nhiều thiết bị này trong năm 1987 để hợp lý hoá chương trình các ma trận đã sử dụng cho dự án ccpds-r , nó được thực hiện như nghiên cứu một trường hợp trong phụ lục d. tài liệu được công bố {royce, walker, 1990} sau 3 năm nghiên cứu lĩnh vực này đã chứng minh được sự hữu ích và đã có kết quả trong một vài lần tinh chế .Đã có rất nhiều những cố gắng khác qua 20 năm qua để định nghĩa các đo lường chất lượng phần mềm .Có một vài nguyên nhân không ai nắm bắt được trong thực nghiệm mặc dù có một số người tái diễn lại thực trạng này và nó dẫm đạp lên sự khuyên nhủ thẳng thắn của tôi. Một số trở ngại tiếp nối là sự cần thiết đối với tính chủ quan và chi phí của các nguồn lực con người đòi hỏi tập hợp lại và làm sáng tỏ các ma trận .

C-1 Khái quát chung

Sự tiếp cận của tôi với các ma trận tương tự với De Marco là người đề xướng ra đo chất lượng phần mềm thông qua việc làm mất đi các hư hỏng [De Morco, 1982] .Để giữ lại công nghệ và đề án độc lập .Các định nghĩa của ông là mập mờ có chủ định: Lý trí của tôi là khá rõ ràng. Tính ổn định của ứng dụng là quan trọng đối với sự lập luận một cách chính xác, chỉ có giá trị với phương pháp đánh giá chủ quan. Đánh giá giá trị phần mềm cần có sự đánh giá đầu vào một cách chủ quan và đánh giá các đầu ra một cách khách quan. Cách tiếp cận của tôi sẽ định nghĩacác đầu vào khách quan. Điều đó có thể đòi hỏi lập luận chủ quan trong ngữ cảnh của một dự án cụ thể .Một cách có hiệu quả để đánh giá chất lượng phần mềm là tái tạo đo lường trong đường biên cấu hình. Đơn vị đo lương có thể là dòng mã nguồn, các điểm chức năng, những điểm khách quan, các tệp, các thành phần hoặc một số cách đo kích thước phần mềm

khác. Thảo luận này sử dụng SLOC như là ma trận kích thước gốc bởi vì phần lớn nó được dùng trong công nghiệp, nó là cách đo lường dễ nhất có thể hiểu được. Dữ liệu nghiên cứu trường hợp này ở trong phụ lục D.

Trong một số trường hợp sự đánh giá chất lượng phần mềm bắt nguồn từ sự tổng hợp khách quan của các ma trận thay đổi nó sẽ đòi hỏi sự đánh giá độc lập ngữ cảnh. Khả năng suy xét là cần thiết để đánh giá chất lượng sử dụng bất cứ ma trận nào. Các ma trận giống nhau sẽ được sử dụng để đánh giá chất lượng trong khi phát triển (xu hướng - được định hướng) và sự phát triển sau này (giá trị định hướng) Ví dụ: Dung lượng tái tạo theo phân phối sản phẩm là một sự đo lường khách quan chất lượng hoặc chất lượng kém. Tổng số tái tạo theo đường biên cấu hình tổng thể phát triển là mơ hồ không tồn tại ngữ cảnh xa hơn. Tái tạo có thể được giải thích như một đường biên hoàn hảo (Nó không được ưa thích), một chương trình kiểm tra không đầy đủ hoặc sự xây dựng ban đầu đáng thất vọng.

*Chất lượng phần mềm

Rất khó khăn để đưa ra một khái niệm khách quan về vấn đề này. Chỉ có hai phương pháp là có sẵn cho mong muốn của khách hàng để định nghĩa về chất lượng: Phần mềm yêu cầu chỉ rõ về chức năng và hoạt động, và một kế hoạch tiêu dùng được chấp nhận, chi phí số lượng và các mục đích rõ ràng. Những vật dụng tự tạo này về cơ bản là tương đương với hợp đồng, theo truyền thống thì những sản phẩm chất lượng kém được sản suất bởi một dự án bởi vì chúng được chấp nhận từ rất sớm trong vòng đời, khi mà còn quá nhiều điều còn chưa được biết tới. Các ma trận phần mềm hiện đại, lặp lại và khách quan sẽ cung cấp khả năng hiểu biết sâu sắc hơn trong một chừng mực nào đó về chức năng, hoạt động ,giá cả và tuân theo ấn định thời lượng và các mong muốn của khách hàng.

*Các thứ tự thay đổi phần mềm

SCOs đã được thảo luận đầy đủ và chi tiết trong chương 12, phương hướng thiết lập để tiến hành với thay đổi một cấu hình bộ phận phần mềm (SCOs thường được gọi là báo cáo sự ccố phần mềm, nhưng là sự cố có tình huống cấm đoán, và không phải tất cả mọi thay đổi bị thúc đẩy bởi các sự cố). Sự thay đổi có thể được cần đến để tái tạo một thành phần chất lượng kém (loai 0,1,một sự phục hồi), để tái tạo một thành phần đạt chất lượng cao hơn (loại 2, sự nâng cao), hoặc để điều tiết trực tiếp một khách hàng trực tiếp thay đổi yêu cầu (loại 3 thay đổi phạm vi). Sự khác nhau giữa ổn định và nâng cao là tính cố hữu trong lý do thay đổi. Giả thiết rằng thành phần ổn định là chấp nhận được, nếu lý do thay đổi là để làm tăng chi phí hiệu lực, tăng khả năng kiểm tra, tăng sự tiện lợi, tăng hiệu qủa trong một số phương thức khác nhau ,sự tái tạo là loại 2. Cả loại 0, hoặc 1 và 2 tái tạo làm tăng chất lượng sản phẩm cuối. Tuy nhiên loại 0 hoặc 1 cũng cho biết chất lượng không tương xứng trong đường biên hiện thời.Trong thực nghiệm sự khác nhau giữa loại 0 hoặc 1 và 2 có thể hoàn toàn do chủ quan. Với thảo luận sau đó, hầu hết các ma trận là không nhay cảm với sự phân loại, nhưng nếu sự khác nhau là ở chỗ ứng dụng, nó có thể cung cấp sự hiểu biết hữu dụng.

