

Tác giả: Karl E. Wiegers

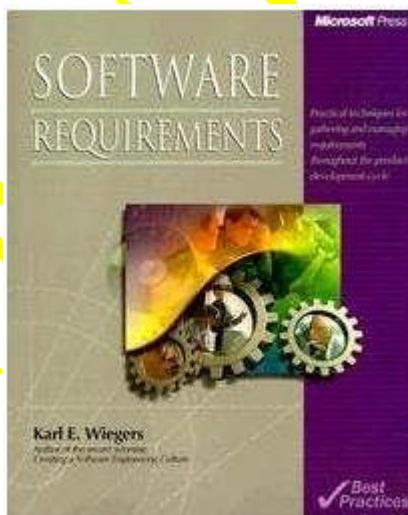
Software Requirements Best Practices

Các kỹ thuật thực hành để thu thập và quản lý yêu cầu trên
tổn bộ chu trình phát triển sản phẩm phần mềm

Microsoft Press, 1st Edition

Người dịch: Hoàng Xuân Thịnh

Phiên bản bản dịch: 10.12.30



GIỚI THIỆU

“TỦ SÁCH CÔNG NGHỆ THÔNG TIN”

Để phát triển, ngành công nghiệp công nghệ thông tin Việt Nam phải hướng ra thị trường thế giới, do đó phải tuân theo các chuẩn mực toàn cầu như làm việc theo quy trình, áp dụng các tiêu chuẩn quản lý chất lượng sản phẩm và dịch vụ phổ biến (ISO 27000, CMMI...), áp dụng các hướng dẫn thực hành tốt và tốt nhất. Vì vậy, đối với các sinh viên đang theo học ngành công nghệ thông tin và chuẩn bị ra làm việc, chúng tôi nghĩ rằng, các cuốn sách hướng dẫn kỹ thuật, hướng dẫn xây dựng quy trình làm việc, hướng dẫn cách tổ chức công việc nhằm nâng cao năng suất lao động có vai trò hết sức quan trọng. Những cuốn sách này giúp người đọc nhận thức về các chuẩn mực công nghiệp trong ngành công nghệ thông tin thế giới, học hỏi về cách những đồng nghiệp trên toàn cầu của họ làm việc như thế nào, qua đó người đọc có thể sẽ thay đổi suy nghĩ của mình sao cho gần hơn với các chuẩn mực và cách làm việc đó. Sự thay đổi trong nhận thức theo chiều hướng này càng diễn ra sâu rộng bao nhiêu thì càng thúc đẩy công nghiệp công nghệ thông tin Việt Nam phát triển bấy nhiêu.

Để đáp ứng nhu cầu sách tham khảo theo chiều hướng đó, chúng tôi xây dựng “Tủ sách công nghệ thông tin” bằng cách tuyển chọn và giới thiệu các bản dịch tiếng Việt của các cuốn sách có nội dung đáp ứng các tiêu chí sau:

1. Có thể làm tài liệu tham khảo cho sinh viên các khoa công nghệ thông tin để họ biết thêm về thực tiễn ngành công nghiệp công nghệ thông tin thế giới.
2. Hướng dẫn xây dựng quy trình làm việc, quản lý chất lượng sản phẩm và dịch vụ một cách thực tiễn, không hàn lâm, có thể ứng dụng được ngay vào thực tế công việc hàng ngày của mỗi người làm việc trong ngành công nghệ thông tin.
3. Giới thiệu các “hướng dẫn thực hành tốt” (good practices) và “hướng dẫn thực hành tốt nhất” (best practices) trong công nghiệp công nghệ thông tin.
4. Giới thiệu các xu hướng công nghệ mới trong ngành công nghệ thông tin thế giới.

Nói gọn lại, thông qua “Tủ sách công nghệ thông tin”, chúng tôi muốn góp phần cỗ vũ xây dựng một nền “VĂN HÓA CÔNG NGHIỆP” trong ngành công nghệ thông tin của Đất nước chúng ta, điều hết sức cần thiết để Việt Nam có một ngành công nghiệp công nghệ thông tin phát triển và hiện đại, đóng góp cho sự giàu mạnh của Đất nước.

Cuốn sách đầu tiên trong “Tủ sách công nghệ thông tin” là cuốn “Software Requirement Best Practices” (Các hướng dẫn thực hành tốt nhất về yêu cầu phần mềm) do Microsoft Press xuất bản. Đây là cuốn sách hết sức cần thiết cho những ai đang thực hiện các dự án công nghệ thông tin và các sinh viên ngành công nghệ thông tin muốn có thêm hiểu biết về thực tiễn ngành trước khi ra làm việc.

Xin trân trọng giới thiệu cùng bạn đọc.

SATA-APTECH

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

LỜI GIỚI THIỆU CỦA NGƯỜI DỊCH

Trong những năm qua tôi đã tham gia một số dự án CNTT và tôi thấy rất thiếu những tài liệu có giá trị bằng tiếng Việt để tham khảo về cách thức tiến hành các dự án này. Tài liệu nước ngoài thì rất nhiều và khá nhiều tài liệu hay, nhưng tài liệu hướng dẫn chi tiết thành một quy trình làm việc thì cũng không nhiều. Cách đây mấy năm, trong một chuyến công tác, tôi mua được cuốn sách *Software Requirements* của tác giả Karl E. Wiegers do Microsoft Press ấn hành. Cuốn này hiện đã có bản mới cũng của Microsoft Press. Trong cuốn sách này tác giả trình bày toàn bộ một *quy trình biến nhu cầu sử dụng phần mềm của khách hàng thành một bản đặc tả yêu cầu phần mềm*. Bản đặc tả yêu cầu phần mềm này sẽ trở thành đầu vào cho quy trình phát triển, triển khai và bảo trì một sản phẩm phần mềm.

Trong **Quy trình lập kế hoạch chất lượng** trên satablog2, nhà tư vấn chất lượng Juran đã dành *Bước 2 - Định danh khách hàng* và *Bước 3 – Khám phá nhu cầu khách hàng* để mô tả cách thức hiện thức hóa nhu cầu của khách hàng thành một bản *đặc tả sản phẩm* – Juran nhấn mạnh đây là điều kiện cần để sản xuất được sản phẩm có chất lượng. Toàn bộ Bước 2 và Bước 3 trong quy trình trên được Karl E. Wiegers cụ thể hóa thành một cuốn sách hay, dễ đọc và thiết thực áp dụng cho việc sản xuất phần mềm.

Ở đây, tôi muốn nói tới sự phân biệt giữa thuật ngữ “nhu cầu” (need) của Juran và thuật ngữ “yêu cầu” của Wiegers. Để phân biệt giữa nhu cầu và yêu cầu tôi lấy ví dụ này. Cả người Việt Nam lẫn người Mỹ đều có nhu cầu như nhau là ăn nhanh, ăn ngon, ăn sạch vào bữa sáng. Nhưng do sự khác nhau về văn hóa mà cái nhu cầu ấy thể hiện ra bên ngoài thành các yêu cầu khác nhau, đối với người Mỹ thường là một bữa sáng với bánh mỳ nhanh của McDonald và một hộp Coca-Cola, đối với người Việt miền Bắc thường là một bát phở và một chén nước chè. Nhu cầu là cái bên trong được thể hiện ra bên ngoài thành những yêu cầu khác nhau như vậy. Juran đã viết khá kỹ về sự khác biệt này trong **Quy trình lập kế hoạch chất lượng**.

Chúc các bạn thu thập được các hướng dẫn thực hành thiết thực khi đọc cuốn sách này!

Hoàng Xuân Thịnh, 12/2010

Dành tặng Miss Chris

SATA-APTECH

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

LỜI NÓI ĐẦU

Mặc dù ngành công nghiệp phần mềm đã có trên năm mươi năm kinh nghiệm thực tế nhưng nhiều tổ chức phát triển phần mềm vẫn phải vật lộn với việc thu thập, tài liệu hoá, quản lý các yêu cầu đầu vào đối với sản phẩm của mình. Thiếu đầu vào từ người dùng, yêu cầu không đầy đủ và thay đổi yêu cầu là những lý do chính khiến các tổ chức phát triển phần mềm không giao được cho khách hàng sản phẩm phần mềm với các chức năng đã được khách hàng đưa vào kế hoạch sản xuất theo đúng lịch biểu và ngân sách đã định. Nhiều nhà phát triển phần mềm không cảm thấy hài lòng với việc thu thập yêu cầu từ khách hàng. Công nghệ yêu cầu (requirement engineering) trong thực tế không được phổ biến rộng rãi tới các nhà phát triển, trong trường học sinh viên cũng không được học các kỹ thuật của công nghệ yêu cầu. Thậm chí, ngay những người tham gia dự án phần mềm còn không thống nhất được với nhau nội dung của thuật ngữ “yêu cầu”.