Loại 3 SCOs loại điển hình phản ánh được bản chất sự thay đổi các yêu cầu mà mong muốn của khách hàng là định nghĩa lại .Những thay đổi đó có tác động mạnh mẽ và việc đó đòi hỏi các cấp độ phần mềm và công nghệ các hệ thống khác nhau cũng như các mức biến đổi cao của sự kiểm tra thoái hoá. Bởi vì giới hạn mở rộng này có thể thay đổi, loại 3 SCOs sẽ được phân tích một cách riêng biệt trong ma trận này. Dữ liệu bắt nguồn từ loại 0, 1 và 2 SCOs sẽ cung cấp nguyên tắc hợp lý cho sự duy trì đánh giá và nỗ lực đòi hỏi đối với loại 3 SCOs.

* Dòng mã nguồn

Dù sloC cung cấp một ma trận tốt cho việc đo lường thì vấn đề số lượng phần mềm luôn luôn được bàn cãi. Jones đồng nhất một số phòng ngừa cần thiết khi xử lý SLOC (Jones 1994) ông ta đi tới một cấp độ để nói rằng " các dòng mã sử dụng tiêu chuẩn hoá các thành phần liên quan đến dữ liệu hoặc các ngôn ngữ khác sẽ được xem xét, một ví dụ của sự bất cẩn nghề nghiệp". Một điểm mà tất cả mọi người đồng ý đó là bất cứ một cái gì được sử dụng phải được định nghĩa một cách khách quan và ổn định, là tiêu chuẩn để so sánh. Một thành phần thuần tuý nào đó của SLOC được định nghĩa thì nó không quan trọng bằng cách định nghĩa nó một cách kiên định thông qua tất cả các dự án và tất cả các lĩnh vực dự án cụ thể . Việc yêu cầu sử dụng công cụ đếm thông dụng nhằm đạt tới một định nghĩa tiêu chuẩn hoá được đề ra.

*Mô hình bảng điều khiển

Một đường biên cấu hình là một bộ các sản phẩm mà nó có tính chất khách quan để thay đổi điều khiển thông qua CCB. Đường biên cấu hình có thể mô tả các sản phẩm ngay lập tức mà các sản phẩm này đã được thiết kế hoàn chỉnh, phát triển và kiểm tra độ bất hợp lệ, hoặc các sản phẩm cuối đã được kiểm tra độ bất hợp lệ.

C-2 Nguồn gốc các ma trận

Phần này miêu tả và định nghĩa một cách chi tiết các số liệu thống kê cần thiết đã được sưu tầm. Các ma trận có nguồn gốc từ những số liệu thống kê này và một số nguyên tắc chỉ đạo chung cho sự hiểu biết về chúng. Phụ lục D cung cấp một số ví dụ chi tiết của các ứng dụng thế giới thực tại nhằm minh hoạ cụ thể hơn, các ma trận như vậy có thể được sử dụng như thế nào để quản lý và điều khiển các dự án .Nguồn gốc không phải rõ ràng là sự phát triển từ trên xuống. Hơn thế nữa họ đưa ra kết quả từ kiểm tra thực chất và lỗi các phân tích thực nghiệm bằng số, khả năng trực giác và tự tìm kiếm.

Các số liệu thống kê thô được sưu tầm bao gồm các số và các loại phần mềm thay đổi, hư hỏng SLOC, phục hồi SLOC. Những thử thách nhằm tìm ra chính xác các kĩ thuật chọn lọc cho các số liệu chỉnh sửa thô nhằm thấy được các xu hướng hữu ích và để tìm ra các đo lường khách quan tiến tới xác định được số lượng (các thuộc tính phát triển trung gian) và chất lượng (thuộc tính của sản phẩm cuối). Sự khách quan cuối cùng nhằm xác định số lượng các sản phẩm lắp ráp, khả năng thích nghi, sự thuần thục và khả năng duy trì là tinh tế hơn.

*Sự lắp ráp: Ma trận này đo lường phạm vi ngắt trung bình hoặc đoạn cắt. Nó nhận biết sự cần thiết để xác định số lượng các đoạn cắt (dung lượng thiệt hại SLOC) và số các trường hợp tái tạo lại (số lượng SCOs). Thực vậy sự lắp ráp được định nghĩa như là quá trình cục bộ hoá đo lương phần ngắt, với giá trị càng nhỏ càng tốt.

* Khả năng thích nghi: Ma trận này đo lường độ phức tạp trung bình của việc phân tách như đo lường trong quá trình tái tạo lại. Nó nhận biết sự cần thiết xác định số lượng tái tạo lại (nỗ lực đòi hỏi giải quyết) và số lượng các trường hợp tái tạo lại (Số lượng SCOs). Khả năng thích nghi xác định số lượng, sự dễ dàng thay đổi, giá trị này càng nhỏ càng tốt.

* Tính thuần thục: Sự trực giác, tính thuần thục tương ứng với cấp độ của tính chất chân thật đáng coi trọng của sản phẩm. Sự khách quan, những ma trận này đo lường tỉ lệ sai sót. Mục đích là tự dặt tên các hư hỏng, để hoàn thành tính thuần thục vô hạn. Sự sự phát triển chủ yếu là qua mở rộng bởi vì phần mềm là sự vận dụng trí óc, không chứa đựng vấn đề vật lý, nó không phơi bày ra ngoài. Phần mềm sẽ trưởng thành cùng thời gian , có nghĩa là những người dùng nó (Nhóm kiểm tra, những người dùng bậc 2, những người sử dụng sản phẩm thông dụng) sẽ có kinh nghiệm để các sai sót hay mắc ít hơn đối với mỗi sản phẩm đưa ra kế tiếp.

Điều thông báo này thừa nhận chức năng lặp lại và hoạt động trong sự công bố mới. Sự mong muốn tăng tính thuần thục là không thể thực hiện ngay cả khi thay đổi chức năng và hoạt động. Tương tự ta có thể chú ý tới các xu hướng trong các công bố chuẩn xác qua sự phát triển trong vòng đời của nỗ lực phát triển có ảnh hưởng tới sức khoẻ. Một chỉ dẫn đơn giản về tỉ lệ sai sót sẽ đòi hỏi tới việc đo lương các hư hỏng (loại 0 và 1 SCOs) và tổng số thời gian sử dụng. Từ những tham số này , ý nghĩa thời gian giữa các lần hư hỏng (MTBF) có thể được bắt nguồn từ công bố đã đưa ra. Các độ chuyên sâu cao hơn thì tốt hơn, sự phản hồi thời gian trung bình giữa các hư hỏng được nhân thức bởi một người sử dụng.

* Khả năng duy trì: Về lý thuyết khả năng tồn tại của một sản phẩm được bắt nguồn từ khâu sản xuất mà với nó nhóm bảo quản có thể làm. Rất khó khăn để so sánh năng suất giữa các dự án tuy nhiên, những khẳng định này bằng trực giác thì chưa thuyết phục lắm. Định mức tiêu chuẩn hoá năng suất, hiệu chỉnh sự tăng năng suất cung cấp giá trị sản xuất độc lập phản ánh được sự phát triển phức tạp. Tỉ lệ tiêu chuẩn hoá những khác nhau năng suất dự án cung cấp ma trận so sánh quan hệ. Kết quả là khả năng bảo quản được định nghĩa như là định mức năng suất hiệu chỉnh nhằm tăng năng suất. Theo trực giác, giá trị này khẳng định rằng một sản phẩm có thể bị thay đổi 3 lần bằng khả năng phát triển (Khả năng tồn tại =0.33) có khả năng bảo quản tốt hơn một sản phẩm được thay đổi 2 lần bằng khả năng (khả năng tồn tai=0.5) nó đượcphát triển, sự độc lập của năng suất duy trì tuyệt đối thực tại. Các số liệu thống kê cần phải tính toán những giá trị này là tổng số nỗ lực phát triển, tổng số SLOC, tổng số nô lực thay đổi, tổng số SLOC được hiệu chỉnh.

Trong khi các giá trị này cung cấp những đo lường khách quan hữu ích của các sản phẩm cuối. Những giá trị trung gian như chức năng của thời gian cung cấp sự hiểu biết sâu sắc để phát triển thành các giá trị sản phẩm cuối mong muốn. Mỗi dự án thu được một số kinh nghiệm phát triển ban đầu, kinh nghiệm này sẽ rất hữu ích để dự báo trước sự thay đổi cố hữu trong xu hướng phát triển.

Nguồn gốc ngắn gọn này có cấp độ đơn giản. Nó không cần thiết để giải quyết những ma trận này như một tập hoàn chỉnh, mặc dù có nhiều viễn cảnh phức tạp được yêu cầu bởi việc quản lý dự án nhằm thực hiện một cách chuẩn xác. Các tập con hoặc các tập khác cũng hữu ích, hầu hết các phân tích, toán học, sưu tầm dữ liệu cố hữu trong các ma trận sẽ tự động hoá vì vậy những người thi hành chỉ cần làm sáng tỏ kết quả và hiểu biết cơ bản về chúng.

C-2-1 Các số liệu thống kê tĩnh

Một số số liệu thống kê cụ thể phải được ổn định trên vòng đời dự án phần mềm nhằm thực hiện các ma trận đã đưa ra. Những số liệu thống kê này được nhận biết trong bảng 1 bao gồm hnư sau:

- Tổng số SLOC (SLOCT): Những số liệu thống kê này theo dõi tổng số lượng ước đoán kích thước sản phẩm đang trong tình trạng phát triển. Giá trị này có thể thay đổi đáng kể trên sự tồn tại của phát triển các yêu cầu không được biết tới được tháo gỡ và sự hiệu chỉnh các giải pháp thiết kế. Tổng này còn bao gồm cả phần mềm sử dụng lại nó là một phần của sản phẩm phân phối và đối tượng thay đổi bởi nhóm phát triển.
- Cấu hình SLOC (SLOCC). Những số liệu thống kê này theo dõi sự chuyển đổi của các thành phần phần mềm từ trạng thái thiết kế hoàn chỉnh đến một cấu hình điều khiển. Đối với bất cứ dự án nào được đưa ra, số liệu thống kê này sẽ cung cấp một cách thấu đáo về sự phát triển và khả năng ổn định của các nhóm phát triển. Với các phần mềm sử dụng lại sẽ là một đóng góp từ sớm cho SLOCC và vì sự phát triển tức thời và các ma trận chất lượng.

Những định nghĩa về số liệu thống kê tĩnh

Số liệu thống kê tĩnh	Định nghĩa
_ Tổng số SLOC	_SLOCT=tổng số SLOC
_Cấu hình SLOC	_SLOCC=đường biên hiện tạiSLOC
_Các lỗi nghiêm trọng	SCO_0 = số lượng loại 0 SCOs
_Các lỗi thường	SCO_1 = số lượng loại 1 SCOs
_Các phát triển	SCO_2 = số lượng loại 2 SCOs
_Các đặc trưng mới	SCO_3 = số lượng loại 3 SCOs
_Số lượng SCOS	$N=SCO_0+SCO_1+SCO_2$
_Tái tạo mở (dừng)	B=ngắt quãng thay đổi dồn đống theo SCO_0 ,SCO_1 và SCO_2
_Tái tạo đóng (dồn)	F=khôi phục đống SLOC
_Nỗ lực tái tạo	E=Nỗ lực tập trung dùng SCO_0, SCO_1, và SCO_2
_Thời gian sử dụng	UT=giờ mà đường biên được đưa ra đã được thao tác trong hoàn cảnh sử dụng thực tại.