Phát triển phần mềm cũng liên quan đến truyền thông (communication) (*y nói là sự truyền thông hay là giao tiếp giữa nhóm phát triển phần mềm và khách hàng mua phần mềm, từ đó nhóm phát triển mới hiểu khách hàng cần gì ở sản phẩm phần mềm sẽ xây dựng - ND*) nhiều như là liên quan đến tính toán (computing), tuy nhiên thường thì chúng ta hay nhấn mạnh đến khía cạnh tính toán mà bỏ quên truyền thông. Cuốn sách này cung cấp các công cụ thúc đẩy việc truyền thông và giúp các nhà phát triển và khách hàng của họ sử dụng hiệu quả các phương pháp của công nghệ yêu cầu. Cuốn sách đưa ra nhiều cách tiếp cận nhằm giúp nhóm dự án và khách hàng của dự án thoả thuận về những gì phần mềm cần đáp ứng để thoả mãn nhu cầu người dùng, cùng với cách thức để tài liệu hoá và quản lý các thay đổi trong các thoả thuận đó. Các kỹ thuật được giới thiệu ở đây thuộc loại “thực hành tốt” (good practices) của công nghệ yêu cầu, chúng không phải là các kỹ thuật “xa lạ” từ giới hàn lâm hoặc là một phương pháp luận khái quát để giải quyết bài toán yêu cầu của bạn.

LỢI ÍCH TỪ CUỐN SÁCH NÀY

Trong số các cải tiến quy trình phần mềm mà bạn có thể nắm bắt, các thực hành quản lý và phát triển yêu cầu được cải tiến sẽ cung cấp lợi ích lớn nhất cho bạn. Các khái niệm và phương pháp ở đây là độc lập với các phương pháp luận phát triển cụ thể, độc lập với các miền ứng dụng – dù bạn phát triển phần mềm viễn *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

thông hay tài chính bạn vẫn dùng những phương pháp này, bạn có thể sử dụng chúng trong một miền rộng lớn các loại dự án. Tôi tập trung mô tả một số kỹ thuật thực hành đã được chứng minh tính hiệu quả nhằm giúp bạn:

- Đạt được sự hài lòng cao hơn của khách hàng.
- Giảm chi phí bảo trì và hỗ trợ.
- Cải thiện chất lượng các yêu cầu trong dự án ngay từ sớm trong chu trình phát triển, giảm bớt các công việc phải làm lại và cải thiện năng suất.
- Đáp ứng các mục tiêu của lịch biểu bằng cách kiểm soát sự phá vỡ phạm vi ứng dụng của phần mềm và các thay đổi yêu cầu.

Mục đích của tôi là giúp bạn cải thiện quy trình bạn sử dụng để thu thập, phân tích yêu cầu, viết và kiểm tra đặc tả yêu cầu (requirement specification), quản lý yêu cầu trên toàn bộ chu trình phát triển sản phẩm. Cải tiến quy trình nhằm giúp nhóm dự án làm việc theo cách mới để tạo ra các kết quả tốt hơn. Vì vậy tôi hy vọng bạn sẽ thực hành những gì viết ở đây thay vì chỉ đọc chúng.

NGHIÊN CỨU TÌNH HUỐNG

Nhằm giúp bạn ứng dụng các phương pháp ở đây, tôi đã cung cấp các ví dụ là một số nghiên cứu tình huống từ các dự án hiện tại, một trong số đó là hệ thống IT có kích thước trung bình được gọi là Chemical Tracking System (Đứng lo lắng - Bạn không cần phải biết bất cứ thứ gì về hóa học để hiểu dự án này). Ví dụ này được thảo luận liên tục trong các tình huống khác nhau để bạn thấy các khía cạnh khác nhau của cùng một bài toán, các đoạn đối thoại giữa các thành viên nhóm dự án đó cũng được đưa ra nhằm minh họa bài toán.

AI NÊN ĐỌC CUỐN SÁCH NÀY?

Bất cứ ai có công việc liên quan đến yêu cầu trong một dự án phát triển mới hay nâng cấp một sản phẩm phần mềm đều có thể tìm thấy những thông tin hữu ích ở đây. Độc giả của cuốn sách có thể gồm: analysts (người phân tích), developers (người phát triển), testers (người kiểm thử) - những người phải hiểu và đáp ứng kỳ vọng của khách hàng. Một nhóm độc giả thứ hai là những người dùng muốn hình dung một sản phẩm phần mềm đáp ứng nhu cầu về chức năng và nhu cầu sử dụng của bản thân họ. Các khách hàng muốn đảm bảo rằng sản phẩm sẽ đáp ứng các nhu cầu kinh doanh của mình sẽ hiểu tốt hơn về bản chất và tầm quan trọng của quy. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

trình yêu cầu. Các nhà quản lý dự án phải đổi mới với việc bàn giao sản phẩm đúng thời hạn sẽ học được cách thức quản lý các thay đổi yêu cầu tiềm tàng.

KẾT CẤU CUỐN SÁCH

Cuốn sách được tổ chức thành 3 phần.

- Phần I bắt đầu bằng việc giới thiệu một số định nghĩa nền tảng về công nghệ yêu cầu và mô tả một số đặc tính của các yêu cầu tuyệt hảo.
- Phần II giới thiệu nhiều kỹ thuật phát triển yêu cầu, bắt đầu bằng định nghĩa yêu cầu nghiệp vụ, tầm nhìn (vision) và phạm vi.
- Phần III giới thiệu các nguyên lý và thực hành về quản lý yêu cầu.

TỪ NGUYÊN LÝ TÓI THỰC TẾ

Khó khăn biết bao khi tập trung năng lượng cần thiết nhằm vượt qua những trở ngại để thay đổi và để đưa các hiểu biết mới vào hoạt động thực tế. Con người và các tổ chức có xu hướng giữ nguyên, duy trì những gì đã có, dù rằng chúng không hiệu quả. Để giúp đỡ bạn trong hành trình cải tiến quy trình yêu cầu, mỗi chương sẽ có một mục gọi là “Các bước tiếp theo” – mô tả chi tiết các hành động cụ thể để bạn áp dụng các phương pháp được giới thiệu trong chương này vào thực tế. Tôi cung cấp các templates cho các tài liệu yêu cầu, các checklists, một bảng tính xếp thứ tự ưu tiên yêu cầu, và nhiều thứ nữa trên trang web của tôi tại <http://www.processimpact.com>. Hãy bắt đầu từ những thay đổi nhỏ trong công việc của bạn ngay ngày hôm nay.

Không phải tất cả những người tham gia dự án của bạn đều ủng hộ bạn trong những thay đổi này. Hãy sử dụng những hiểu biết thu thập được ở đây để thuyết phục họ, hãy lôi kéo họ cùng học và cùng cải tiến.

Bạn không cần phải khởi động một dự án mới để bắt đầu áp dụng những kỹ thuật của công nghệ yêu cầu. Một điểm bắt đầu tốt là hãy sử dụng một quy trình kiểm soát thay đổi thích hợp. Nghĩa là, bạn có thể mau chóng bắt đầu bằng việc quản lý các đề xuất thay đổi yêu cầu. Khi bạn thêm các tính năng mới vào sản phẩm đã có, hãy bắt đầu phân tích ảnh hưởng một cách hệ thống và tạo một ma trận lằn vết để liên kết các yêu cầu mới tới các bản thiết kế, mã nguồn và tình huống kiểm thử (test cases) tương ứng. Rất hiếm khi bạn quay lại và viết lại những bản đặc tả yêu. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

cầu mới cho một hệ thống đã có. Tuy nhiên, bạn có thể viết các yêu cầu cho phiên bản tiếp theo bằng một cách chặt chẽ hơn so với trước đây, viết các analysis models cho các tính năng mới, kiểm tra các yêu cầu mới. Thực hành dần các kỹ thuật của công nghệ yêu cầu là một cách tiếp cận cải tiến quy trình có độ rủi ro thấp, những kinh nghiệm thu được sẽ là nền tảng cho bạn chuẩn bị một dự án mới.

Mục tiêu của công nghệ yêu cầu là phát triển các yêu cầu chất lượng cao - chứ không phải hoàn hảo – các yêu cầu cho phép bạn xây dựng dự án với một mức độ rủi ro chấp nhận được. Bạn cần dành đủ thời gian cho giai đoạn làm yêu cầu để tối thiểu hóa rủi ro phải làm lại công việc, tạo ra sản phẩm không thể chấp nhận, lịch biểu bị tràn. Cuốn sách này giúp bạn xác định khi nào bạn đạt tới điểm có được các yêu cầu chất lượng và gợi ý một số cách để làm điều đó.

LỜI CẢM ƠN

Một số bạn hữu đã dành thời gian để xem xét các bản thảo và cho tôi những lời khuyên quý báu. Đặc biệt cảm ơn Kathy Rhode, người đã xem xét tỷ mỷ bản thảo và giúp tôi tư duy, trình bày vấn đề tốt hơn. Các bạn Chris Fahlbusch, Tammy Hoganson, Deependra Moitra, Mike Rebatzke...

Tôi cũng cảm ơn Steve McConnell, người đã khuyến khích tôi viết một cuốn sách về yêu cầu phần mềm và đã chuyển bản thảo cho biên tập viên Ben Ryan của Microsoft Press. Ben đã giúp tôi làm việc với các biên tập viên khác của nhà xuất bản. Mary Kalbach Barnard của Microsoft Press đã quản lý dự án này và cùng với sự giúp đỡ của Michelle Goodman, đã biên tập từ bản thảo đầu tiên đến bản thảo cuối cùng của cuốn sách.

Tôi rất biết ơn một số khách hàng mà tôi tư vấn, đặc biệt là Sandy Browning, Matt DeAthos, Kathy Rhode, Kathy Wallace, những người đã mời tôi tham gia làm việc cùng họ trong các quy trình yêu cầu.

Tôi cũng cảm ơn sự đóng góp từ hàng nghìn người tham gia các xêmina về yêu cầu phần mềm của tôi trong những năm qua. Là một nhà tư vấn và một nhà giáo dục, tôi đã học được rất nhiều từ mỗi công ty mà tôi làm việc, từ những người đã tham

gia các xêmina của tôi. Những gì hữu ích thu thập đều được tôi đưa vào cuốn sách này. Mọi thư từ trao đổi xin gửi cho tôi kwiegers@acm.org.

Tôi trân trọng sự đóng góp sâu sắc nhất cho cuốn sách này từ vợ của tôi - Chrish Zambito.

Xin mời bạn nghiên cứu cuốn sách và thực hành những gì bạn thu thập được. Chúc bạn thành công!

VỀ TÁC GIẢ KARL E. WIEGERS

Karl E. Wiegers là nhà tư vấn chính tại Process Impact, một công ty đào tạo và tư vấn quy trình phần mềm có trụ sở tại Portland, Oregon. Ông đã tư vấn và thuyết trình trong các xêmina tại hàng chục công ty vùng Bắc Mỹ. Trước đây, Karl đã làm việc 18 năm tại công ty Eastman Kodak, nơi ông làm công việc của một nhà khoa học nghiên cứu về ảnh, nhà phát triển phần mềm, nhà lãnh đạo quy trình phần mềm và cải tiến quy trình. Karl đã nhận bằng tốt nghiệp đại học ngành hóa học từ Boise State College, bằng cao học và tiến sĩ hóa hữu cơ của University of Illinois. Ông là thành viên của IEEE, IEEE Computer Society và Hiệp hội máy tính Mỹ (ACM).

Karl là tác giả của cuốn sách đoạt giải thưởng *Năng suất phát triển phần mềm* (*Software Development Productivity*) - cuốn *Tạo lập một nền văn hóa công nghệ phần mềm* (*Creating a Software Engineering Culture*) do Dorst House xuất bản năm 1996. Ông cũng là tác giả của hơn 150 bài báo về nhiều khía cạnh của khoa học máy tính, hóa học và lịch sử quân sự. Ông đồng thời là biên tập viên bán thời gian cho tạp chí *Software Development*, là một thành viên trong ban biên tập của tạp chí *IEEE Software*.

Khi không làm việc, ông chơi guitar với tay guitar Gibson Les Paul, đua trên chiếc mô tô Suzuki VX800 của mình, nghiên cứu lịch sử quân sự, nấu nướng và nhâm nhấp rượu vang với vợ và con mèo đen Gremlin của họ.

MỤC LỤC

PHẦN I – YÊU CẦU PHẦN MỀM: CÁI GÌ VÀ TẠI SAO

1. Cơ bản về yêu cầu phần mềm
2. Yêu cầu phần mềm từ quan điểm của khách hàng
3. Các thực hành hiệu quả cho công nghệ yêu cầu
4. Cải tiến quy trình yêu cầu của bạn
5. Yêu cầu phần mềm và quản lý rủi ro

PHẦN II – PHÁT TRIỂN YÊU CẦU PHẦN MỀM

6. Thiết lập tầm nhìn và phạm vi của dự án
7. Tìm kiếm tiếng nói của khách hàng
8. Lắng nghe tiếng nói của khách hàng
9. Tài liệu hóa yêu cầu
10. Một bức tranh đáng giá 1024 lời nói
11. Các thuộc tính của chất lượng phần mềm
12. Giảm rủi ro thông qua làm nguyên mẫu
13. Thiết lập các ưu tiên của yêu cầu
14. Kiểm tra chất lượng yêu cầu
15. Nhìn xa hơn việc phát triển yêu cầu

PHẦN III - QUẢN LÝ YÊU CẦU PHẦN MỀM

16. Các nguyên lý và thực hành quản lý yêu cầu
17. Quản lý đề nghị thay đổi
18. Các liên kết trong chuỗi yêu cầu
19. Công cụ cho quản lý yêu cầu

PHẦN I

YÊU CẦU PHẦN MỀM: CÁI GÌ VÀ TẠI SAO

SATA-APTECH

CHƯƠNG 1

CƠ BẢN VỀ YÊU CẦU PHẦN MỀM

“Xin chào, Phill? Tôi là Maria ở Bộ phận Nguồn nhân lực. Chúng tôi đang có vấn đề về hệ thống nhân lực (employee system) mà anh đã lập trình cho chúng tôi. Một nhân viên vừa mới thay đổi tên của cô ta thành Sparkle Starlight và chúng tôi không thể khiến cho hệ thống chấp nhận tên thay đổi này. Giúp tôi được không?”

“Cô ta vừa cưới một gã tên là Starlight à?”

“Không, cô ta không cưới chồng, chỉ thay đổi tên thôi,” Maria đáp. “Đó mới là vấn đề. Hệ thống chỉ chấp nhận ai đó thay đổi tên nếu họ thay đổi tình trạng hôn nhân của mình.”

“Ồ, tôi chưa bao giờ nghĩ rằng có ai đó lại thay đổi tên của mình. Tôi không thấy chị nói với tôi về việc này khi chúng ta trao đổi với nhau về hệ thống trước đây. Đó là lý do chị không thể nhấn vào hộp thoại Change Name nếu trước đó không nhấn vào hộp thoại Change Marital Status,” Phill nói.

Maria nói, “Tôi nghĩ anh biết mọi người đều có thể thay đổi tên của mình một cách hợp pháp bất cứ lúc nào nếu họ thích. Thêm nữa, chúng tôi phải khoá sổ vào thứ 6 tới nhưng Sparkle không thể thanh toán được hoá đơn của cô ấy. Có thể giúp tôi sửa lỗi (bug) này không?”