- Các nhược điểm (SCO0 và SCO1). Các thay đổi giải quyết lại các lỗi phần mềm cấu thành số liệu thống kê quan trọng từ nó độ tin cậy và hoàn thiện của đường biên có thể được bắt nguồn. Mong muốn tác động cao nhất vào việc phát hiện lỗi xảy ra tức thì sau khi chứng nhận và làm giảm thời gian như các kỳ hạn phần mềm .
- Sự tiến bộ (SCO2). Một kích thích khác làm thay đổi đường biên, sự tiến bộ cũng là chìa khoá để đánh giá chất lượng và tiến độ chất lượng sản phẩm sau này .Sự mong muốn phát triển tỉ lệ nghịch với nhược điểm bởi vì tỉ lệ nhược điểm bắt đầu cao lên và cản trở, những phát triển bắt đầu giảm xuống và tăng thêm. Hiện tượng này mất đi dựa trên giả thuyết rằng nhóm phục hồi đang thực hiện kiểm tra và hoạt động bảo quản được nắm bắt bởi quan hệ sau.

  Nỗ lực(nhược điểm)+Kết quả(Sự phát triển)=hằng số

  Sự khác nhau giữa các nhược điểm và những tiến bộ là do một số sự chủ quan nào đó.

Các ma trận đã định nghĩa trong sách này chúng không mang tính nhạy cảm đặc biệt đối với các loại khác bởi vì chúng lệ thuộc vào tổng số các tác động từ cả 2 loại này. tuy nhiên sự khác nhau giữa các hư hỏng và hoàn thiện có thể có một tác động cần thiết dựa trên những đo lường được mô tả trong phần C-2-2.

- Các đặc trưng mới (SCO3) : Loại 3 phản ánh những thay đổi cập nhật các mong muốn của người quản lý ngăn xếp đối với các đặc trưng mới hoặc khả năng mới bên ngoài phạm vi cho phép hiện tại. Các số liệu thống kê loại 3 được phân tích một cách riêng biệt bởi vì chúng phản ánh được công việc mới hơn là sự tái tạo lại .
- Số lượng SCOs (N) . Bởi vì một SCO là một thành phần riêng rẽ của thay đổi, nó là một điều quan trọng cho việc định nghĩa chúng phù hợp thông qua tất cả những miền mà ở đó các ma trận được so sánh. Vậy thì cấp độ thay đổi nào được chứng minh bằng tư liệu và được theo dõi?. Hầu hết các định nghĩa đều tập trung tới một định nghĩa tương đối mơ hồ của một SCO dựa trên kích thước, tác động

bề ngang vào cá cá nhân và các nhóm và sự trao đổi CCB. Sự tiếp cận mơ hồ này sẽ làm cho các dự án mang tính riêng rẽ nhưng nếu tất cả các dự án sử dụng một định nghĩa khác, có khả năng so sánh qua các dự án thì chấp nhận được. Nhìn chung, SCOs sẽ tác động đến một thành phần đơn lẻ và sẽ được phân chia thành một thành phần đơn lẻ hoặc một nhóm chính. Với tiêu chuẩn đơn giản này, các định nghĩa chính xác hơn của những nguyên mẫu này là không cần thiết . Sự tinh luyện tác động tớ những nguyên mẫu chưa hoàn tất , và hoàn tất hơn làm tăng thêm giá trị nhỏ không đáng kể. Như vậy sự sưu tầm các ma trận ngày càng được hỗ trợ bởi các công cụ tự động hoá, sẽ có sự nhất quán hơn vè toàn bộ các kĩ thuật đo lường và các thành phần nguyên thuỷ.

- Tái tạo mở (B). Theo lý thuyết tất cả tái tạo đều nhằm làm tăng chất lượng. Sự tái tạo là cần thiết hoặc để chuyển đổi một trường hợp chất lượng tồi (SCO0, và SCO1) hoặc tác động tới một thành phần để làm tăng hiệu lực chi phí vòng đời (SCO2). Để đánh giá chính xác các xu hướng chất lượng của vòng đời, các tác động tái tạo phải có giá trị trong pha ngữ cảnh của vòng đời. Một tổng thể tái tạo nào đó là cần thiết đối với nỗ lực công nghệ phần mềm lớn: Tái tạo gần với thời gian khởi đầu được coi là dấu hiệu của tiến độ trong một mẫu tiến trình hiênj đại. Tái tạo tiếp theo, tái tạo cuối cùng, hoặc tái tạo không do thiếu đường biên cấu hình nhìn chung là cho ra chất lượng kém. Sự giải thích về các số liệu thống kê này đòi hỏi ngữ cảnh dự án. Tuy nhiên ,nhìn chung về cơ bản sẽ tiệm cận tới không trong việc phân phối sản phẩm. Để cung cấp một tiến trình sưu tầm thích hợp mà nó có thể được tự động hoá, tái tạo có thể được định nghĩa như số hiệu đánh giá SLOC để thay đổi do một SCO. Sự chính xác tuyệt đối của một đánh giá nhìn chung là không quan trọng. Bởi vì tái tạo mở được theo dõi một cách riêng biệt với thực tế, các giá trị tiếp nối chính xác và thích hợp.
- Tái tạo đóng (F). Trong khi các số liệu thống kê ngắt quãng đánh giá sự thiệt hại, sự hiệu chỉnh các số liệu thống kê cho biết thiệt hại thực tế đã được chỉnh sửa. Giải pháp, sự đánh giá nhận định chỗ ngắt tương ứng được cập nhật để phản ánh thực tế yêu cầu hiệu chỉnh các vấn đề còn lại trong đường biên. Mặc dù thực tế SLOC được hiệu chỉnh (F) sẽ không bao giờ chính xác tuyệt đối, chính xác ra nó liên quan tới việc đánh giá các xu hướng. Bởi vì “Hiệu chỉnh ” có thể mang một số nghĩa khác nhau dựa vào những cái được thêm vào, xoá đi, thay đổi , một tập các nguyên tắc thích hợp là cần thiết. SLOC thay đổi sẽ làm tăng B và F mà không làm thay đổi SLOCC. Mã được thêm làm tăng B, F và SLOCC, mặc dù không có phần giống nhau. Mã bị xoá (vấn đề xảy ra thường xuyên) không có sự tăng thêm tương ứng có thể làm tăng B và giảm SLOCC. Việc đưa ra các số lượng thay đổi và sự cần thiết duy nhất cho dữ liệu chính xác một cách tương đối với các xu hướng đồng nhất, sự chính xác của các dữ liệu thô liên quan là không quan trọng.
- Nỗ lực tái tạo (E). Sự nỗ lực tuyệt đối dùng toàn bộ cho việc làm sáng tỏ SCOS là một kĩ thuật cần thiết khác truy tìm sự phức tạp của tái tạo. Các hoạt động sẽ bị hạn chế với các yêu cầu kĩ thuật, công nghệ phần mềm, thiết kế, sự phát triển, và kiểm tra chức năng. Các hệ thống công nghệ cấp cao hơn, sự quản lý, mô hình điều khiển kiểm tra xác minh và kiểm tra hệ thống sẽ được thực hiện, bởi vì các hoạt động này hướng tới nhiều hơn mộtm chức năng của công ty, khách hàng, các thuộc tính dự án, sự độc lập của chất lượng. Mục đích ở đây là bình thường hoá sự mở rộng các hoạt động quan liêu khác ra khỏi ma trận.
- Thời gian sử dụng