“Đó không phải là lỗi! Tôi không hề biết là chị cần tính năng này. Tôi bận vào vụ hệ thống đánh giá hiệu suất mới rồi. Tôi nghĩ tôi có một số đề nghị thay đổi khác cho hệ thống nhân lực đây,” [tiếng giấy sột soạt], “Đây rồi. Tôi có thể cố định chức năng khoá sổ vào cuối tháng chứ không phải cuối tuần. Xin lỗi nhé. Lần tới, hãy nói những thứ đó cho tôi sớm hơn và làm ơn ghi ra giấy.”

“Vậy tôi có thể giúp gì cho Sparkle đây”, Maria vặn hỏi, “Cô ấy không thể thực hiện công việc được nếu không thanh toán được hoá đơn.”

“Ồ, Maria, đó không phải lỗi của tôi.” Phill nói. “Nếu cô nói trước với tôi cái vụ thay đổi tên đó thì mọi việc đã xuôi chèo mát mái. Cô không thể xử lý tôi vì tôi không đọc được ý nghĩ của cô.”

Giận dữ Maria nặng lời, “Ồ, thê cơ đáy, đây chính là lý do làm tôi ghét máy tính. Hãy gọi lại tôi càng sớm càng tốt để sửa cái này.”

Nếu bạn đã từng trao đổi với khách hàng như trên thì bạn biết khách hàng sẽ rắc rối như thế nào nếu phải sử dụng các phần mềm không thể thực hiện được các tác vụ cơ bản (*tác vụ - task, được hiểu là các công việc trước đây là của người dùng những giờ được giao cho máy tính thực hiện, ví dụ tác vụ in hóa đơn, tác vụ lập báo cáo tài chính - ND*). Còn về phía bạn, bạn đã thấy mọi việc rối tinh ra sao nếu nhận được mong muốn của khách hàng đối với hệ thống sau khi hệ thống đã được cài đặt. Cũng thật bức bối nếu bạn phải sửa hệ thống ngay trong khi bạn đang xây dựng hệ thống đúng như những gì bạn đã được nghe từ lúc đầu.

Nhiều vấn đề nảy sinh trong khi phát triển phần mềm có thể có nguồn gốc từ quy trình và từ sự thực hiện các quy trình đó trong việc thu thập, tài liệu hóa, thỏa thuận và lựa chọn các yêu cầu phần mềm. Như câu truyện trên, miền thông tin thu thập có thể gồm cả những gì không được nói ra, không được ghi nhận, không được thỏa thuận.

Khi xây dựng một ngôi nhà, phần lớn trong số chúng ta đều trao đổi với nhà thầu về nhu cầu và mong muốn của chúng ta từ đó ước tính chi phí. Tuy vậy, phần lớn trong số đó lại không làm như vậy khi đặt hàng một sản phẩm phần mềm. Khoảng từ 40% đến 60% các lỗi xuất hiện là do không tìm hiểu kỹ bài toán trong khâu yêu cầu (Leffingwell 1997). Nhiều tổ chức vẫn ứng dụng các phương pháp theo kiểu thuận tiện, có sao thì làm vậy (ad hoc) để làm yêu cầu. Kết quả đạt được là một sự vênh nhau giữa cái khách hàng nghĩ là chúng ta sẽ xây dựng cho họ và cái mà chúng ta nghĩ chúng ta đang làm cho khách hàng.

Do yêu cầu là đầu vào của quy trình sản xuất phần mềm và mọi hoạt động quản lý dự án, nên tất cả những người liên quan cần phải đồng thuận trong một quy trình làm yêu cầu (*còn gọi là công nghệ yêu cầu - ND*) hiệu quả. Chương này giúp bạn:

- Hiểu một số khái niệm chính trong việc phát triển yêu cầu phần mềm.
- Có hiểu biết về một số vấn đề liên quan đến yêu cầu có thể nảy sinh trong một dự án phần mềm.
- Tìm hiểu về một số đặc tính mà một yêu cầu tuyệt vời hoặc một đặc tả yêu cầu cần phải có.
- Nhận biết sự khác nhau giữa phát triển yêu cầu và quản lý yêu cầu.

I. ĐỊNH NGHĨA YÊU CẦU PHẦN MỀM (SOFTWARE REQUIREMENTS DEFINED)

Một khó khăn đối với ngành công nghiệp phần mềm là thiếu các định nghĩa chung về các khái niệm mà chúng ta sử dụng để mô tả các công việc, một trong số đó là định nghĩa khái niệm “requirement”. Định nghĩa này cần đáp ứng cho nhiều đối tượng liên quan khác nhau như: developers, customers, others (*người phát triển, khách hàng, người khác – Một số thuật ngữ sẽ được để nguyên tiếng Anh – ND*).

The IEEE Standard Glossary of Software Engineering Technology (1997) định nghĩa một yêu cầu là:

- (1) Một quy ước (condition) hoặc một công năng (capability) của sản phẩm phần mềm cần thiết cho người dùng để giải quyết một vấn đề hoặc để đạt được một mục tiêu.
- (2) Một quy ước hoặc một công năng cần phải thỏa mãn hoặc cần phải có của một hệ thống hoặc một thành phần hệ thống (system component) nhằm đáp ứng một hợp đồng, một tiêu chuẩn, một đặc tả hoặc một tài liệu bắt buộc khác.
- (3) Một sự diễn giải (representation) được ghi thành tài liệu của một quy ước hoặc một công năng theo 1 hoặc 2.

1. MỘT SỐ DIỄN GIẢI VỀ “YÊU CẦU” (SOME INTERPRETATIONS OF “REQUIREMENTS”)

Định nghĩa của IEEE bao hàm cả 2 quan điểm về yêu cầu từ người dùng (quan sát hành vi của hệ thống từ bên ngoài) và từ người phát triển (quan sát hệ thống từ bên trong).

Một trong các yếu tố quan trọng của requirements là cần được tài liệu hoá. Tôi đã làm việc trong một dự án mà tại đó developers bị quay như chong chóng. Khách hàng chính phát hoảng khi người phân tích yêu cầu mới đến và nói: “Chúng tôi cần trao đổi về yêu cầu của anh.” Khách hàng phản ứng lại: “Tôi đã đưa cho người tiền nhiệm của anh yêu cầu rồi.” Thực tế thì không có yêu cầu nào được tài liệu hoá, vì vậy mỗi người phân tích mới đã phải bắt đầu từ một đống lộn xộn.

Thường thì bạn đã có “yêu cầu” rồi nếu bạn có được các emails, voice-mails, các ghi chú, các cuộc gặp gỡ ngắn, các tập hợp giấy tờ linh tinh từ các cuộc họp với người dùng.

Một định nghĩa khác gợi ý rằng yêu cầu là “các phát biểu về nhu cầu (need) của người dùng làm điểm xuất phát cho sự phát triển một chương trình hoặc một hệ thống” (Jones 1994). Chuyên gia về yêu cầu Alan Davis (1993) đã mở rộng khái niệm này thành “một nhu cầu của người dùng hoặc một đặc tính, một chức năng, một thuộc tính cần thiết của một hệ thống **có thể được hiểu từ một vị trí bên ngoài** hệ thống đó”. Các định nghĩa này nhấn mạnh *cái* mà sản phẩm phải cung cấp thay vì sản phẩm sẽ được làm *nhu thế nào* từ góc nhìn của người thiết kế và thi công.

Định nghĩa sau đi xa hơn: từ nhu cầu người dùng tới các đặc tính (characteristic) của hệ thống (Sommerville and Sawyer 1997):

Yêu cầu là... một đặc tả về cái phải được thi công. Chúng là các mô tả về hành vi của hệ thống phải như thế nào, hoặc về một thuộc tính của hệ thống phải như thế nào. Yêu cầu có thể là một ràng buộc về quy trình phát triển của hệ thống.

Các định nghĩa đa dạng trên chỉ ra rằng không có cách hiểu sáng sủa, không nhập nhằng về khái niệm “yêu cầu”. **Yêu cầu thật sự thường tồn tại trong tâm trí con người.** Bất cứ hình thức tài liệu hoá nào về yêu cầu (ví dụ đặc tả yêu cầu) cũng chỉ là một mô hình, một sự trình bày ra bên ngoài về yêu cầu mà thôi (Lawrence 1998). Chúng ta cần đảm bảo rằng tất cả những người liên quan của dự án đều có một cách hiểu chung về các khái niệm được dùng để mô tả các yêu cầu.

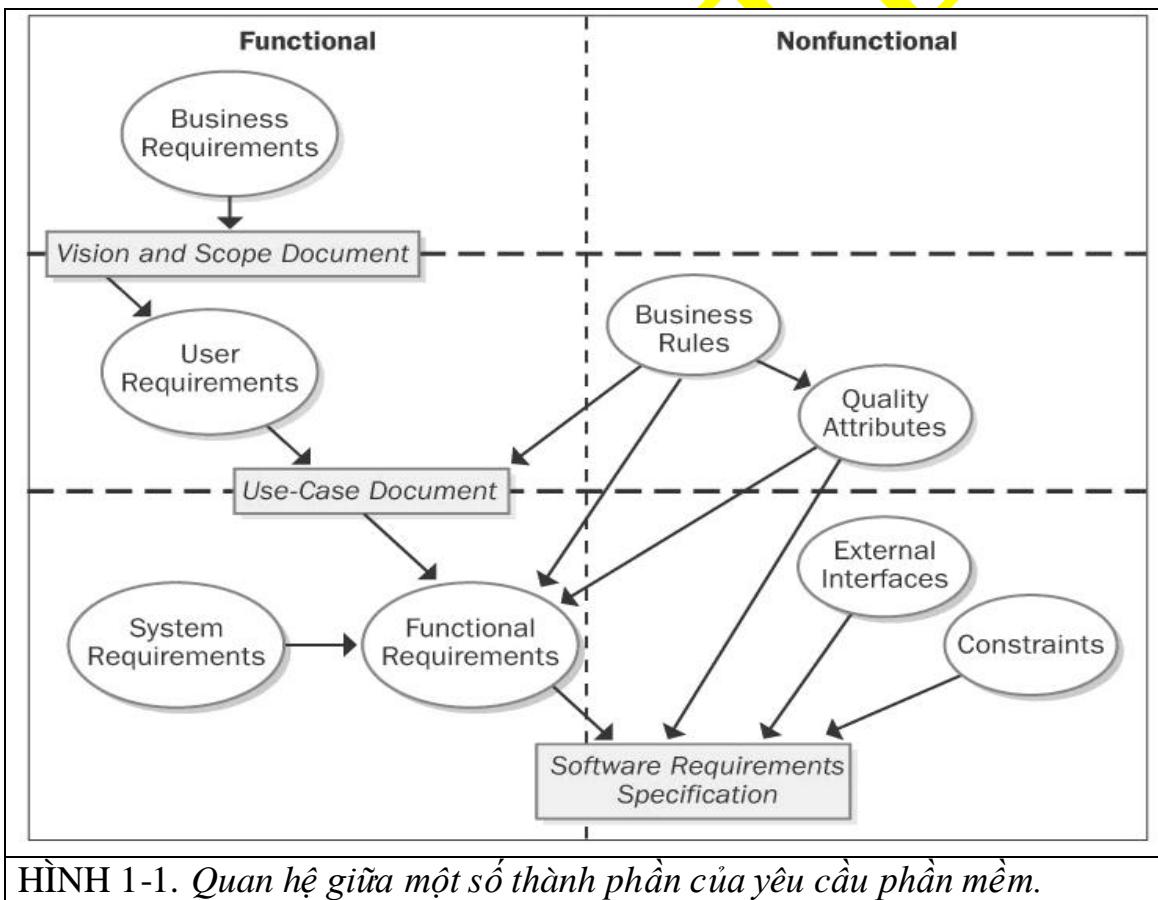
2. CÁC MỨC CỦA YÊU CẦU (LEVELS OF REQUIREMENTS)

Các định nghĩa sau mà tôi sẽ sử dụng đối với một số khái niệm chung, bạn sẽ thường gặp chúng trong lĩnh vực công nghệ yêu cầu.

Yêu cầu phần mềm gồm 3 mức phân biệt – **yêu cầu kinh doanh (business requirements, BR)**, **yêu cầu người dùng (user requirements, UR)**, **yêu cầu chức năng (functional requirements, FR)** hoặc **yêu cầu phi chức năng (NFR)**.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- BR biểu diễn các mục tiêu ở mức cao của tổ chức hoặc của khách hàng đối với hệ thống. Chúng được trích ra (capture) từ một tài liệu mô tả tầm nhìn (vision) và phạm vi (scope) của dự án.
- UR mô tả các tác vụ (task) mà người dùng phải hoàn thành bằng cách sử dụng hệ thống như một công cụ làm việc. Chúng được trình ra trong các mô tả tình huống sử dụng (use case) hoặc kịch bản sử dụng.
- FR định nghĩa chức năng của phần mềm mà người phát triển cần phải đưa vào sản phẩm để người dùng hoàn thành tác vụ của họ, do đó mà đáp ứng các yêu cầu kinh doanh (BR). Một *tính năng* (feature) là một tập hợp logic các yêu cầu chức năng có liên quan lẫn nhau nhằm cung cấp cho người dùng khả năng (capability) đáp ứng một yêu cầu kinh doanh. Xem Hình 1-1 về mối quan hệ giữa các mức của yêu cầu.



FR được tài liệu hóa trong một *đặc tả yêu cầu phần mềm* (*Software requirements specification*, SRS), tài liệu này mô tả đầy đủ ở mức có thể

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

hành vi mong muốn đối với hệ thống bằng một cái nhìn từ bên ngoài. SRS được sử dụng trong phát triển, kiểm thử, đảm bảo chất lượng, quản lý dự án và các công việc liên quan khác của dự án. Đối với các sản phẩm phức tạp, yêu cầu chức năng có thể là tập con của yêu cầu hệ thống, nếu một phần của yêu cầu chức năng đã được định vị thành các software components.

Ngoài các FR, SRS còn chứa các yêu cầu phi chức năng. Đó là các tiêu chuẩn, quy định, khé ước mà sản phẩm phải tuân thủ; các mô tả của giao diện ngoài (external interface, nói thêm, giao diện trong là giao diện giữa các hệ thống với nhau); các yêu cầu hiệu suất (performance requirements); các ràng buộc về thiết kế và thi công; các thuộc tính chất lượng. Các ràng buộc (constraints) là các giới hạn được định ra trong miền chọn lựa khả năng của người phát triển khi thiết kế và thi công hệ thống. Các thuộc tính chất lượng là các mô tả về các thuộc tính của sản phẩm theo nhiều khía cạnh quan trọng với người dùng hoặc với người phát triển.

Chúng ta lấy một chương trình xử lý từ làm ví dụ về các kiểu yêu cầu khác nhau. Một BR có thể viết: “Người dùng có thể sửa các lỗi chính tả (spelling errors) trong tài liệu một cách hiệu quả”. Như vậy, trong tài liệu hướng dẫn sử dụng sản phẩm cần ghi rằng một spelling checker được đưa vào sản phẩm – đó là tính năng đáp ứng BR này. Các UR tương ứng có thể mô tả (use case): “Tìm kiếm các spelling errors (lỗi chính tả) trong tài liệu và quyết định liệu có nên thay thế mỗi misspelled word (từ sai chính tả) bằng một từ đúng lựa ra từ một danh sách các từ được gọi ý”. Spelling checker có nhiều yêu cầu chức năng cụ thể như tìm kiếm và làm sáng (highlight) một misspelled word, hiển thị một dialog box với các từ thay thế được gợi ý...

Các nhà quản lý hoặc tiếp thị có thể định nghĩa các yêu cầu kinh doanh cho phần mềm, điều đó giúp công ty của họ hoạt động hiệu quả hơn (đối với các hệ thống thông tin) hoặc thành công trên thị trường (đối với các sản phẩm phần mềm thương mại). Tất cả các yêu cầu người dùng cần phải song đôi với các yêu cầu kinh doanh. Các yêu cầu người dùng giúp người phân tích cài đặt một phần chức năng (the bits of functionality) mong muốn trong sản phẩm nhằm giúp người dùng thực hiện tác

vụ của họ. Người phát triển (developers) sử dụng các yêu cầu chức năng để thiết kế các giải pháp thực thi chức năng cần thiết.