Số liệu thống kê quan trọng thích hợp với tổng số giờ mà một đường biên đưa ra được thực hiện trong viễn cảnh sử dụng thực tế. Đối với một số hệ thống sso liệu thống kê này tương ứng với các phương pháp đo lường thời gia thực cho rất nhiều vấn đề khác, tự động kiểm tra có thể mô tả bằng cách là mỗi ngày kiểm tra một giờ. Ví dụ: Hầu hết việc thực hiện các hệ thống xử lý có một tải trọng trung bình mong

muốn được xử lý hàng ngày. Nếu tải trọng trung bình này có thể được kiểm tra trong lúc đóng gói và thực hiện trái lại với đường biên sản xuất trong một giờ, nó sẽ được tính bằng 24 giờ sử dụng. Một ví dụ khác xét đến công cụ phát triển được sử dụng bởi những thao tác với tốc độ là một số phím trên giây. Nếu tự động kiểm tra GUI các công cụ có thể hỗ trợ các ảnh hưởng lẫn nhau giữa các tập lệnh mà có thể đối lập được kiểm tra sản phẩm ở tốc cao hơn 10 lần sau đó tất cả các giờ kiểm tra tính bằng 10 giờ thời gian sử dụng. Định nghĩa thời gian kiểm trản vẽ để sử dụng thời gian nhìn chung là dễ hiểu. Đây cũng là bài tập phân tích các yêu cầu lớn thường xuyên nhân thấy không rõ ràng trong các hoàn cảnh sử dụng giữa những người quản lý ngăn xếp.

C-2-2 Các ma trận chất lượng sản phẩm cuối

Các ma trận chất lượng sản phẩm cuối cung cấp sự hiểu biết về bảo trì các sản phẩm phần mềm với các đặc trưng của loại 0,1, và 2 SCOs. Loại 3 SCOs không nằm trong nhóm đó bởi vì họ định nghĩa lại chất lượng cuối cùng cố hữu của hệ thống và khuynh hướng để yêu cầu thêm hệ thống và các công nghệ phần mềm cũng như một số yếu tố chủ yếu của các yêu cầu mức hệ thống. Bởi vì những loại thay đổi này được giải quyết trong các cách cực kỳ khác nhau bởi các dự án và những khách hàng khác nhau. Họ có khuynh hướng làm rối thêm và khả năng so sánh của dữ liệu. Dữ liệu các ma trận sẽ rất có ích trong việc đưa ra quyết định và lên kế hoạch toàn bộ nỗ lực quan trọng cần thiết để thực hiện loại 3 SCOs. Chúng hữu dụng khi ứng dụng lại các bộ sản phẩm như các thành phần hoặc cácn loại bỏ. Từ *sản phẩm* được sử dụng như là thứ cơ bản của những thứ được đo lường.

Bảng C-2 . *Các ma trận chất lượng sản phẩm cuối*

Ma trận	Định nghĩa
_Tỷ lệ đoạn cắt	B/SLOCT, phần trăm của sản phẩm bị tách
_Tỷ lệ tái tạo	E/nỗ lực phát triển , phần trăm nỗ lực tái tạo
_Lắp giáp	B/N đoạn cắt trung bình trên SCO
_Khả năng thích nghi	E/N , nỗ lực trung bình trên SCO
_Kỳ hạn	UT/(SCO_0+SCO_1) , thời gian ý nghĩa giữa các lỗi
_Khả năng bảo quản	(Tỷ lệ đoạn cắt)/(tỷ lệ tái tạo), hiệu quả bảo quản

- Tỉ lệ đoạn cắt. Ma trận này cung cấp giả trị so sánh các dự án mang tính lịch sử, các phát triển tương lai, hoặc các dự án tương lai. Nó định nghĩa tỉ lệ phần trăm của sản phẩm mà các sản phẩm này được tái tạo trong vòng đời.
- Tỉ lệ tái tạo. Giá trị này xác định tỉ lệ phần trăm của nỗ lực dùng cho tái tạo được so sánh với nỗ lực tổng thể . Nó có thể cung cấp chỉ dẫn tốt nhất của hiệu năng tái tạo (hay bảo quản).
- Sự lắp ghép. Giá trị này xác định tổng số lượng trung bình của SLOC bị cắt trên SCO, nó phản ánh khả năng cố hữu của sản phẩm tích hợp nhằm cục bộ hoá ảnh hưởng thay đổi: Với khả năng mở rộng cực đại có thể, CCBs sẽ bảo đảm rằng SCOs sẽ được soạn thảo cho các thay đổi nguồn đơn và được ứng dụng qua dự án.
- Khả năng thích nghi. Giá trị này cung cấp sự hiểu biết ban đầu mà với nó sản phẩm có thể thay đổi. Trong khi số lượng các thay đổi nhỏ nhìn chung là sự chỉ dẫn tốt của một tiến trình chất lượng. Độ lớn của nỗ lực trên thay đổi thì luôn luôn qcó vai trò quan trọng hơn.
- Kỳ hạn. Giá trị này cung cấp một chỉ dẫn về thời gian có ý nghĩa hiện thời giữa những hư hỏng (MTBF) đối với sản phẩm. Trong khi mục đích cuối cùng của kỳ hạn luôn luôn là không xác

định (Đặt tên, không có sai sót), tất cả các dự án không ổn định lắm. Mỗi một sản phẩm được đưa ra cho một nhóm người sử dụng, MTBF nhìn chung được khôi phục và ổn định. Xuyên xuốt sự phát triển vòng đời, tuy nhiên hoạt động bảo trì mong muồn làm tăng thêm kỳ hạn tồn tại của một thành phần loại bó đơn lẻ, và các xu hướng đã đưa ra sẽ cho biết mục đích cuối cùng phát triển của dự án đối với kỳ hạn.