Chú ý rằng yêu cầu, theo định nghĩa, không bao hàm các chi tiết thiết kế, các chi tiết thi công, thông tin lập kế hoạch dự án, hoặc thông tin kiểm thử. Hãy tách các thông tin đó ra khỏi yêu cầu sao cho yêu cầu chỉ chứa các thông tin hệ thống cần phải làm cái gì. Dự án cũng có thể có các kiểu yêu cầu khác nhau như yêu cầu về môi trường phát triển, yêu cầu về phát hành sản phẩm và thích ứng nó với môi trường hỗ trợ. Các kiểu yêu cầu đó có thể ảnh hưởng nghiêm trọng đến sự thành công của dự án nhưng trong cuốn sách này ta không xét đến chúng.

II. MỌI DỰ ÁN ĐỀU CÓ YÊU CẦU (EVERY PROJECT HAS REQUIREMENTS)

Fredrick Brooks xác định vai trò đặc biệt quan trọng của quy trình yêu cầu đối với các dự án phần mềm trong bài luận kinh điển năm 1987 của ông, “No Silver Bullet: Essence and Accidents of Software Engineering”:

Việc duy nhất khó khăn khi làm một hệ thống phần mềm là quyết định chính xác cái cần phải xây dựng. Không có công việc nào là khó khăn như thiết lập các yêu cầu kỹ thuật chi tiết, gồm tất cả giao diện với người dùng, giao diện với máy móc (machines) và với các hệ thống phần mềm khác. Không có công việc nào ảnh hưởng xấu đến hệ thống bằng công việc này nếu nó bị thực hiện sai. Không có công việc nào là khó khăn hơn công việc này nếu phải chỉnh sửa lại sau.

Mỗi hệ thống phần mềm có người dùng của riêng nó. Thời gian để tìm hiểu nhu cầu của họ là một khoản đầu tư then chốt khiến dự án thành công. Nhận định này là hiển nhiên đối với các ứng dụng đầu cuối thương mại, các hệ thống thông tin doanh nghiệp, các sản phẩm chứa các phần mềm nhúng. Nếu chúng ta, với tư cách là những người phát triển, không viết ra các yêu cầu để khách hàng có thể thảo luận về nó, thì làm sao chúng ta biết khi nào thì dự án thực hiện xong. Chúng ta có thể đáp ứng như thế nào cho khách hàng nếu không biết cái gì quan trọng đối với họ. Thậm chí đối với ngay cả các phần mềm phi thương mại thì các yêu cầu cũng cần phải được hiểu cho rõ. Ví dụ các thư viện phần mềm, các components, các tools được tạo ra cho các nhóm phát triển sử dụng.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

III. KHI CÁC YÊU CẦU XẤU XÂY RA VỚI CON NGƯỜI TỐT (WHEN BAD REQUIREMENTS HAPPEN TO NICE PEOPLE)

Khi các nhóm dự án không quan tâm đến các quy trình yêu cầu thì họ sẽ khiến dự án lâm vào tình thế nguy hiểm, sự thành công của dự án là khó đảm bảo. Nếu “thành công” được hiểu là chuyển giao một sản phẩm đáp ứng các mong đợi của người dùng về chức năng, về chất lượng ở mức chi phí đã thoả thuận và trong một thời gian giới hạn. Một số rủi ro về yêu cầu sẽ được thảo luận dưới đây. Chương 5 sẽ thảo luận việc bạn có thể làm gì để tránh các rủi ro yêu cầu.

MỘT SỐ RỦI RO TỪ SỰ KHÔNG ĐẦY ĐỦ CỦA QUY TRÌNH YÊU CẦU (SOME RISKS FROM INADEQUATE REQUIREMENTS PROCESS)

- Nếu các kiểu người dùng không được tính đến đầy đủ trong khi làm yêu cầu thì hệ thống có thể sẽ không được chấp nhận.
- Việc phát sinh các yêu cầu không chính thức (creeping user requirements) góp phần làm hỏng và giảm chất lượng sản phẩm.
- Các yêu cầu nhập nhằng dẫn tới tiêu phí nhiều thời gian để làm lại.
- “Sự mạ vàng” (gold – plating) của người phát triển và người dùng sẽ thêm vào hệ thống các tính năng (features) không cần thiết.
- Các đặc tả tối thiểu dẫn tới các yêu cầu chính bị thiếu hụt (missing key requirements).
- Bỏ qua nhu cầu của một số kiểu người dùng nào đó sẽ dẫn tới sự không hài lòng của khách hàng.
- Các yêu cầu được định nghĩa không đầy đủ khiến đòi hỏi lập kế hoạch dự án chính xác và giám sát là không thể.

Các kiểu người dùng không được tính đến đầy đủ (Insufficient user involvement)

Khách hàng thường không hiểu tại sao việc thu thập yêu cầu và đảm bảo chất lượng yêu cầu lại khó khăn đến thế. Người phát triển không thể nhấn mạnh đến sự tham gia của người dùng (user involvement), hoặc cũng có thể là làm việc với người dùng thì không vui như viết code, hoặc cũng có thể họ nghĩ họ đã biết cái người dùng cần. Trong một số trường hợp, trao đổi trực tiếp với những người sẽ sử

dụng sản phẩm không phải là điều dễ dàng, trong khi đó những người đại diện của khách hàng không phải bao giờ cũng hiểu được chính xác cái mà người dùng thật sự cần. Không có sự thay thế nào cho sự tham gia của các đại diện người dùng một cách trực tiếp trong nhóm dự án ngay từ sớm và kết hợp với họ liên tục trong suốt quy trình phát triển.

Việc phát sinh các yêu cầu không chính thức (Creeping user requirements)

Do các yêu cầu liên tục tiến hoá khi phát triển dự án, nên dự án luôn có thể vượt khỏi các lịch biểu và ngân sách đã định. Các dự án như vậy không phải luôn dựa trên sự hiểu biết thực tế về kích thước và độ phức tạp của các yêu cầu, các rủi ro, năng suất làm việc của dự án, việc sinh ra các yêu cầu không chính thức có thể gây ra các vấn đề to lớn. Vấn đề có thể phụ thuộc một phần vào đề nghị thay đổi yêu cầu của người dùng, phụ thuộc một phần vào cách những người phát triển đáp ứng các yêu cầu và chỉnh sửa mới.

Để quản lý việc sinh ra các yêu cầu không chính thức, **chúng ta cần làm sáng sủa các báo cáo về tầm nhìn, phạm vi, mục tiêu, giới hạn và các tiêu chuẩn thành công của dự án**. Báo cáo này sẽ được sử dụng như một khung tham chiếu mỗi khi một tính năng mới được đề nghị, hoặc các thay đổi yêu cầu được đề xuất. Một quy trình kiểm soát thay đổi được thiết kế tốt bao gồm việc phân tích ảnh hưởng của mỗi thay đổi được đề xuất, việc này sẽ giúp các người liên quan đề ra quyết định liệu thay đổi đó có được chấp nhận hay không trong sự cân bằng các chi phí, thời hạn, nguồn lực hoặc các tính năng khác.

Do các thay đổi có thể lan truyền trên toàn bộ sản phẩm đang được phát triển nên kiến trúc của hệ thống có thể bị phá vỡ dần dần. Các miếng vá sẽ khiến chương trình khó khăn hơn để hiểu và bảo trì. Sự liên quan lẩn nhau của các mã thêm vào (insertion of additional code) có thể gây ra các vấn đề đối với nguyên tắc thiết kế hệ thống bền vững. Nếu bạn có thể xác định sớm các tính năng có khả năng sẽ thay đổi theo thời gian thì bạn có thể thiết lập một kiến trúc bền vững cho hệ thống, do vậy mà kiểm soát được tốt hơn sự thay đổi. Các thay đổi yêu cầu nếu được thực hiện thông qua thay đổi thiết kế, thay vì phát hành các bản vá lỗi, thì sẽ giúp kiểm soát tốt hơn chất lượng sản phẩm.

Các yêu cầu nhập nhằng (Ambiguous Requirements)

Sự nhập nhằng là khó khăn to lớn đối với việc đặc tả yêu cầu (Lawrence 1996). Sự nhập nhằng khiến cho những người đọc các báo cáo mô tả yêu cầu đi đến những cách hiểu khác nhau về cái mà phần mềm phải làm. Một dấu hiệu khác nữa của sự nhập nhằng là ngay cả một người đọc cũng có thể diễn giải một yêu cầu theo nhiều cách khác nhau.