- Khả năng bảo quản. Giá trị này xác định mối quan hệ giữa chi phí phát triển và chi phí phát triển. Nó cung cấp tiêu chuẩn hoá hợp lý cho các so sánh giữa các dự án. Bởi vì nhóm bảo quản các mục của nỗ lực và mẫu số của nó là ở trong các mục của SLOC, nó là tỉ lệ của các hiệu suất (nỗ lực trên SLOC). Một bố trí tính toán đơn giản sx chỉ ra rằng bảo quản (hoặc chất lượng bảo quản ,QM) là tương đương với công thức sau đây:

QM=Năng suất $_{\text{bảo quản}}$ /Năng suất $_{\text{phát triển}}$

Ví dụ , nếu (tỉ lệ đoạn cắt)=(tỉ lệ tái tạo) , năng suất biến đổi tương đương với năng suất phát triển và QM=1. Bằng trực giác giá trị 1 hể hiện mức "nghèo" bảo quản bởi vì nó sẽ dễ dàng tha đổi phần mềm đang tồn tại hơn là phát triển luân phiên từ điểm xuất phát. Thực tế các dự án truyền thống có khuynh hướng sử dụng tất cả 2$ vào bảo trì 1$ cho phát triển (Bochm 1987) có thể đáp ứng như là điểm chuẩn của những cái sẽ đạt tới mức "tốt" bảo quản. Hãy xem xét một đường lối quản lý phần mềm với tuổi thọ trung bình là 165 và tỉ lệ đóng gói hàng năm trung bình là 12%. Nếu QM=1 sẽ có tỉ lệ khoảng 1:2 giữa những phí tổn phát triển và phí tổn bảo quản hoặc khả năng bảo trì là nguyên tắc công nghệ phần mềm tương đối. Một giá trị bảo quản sẽ nhỏ hơn so với 1, trong hầu hết các trường hợp, cho biết mốt bảo trì ở mức độ cao, ít nhất đối với chi phí phát triển và các kinh nghiệm truyền thống.

Sự định nghĩa các mô tả này được lý tưởng hoá các xu hướng cho những ma trận này. Các hoàn cảnh dự án thực tế sẽ không bao giờ kết thúc .Nó quan trọng, tuy nhiên đối với những người quản lý ngăn xếp để hiểu quy mô mà nó thay đổi các ma trận từ ý tưởng. ứng dụng của những ma trận này qua các tiến triển dự án sẽ có ích cho toàn bộ dự án và các so sánh với các dự án khác.

C.2.3 Các chỉ dẫn tiến hành

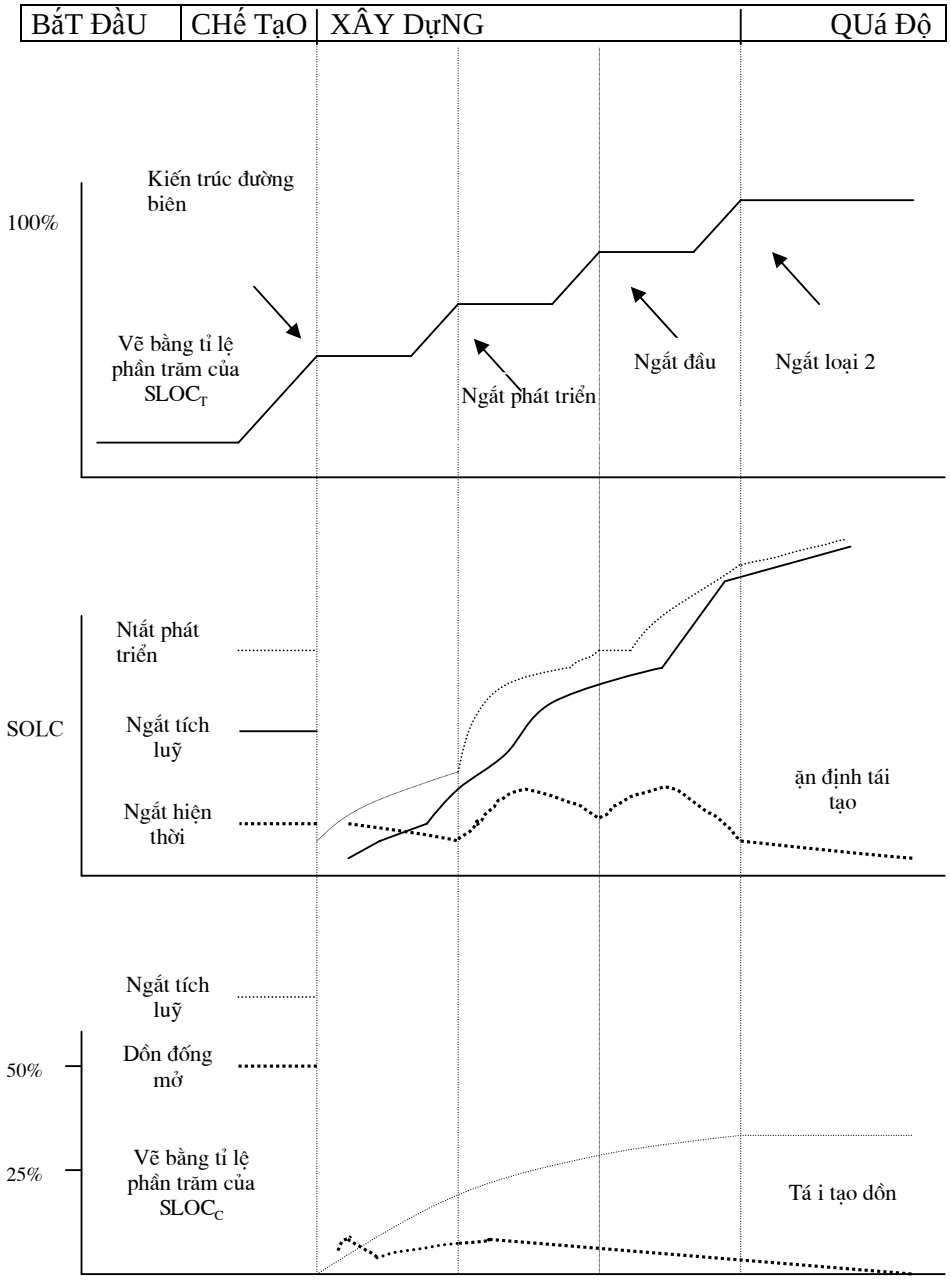
Các chỉ dẫn tiến hành được định nghĩa trong bảng C-3. Các dự tính tương đối được mô tả tiếp theo và được minh hoạ trong hình C-1 và C-2.

- Độ ổn định tái tạo. Các ma trận này xác định số lượng khác nhau giữa tái tạo tổng thể và tái tạo đống. Sự quan trọng của nó là để chỉ ra rằng có hay không sự đánh giá giải pháp đang duy trì tốc ngắt. Hình C-1 cho thấy 1 ví dụ về dự án sức khoẻ trong đó đánh giá giải pháp không làm lệch hướng (chấp nhận trong 1 khoảng thời gian ngắn) tốc độ ngắt. Tốc độ ngắt cũng được theo dõi quan hệ với tốc độ phân phối SLOCC , Bởi vì mức độ cố gắng dành cho kiểm tra và các thay đổi bảo quản trên vòng đời. Đây là mục đích của ma trận tiếp theo.

Bảng C-3 Những định nghĩa các chỉ dẫn tiến hành

Chỉ dẫn	Định nghĩa
ổn định tái tạo	B-F, ngắt trừ đi cố định trong suốt thời gian
Tái tạo đống	(B-F)/SLOCC,tái tạo mở hiện thời
Xu hướng lắp ráp	Sơ đồ ghép nối trong suốt thời gian
Xu hướng thích nghi	Sơ đồ thích ứng trong suốt thời gian
Xu hướng kỳ hạn	Sơ đồ giới hạn trong suốt thời gian

- Tái tạo đống. Tái tạo đống là tỷ lệ phần trăm của đường biên sản phẩm tồn tại, SLOCc , mà hiện thời cần thiết phải sửa chữa. Nhìn chung, sự dồn đống sẽ làm tăng một số thiết lập theo cấp của đường biên quản lý ban đầu, hoặc kiểm tra khám phá những thay đổi cần thiết, sự tái tạo đống sẽ liên quan đến tính ổn định chủ yếu thông qua chương trình kiểm tra cho đến khi nó trở về không. Những thay đổi lớn hoặc giữ vững được sự tăng trưởng hay tạo đống, từ tháng này sang tháng khác sẽ được nghiên cứu một cách cẩn thận. Giữ vững được sự tăng trưởng có thể chỉ ra được sự thiếu ổn định và sai lệch của kế hoạch.



Hình C-1 . Những xu hướng cho các chỉ dẫn tiến hành

bắt đầu chế tạo xây dựng quá độ chế tạo xây dựng quá độ	chế tạo	xây dựng	quá độ

- Xu hướng lắp ráp. Những thay đổi giá trị này cho biết sự mở rộng thay đổi đang thực hiện trên vòng đời như thế nào?. Xu hướng chung cung cấp sự hiểu biết sâu sắc về chất lượng (kiến trúc phù hợp, cục bộ hoá thay đổi tốt như thế nào) và sự quản lý (kế hoạch tập trung và những rủi ro thay đổi theo dòng). Hầu hết các thay đổi không đáng kể được nắm bắt và thực hiện các hoạt động kiểm tra đơn lẻ. Những giá trị địa chỉ thay đổi đang tiến dần tới đường biên cấu hình. Trong khi khó có thể xác định cách hình thành nào là một xu hướng tốt, nguyên tắc ra hiệu sau là loại mà các dự án đã thành công: SCO trung bình sẽ tác động tương đương đến đơn vị chương trình đơn (Mức độ thấp nhất của các khả năng biên dịch các hành phần mã. Ví dụ: Sự ngắt quãng trung bình trên SCO đối với phần mềm được viết bằng C++, C. Trong đó đơn vị chương trình trung bình có khoảng 50 dòng lệnh) sẽ chiếm 50 ở sự hoàn thành đề án. Trong tiến trình phát triển lặp lại có kỳ hạn, các thay đổi gần (những thay đổi thiết kế tác động đến đa thành phần và con người) được mong đợi để yêu cầu thêm sự tái tạo những thay đổi cuối cùng (Thay đổi yếu tố, có khuynh hướng bị hạn chế bởi một cá nhân hoặc thành phần đơn lẻ). Các xu hướng lắp ghép đang phát triển cùng thời gian chỉ ra một cách rõ ràng kiến trúc sản phẩm đang phân hoá.
- Xu hướng thích nghi. Giá trị này cung cấp quy trình các xu hướng đánh giá độ phức tạp thay đổi, như là tính chống đối với sự mở rộng thay đổi, do đó chất lượng sẽ tăng lên. Với tiến trình truyền thống, nó sẽ đắt hơn để thay đổi các thành phần cuối cùng trong vòng đời. Trong một tiến trình lặp lại hiện đại, đối tượng thiết lập một tiến trình và một kiến trúc khắt khe đến nỗi mà các thay đổi được dễ dàng hơn và kết quả có thể dự đoán trước, lâu hơn trong vòng đời. Các xu hướng tái tạo chỉ có thể ổn định hơn để đạt được sự đơn giản hơn trong suốt thời gian. Không bao giờ có sự khác nhau quan trọng từ tiến trình truyền thống. Một xu hướng tốt thì khó khăn cho việc chuẩn hoá các thuật ngữ chính xác. Trong thực tế các dự án thành công có hướng từ kinh nghiệm một chi phí trung bình thay đổi ít hơn một tuần làm việc.
- Xu hướng kỳ hạn. Thật dễ dàng để giải thích mong muốn về giá trị này cho một ngắt đơn. Tuy nhiên hầu hết các dự án phần mềm hiện đại bao gồm một số sự lặp lại và tăng trưởng với các hoạt động trùng lặp toàn phần và các phương án ngắt. Đánh giá thời hạn của của toàn bộ một hệ thống thì phức tạp hơn đánh giá kỳ hạn của ngắt đưa ra. Sự định trước đối với mỗi ngắt đơn sẽ có liên quan đến sản phẩm không có kỳ hạn (các lỗi gặp phải mang tính chu kỳ) nó cho biết sự không tăng thêm kỳ hạn như đã được ấn định là thích hợp trong các cập nhật bảo quản. Sự kỳ vọng đối với một dự án đơn giản được minh hoạ trong hình C-2 cho thấy răng với mỗi ngắt kế tiếp, toàn bộ các đường biên gặp các sai sót ít hơn và thêm được thời gian sử dụng. Sự phat triển theo hàm mũ được đưa ra trong hình, có thể làm mất