Sự nhập nhằng dẫn tới các kỳ vọng (expectation) khác nhau từ người liên quan, một số trong họ sẽ ngạc nhiên về những gì được chuyển giao. Các yêu cầu nhập nhằng dẫn tới lãng phí thời gian khi các nhà phát triển cài đặt một giải pháp cho một vấn đề sai, khi các nhà kiểm thử kỳ vọng sản phẩm có những hành vi khác với những gì mà các nhà phát triển đã xây dựng. Trước đây, một nhà kiểm thử hệ thống đã nói với tôi rằng nhóm kiểm thử của cô thường diễn giải sai các yêu cầu, vì vậy đã phải viết lại nhiều test cases và lặp đi, lặp lại nhiều kiểm thử.

Kết quả không thể tránh được của sự nhập nhằng là phải làm lại công việc đã làm. Các công việc làm lại có thể tiêu tốn 40% tổng chi phí phát triển, 70% - 80% việc sửa lại được quy cho các lỗi yêu cầu (Leffingwell 1997).

Một cách để truy tìm sự nhập nhằng là có một nhóm đại diện cho nhiều quan điểm khác nhau thanh tra (inspect) các yêu cầu. Nếu mỗi người trong nhóm đó diễn giải một yêu cầu theo nhiều cách khác nhau thì đã có sự nhập nhằng trong báo cáo yêu cầu. Các kỹ thuật khác nhau để phát hiện sự nhập nhằng được mô tả bởi Gause và Weinberg (1989) trong phần sau của chương này.

Các tính năng không cần thiết (Unnecessary Features)

Mạ vàng (gold plating) được hiểu là khuynh hướng một số nhà phát triển thêm một chức năng mới không có trong đặc tả yêu cầu nhưng họ lý giải rằng “người dùng sẽ thích nó”. Thường thì người dùng không tìm kiếm chức năng này và nguồn lực tiêu tốn để cài đặt nó đã bị lãng phí. Một cách làm mà những người phát triển thích hơn so với chỉ đơn giản thêm các tính năng mới là họ giới thiệu với khách hàng các ý tưởng, các lựa chọn và các cách tiếp cận sáng tạo nhằm giải quyết vấn đề của khách hàng. Quyết định về chức năng nào được thêm vào sẽ được

tính toán cân bằng giữa điều kiện hàng muôn và sự xem xét, cân nhắc về tính khả thi kỹ thuật, khung thời gian.

Để tối thiểu hoá nguy cơ mạ vàng, bạn cần giám sát tất cả các tính năng được thêm vào, mỗi tính năng cần phải có lý do chính đáng để được chấp nhận.

Đặc tả tối thiểu (Minimal Specification)

Đôi khi, khách hàng và các bên khác nhau (marketing, quản lý) không hiểu được tại sao phải nhấn mạnh các yêu cầu là quan trọng. Để tạo ra một đặc tả tối thiểu, có lẽ không gì hơn là vạch ra các ý tưởng về sản phẩm trên một tờ giấy mỏng và yêu cầu các nhà phát triển làm mịn, bổ sung nó khi dự án tiến triển trên thực tế. Một dấu hiệu xấu của việc này là các nhà phát triển viết các yêu cầu sau khi hoàn tất xây dựng sản phẩm. Cách làm này có thể thích hợp với các sản phẩm có tính cách thăm dò cao, hoặc khi mà các yêu cầu phải thật sự mềm dẻo (McConnell 1996). Trong phần lớn các trường hợp, cách làm này khiến các nhà phát triển thất bại (người có thể phải làm việc dưới các giả thiết sai và trong một định hướng hạn chế) và khách hàng thấy mình bị làm phiền (người nhận sản phẩm như họ đã mường tượng).

Bỏ qua nhu cầu của một số kiểu người dùng (Overlooked User Classes)

Phần lớn các sản phẩm được sử dụng bởi các nhóm người khác nhau, họ có thể sử dụng các tập con tính năng khác nhau, có tần suất sử dụng khác nhau, có kinh nghiệm và trình độ khác nhau. Nếu bạn không xác định tất cả các kiểu người dùng chính – các lớp người dùng (user classes) - của sản phẩm ngay từ sớm thì sẽ có ai đó trong số khách hàng không hài lòng khi sản phẩm được chuyển giao. Ví dụ, các xử lý sử dụng menu là không hiệu quả đối với những người dùng chuyên nghiệp, trong khi các câu lệnh và các phím tắt lại làm những người sử dụng không thường xuyên cảm thấy bối rối.

Lập kế hoạch không chính xác (Inaccurate Planning)

“Đây là ý tưởng của tôi về một sản phẩm mới; bạn có thể cho tôi biết ý tưởng này khi nào thì được thực hiện?”. Nhiều nhà phát triển phải đối mặt với vấn đề khó khăn này. Sự thiếu hiểu biết về yêu cầu sẽ dẫn tới các ước lượng quá lạc quan về dự án. Năm nguyên nhân hàng đầu đã được tổng kết về các ước lượng chi phí phần *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

mềm xa thực tế liên quan đến yêu cầu là: thay đổi yêu cầu thường xuyên, thiếu yêu cầu, truyền thông không đầy đủ về yêu cầu cho người dùng, đặc tả sơ sài về yêu cầu, phân tích yêu cầu không đầy đủ (Davis 1995).

IV. LỢI ÍCH TỪ MỘT QUY TRÌNH YÊU CẦU CHẤT LƯỢNG CAO (BENEFITS FROM A HIGH - QUALITY REQUIREMENTS PROCESS)

Các tổ chức sử dụng các quy trình công nghệ yêu cầu hiệu quả có thể vui lòng với nhiều lợi ích thu được. Có lẽ phần thưởng lớn nhất là giảm được các công việc phải làm lại trong giai đoạn phát triển và giai đoạn bảo trì dài. Boehm (1981) phát hiện ra rằng việc sửa một lỗi yêu cầu sau khi sản phẩm đã được phát hành tốn kém gấp 68 lần so với sửa yêu cầu đó ngay trong giai đoạn yêu cầu. Các nghiên cứu gần đây cho biết yếu tố khuếch đại chi phí này có thể là 200 lần. Hiệu quả của việc làm các yêu cầu chất lượng cao ngay từ đầu không phải là rõ ràng với nhiều người, họ chỉ đơn giản cho rằng thời gian tiêu phí cho giai đoạn yêu cầu sẽ làm chậm thời điểm bàn giao sản phẩm cho khách hàng.

Quy trình yêu cầu hiệu quả nhấn mạnh cách tiếp cận hợp tác khi xây dựng sản phẩm, quy trình này liên quan đến nhiều người liên quan khác nhau của dự án. Tập hợp các yêu cầu cho phép nhóm dự án hiểu tốt hơn thị trường của sản phẩm, một yếu tố thành công then chốt của bất cứ dự án nào. Cuối cùng, các yêu cầu được tài liệu hóa và không nhập nhằng làm thuận tiện tối đa đối với việc kiểm thử hệ thống và tăng cơ hội chuyển giao một sản phẩm chất lượng cao đáp ứng mong đợi của tất cả những người liên quan.

V. ĐẶC TÍNH CỦA CÁC YÊU CẦU TUYỆT VỜI (CHARACTERISTICS OF EXCELLENT REQUIREMENTS)

Các đặc tính mà một báo cáo yêu cầu tốt cần tuân theo sẽ được thảo luận ở đây (Davis 1993; IEEE 1998). Sự soát xét cẩn thận SRS bởi những người liên quan đại diện cho các góc nhìn khác nhau là cách tốt nhất để xác định liệu các yêu cầu có đầy đủ các đặc tính cần thiết hay không. Nếu bạn lưu giữ các đặc tính đó trong tâm trí khi viết và soát xét các yêu cầu thì bạn sẽ có một thiết kế yêu cầu tốt hơn (mặc dù chưa bao giờ hoàn hảo cả). Trong chương 9, bạn sẽ sử dụng các đặc tính này để tìm kiếm các vấn đề với một số báo cáo yêu cầu sao cho bạn có thể cải tiến chúng.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

1. CÁC ĐẶC TÍNH CỦA LỜI THÊ HIỆN YÊU CẦU (REQUIREMENT STATEMENT CHARACTERISTICS)

Đầy đủ (Complete)

Mỗi yêu cầu cần mô tả đầy đủ chức năng được chuyển giao. Nó phải chứa tất cả các thông tin cần thiết để nhà phát triển thiết kế và thi công chức năng này.