tính thực tế cho hầu hêt cá hệ thống. Sự phát triển tuyến tính có thể có tính thực tế hơn”Lợi thế của bạn có thẻ biến đổi “ hnưng tiến trinh mạnh mẽ không trải qua một sự giảm bớt kỳ hạn chấp nhận được và thuật ngữ ngắn gọn các suy giảm sẽ có một lý do rõ ràng.

C-3 Các ma trận thay đổi thực dụng

Phần 13.1 miêu tả một số mục đích của chương trình các ma trận hữu ích. Các mục đích nay được nói lại ở phần tiếp theo và được thảo luận trong các phần mà các ma trận cho thấy được những mục đích này.

- Các ma trận phải đơn giản, khách quan, dễ sưu tầm, dễ thực hiện, và khó suy luận sai. Số lượng các thông số thống kê được quản lý trong một cơ sở dữ liệu SCO đẻ tiếp cận được là nhỏ, nhỏ hơn 10. Chúng đơn giản đẻ tính toán và có các định nghĩa đơn giản, mặc dù tronmg thực tế rất nhiều thành phần của tổng này là không rõ ràng. Dựa vào kỷ luật, tính kiên định và mức tự đọng vốn có trong một tiến trình cố hữu. Việc định nghĩa và sưu tầm những ma trận này có thể được tiến hành dể dàng. Nói khác đi, một tổ shức có nhu cầu đặc biệt với các dự án phần mềm theo khuynh hướng khác nhau có thể tìm thấy sự khó khăn cho sự tập trung về các thực nghiệm có thể chấp nhận được. Các triển vọng được cung cấp bởi các ma trận này có sự sáng tỏ không phức tạp trong hầu hết các trường hợp. Hầu hết các xu hướng biểu thị là xấu hoặc tốt. Hầu hết các giá trị này là độc lập ngữ cảnh nhưng với dữ liệu từ đa dự án trong ngữ cảnh thông thường, nó dể dàng để giải thích những sự giống và khác nhau.
- Sự sưu tầm các ma trận phải được tự động và không có tính xâm phạm. Đó là không làm cản trở các hoạt động của dự án. Tất cả dữ liệu được sưu tầm và phân tích yêu cầu trong cách tiếp cận các ma trận này là có thể được, đã được, tự động hoá. Trong khi các công nghệ đơn giản theo các dòng công việc đối với các mẫu sinh ra, mô hình điều khiển hệ thống có thể bị xâm phạm để sưu tầm và tiến hành tất cả các dữ liệu yêu câù khai thác các ma trận và các xu hướng.
- Các ma trận phải cung cấp các đánh giá mang tính kiên định xuyên suốt vòng đời, đặc biệt là trong các pha đầu, khi các nỗ lực để tăng chất lượng có sự trả công thoả đáng. Cách tiếp cận được mô tả ở đây được bắt nguồn từ triển vọng bảo quản phần mềm. Tuy nhiên, một tiến trình phát triển có tính tương tác có thể được thấy như một sự pha trộn các hoạt động phát triển và bảo quản thành một tập vòng đời phjổ biến hơn các hoạt động mà nó sử dụng các công cụ và các kĩ thuật như nhau. Từ triển vọng này, một cách tiếp cận có tính tương tác có thể coi là sự tăng tốc đơn giản thiết lập các đường biên để đường biên thay đổi, và tiến độ vốn có của chúng và sự hiểu biét sâu sắc về chất lượng, có thể sử dụng các công cụ tiến hành tốt hơn. Với các công nghệ mang tính truyền thống, điều này được cho ở sổ tay, hoạt động có khuynh hướng lỗi. Với thuận lợi tự động quản lý thay đổi hiện nay và công nghệ hành trình vòng hỗ trợ giữa các tiêu chuẩn công nghệ khác nhau, tự do thay đổi được tăng lên và sự chuyển tới một tiến trình có tính tương tác là kĩ thuật có tính khả thi và lợi nhuận kinh tế.
- Các ma trận, các giá trị và những xu hướng phải được hoạt động bởi cá nhân quản lý và người làm công nghệ về phát triển viễn thông và chất lượng trong sự địng dạng thích hợp. Những ma trận này thích hợp với các đo lường xác thực của các mẫu phần mềm đưa ra. Chúng bắt nguồn trực tiếp từ các đường biên tiến hoá của sản phẩm, không phải từ tài liệu riêng hay các suy đoán chủ quan. Các công nghệ phần mềm sẽ chấp nhận và sử dụng các ma trận khách quan này để tránh những kĩ thuật kém và các quyết định sai. Khi các giám đốc bị thuyết phục họ sẽ thích nghi đối với bất kỳ đo lương khách quan nào. Việc đưa ra những ma

trận này là đơn giản. Hầu hết những người nắm giữ ngăn xếp có thể hiểu chúng, các ma trận có thể tự động, và có thể được so sánh từ các dự án khác nếu sử dụng đúng đắn.