Đúng đắn (Correct)

Mỗi yêu cầu cần mô tả chính xác chức năng được xây dựng. Sự đảm bảo cho tính đúng đắn đó là tham chiếu đến nguồn của yêu cầu, đó có thể là khách hàng hoặc một đặc tả yêu cầu hệ thống mức cao hơn. Một yêu cầu phân mềm mà xung đột với một yêu cầu hệ thống tương ứng thì là không đúng đắn. Chỉ sự trình bày của người dùng mới có thể xác định tính đúng đắn của yêu cầu người dùng, điều đó cho biết tại sao khi soát xét yêu cầu ta cần sự có mặt của chính người dùng hoặc người đại diện của họ. Soát xét yêu cầu mà không có mặt người dùng có thể khiến những người soát xét nói: “Điều này không có ý nghĩa. Đó có thể là suy diễn.”

Khả thi (Feasible)

Khả thi có nghĩa là thực thi mỗi yêu cầu trong các khả năng và giới hạn đã biết của hệ thống và môi trường hoạt động của hệ thống. Để tránh các yêu cầu không khả thi, cần một thành viên của nhóm dự án làm việc với các nhà marketing hoặc các nhà phân tích yêu cầu trong quá trình xử lý yêu cầu (elicitation process). Người này sẽ đánh giá về tính khả thi của các yêu cầu về mặt kỹ thuật hoặc các yêu cầu có thể thực thi nhưng với một chi phí lớn.

Cần thiết (Necessary)

Mỗi yêu cầu cần phải tài liệu hóa một cái gì đó mà khách hàng thật sự cần hoặc một hệ thống khác bên ngoài cần. Một cách khác để xác nhận “tính cần thiết” là yêu cầu đó được đề xuất từ một bên mà bạn biết rất rõ rằng anh ta có thẩm quyền đề ra yêu cầu.

Được xếp thứ tự ưu tiên (Prioritized)

Gán một thứ tự ưu tiên cho mỗi yêu cầu, tính năng (feature), hoặc use-case để có thể hình dung lịch trình phát hành các phiên bản của sản phẩm. Nếu tất cả các yêu
Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

cầu được coi là quan trọng như nhau thì quản trị dự án sẽ không xác định được cách thức thi công khi một yêu cầu mới phát sinh ngay trong quá trình thi công của dự án, anh ta sẽ không kiểm soát được ngân sách, lịch biểu và nhân lực của dự án. Chương 13 sẽ thảo luận về thứ tự ưu tiên chi tiết.

Không nhập nhằng (Unambiguous)

Tất cả những ai khi đọc bản báo cáo yêu cầu đều có cùng một cách hiểu, một cách diễn giải nhất quán về nội dung của các yêu cầu. Do ngôn ngữ tự nhiên là có tính nhập nhằng cao nên viết một yêu cầu rõ ràng, cụ thể, đơn nghĩa không phải là dễ. Cách hiểu quả để loại bỏ tính nhập nhằng là mô tả các báo cáo yêu cầu bằng các ngôn ngữ hình thức như use-case chẳng hạn, qua các kịch bản sử dụng cụ thể.

Có thể kiểm tra (Verifiable)

Hãy kiểm tra mỗi yêu cầu để xem liệu bạn có thể nghĩ ra một số lượng nhỏ các phép tests hoặc sử dụng một cách tiếp cận kiểm tra khác như thanh tra (inspection) hoặc chứng minh (demonstration) để biết liệu yêu cầu đó đã được cài đặt hợp lệ trong sản phẩm hay không. Nếu một yêu cầu không thể kiểm tra thì xác định liệu nó có được cài đặt đúng đắn hay không sẽ trở thành vấn đề gây tranh cãi. Các yêu cầu không nhất quán, không khả thi hoặc nhập nhằng thì cũng không thể kiểm tra được.

2. CÁC ĐẶC TÍNH CỦA ĐẶC TẢ YÊU CẦU (REQUIREMENT SPECIFICATION CHARACTERISTICS)

Đầy đủ (Complete)

Không yêu cầu hoặc thông tin cần thiết nào bị mất. Các yêu cầu bị mất rất khó phát hiện do sự “vô hình” (invisible) của chúng. Tập trung vào các tác vụ của người dùng thay vì tập trung vào các chức năng hệ thống sẽ giúp bạn tránh được sự không đầy đủ đó. Nếu bạn biết bạn thiếu một thông tin nào đó, hãy sử dụng TBD (“to be determined”) như là một dấu hiệu tiêu chuẩn (standard flag) để đánh dấu các “nếp gãy” (gaps) đó. Hãy phân giải (resolve) tất cả các TBDs trước khi tiến hành xây dựng.

Nhất quán (Consistent)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Các yêu cầu nhất quán không xung đột với các yêu cầu phần mềm khác hoặc các yêu cầu cấp cao hơn (hệ thống hoặc kinh doanh). Tất cả các mâu thuẫn trong yêu cầu cần phải được phân giải trước khi quá trình phát triển diễn ra. Bạn có thể không biết một yêu cầu đơn nhất (single requirement) nào đó là đúng đắn cho đến tận khi bạn tiến hành một số nghiên cứu nào đó về yêu cầu này.

Có thể sửa chữa (Modifiable)

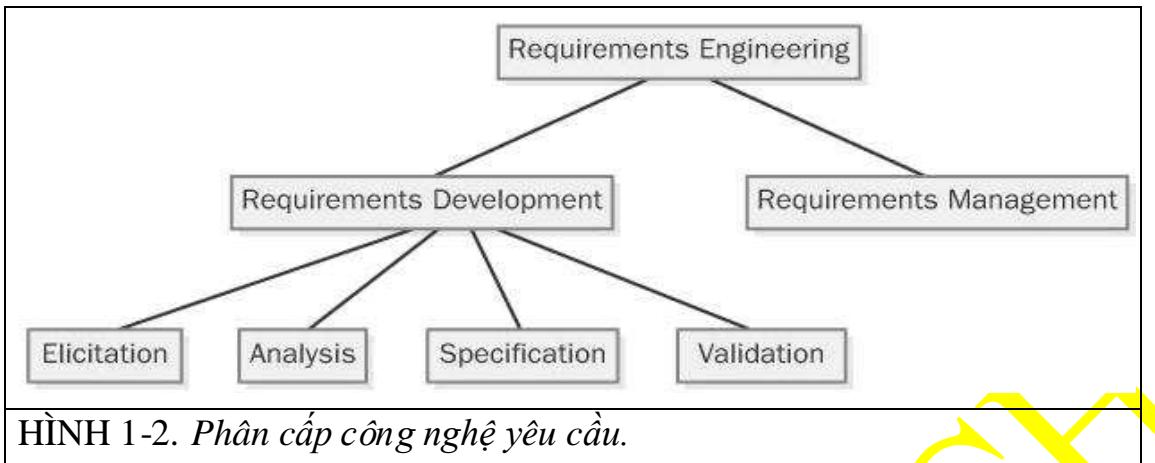
Bạn cần nghiên cứu lại SRS khi cần thiết và duy trì thông tin diễn biến thay đổi của mỗi yêu cầu. Điều này đòi hỏi mỗi yêu cầu được dán nhãn duy nhất và được diễn giải riêng rẽ với các yêu cầu khác sao cho mỗi yêu cầu đều được tham chiếu chính xác. Mỗi yêu cầu chỉ được xuất hiện duy nhất 1 lần trong SRS để tránh sự không nhất quán giữa các thể hiện (instance) của cùng 1 yêu cầu tại những nơi khác nhau. Một bảng nội dung (table of contents), một index, một danh sách tham chiếu chéo (cross – reference listing) sẽ làm SRS dễ sửa chữa hơn.

Có thể theo vết (Traceable)

Bạn cần phải liên kết các yêu cầu tới nguồn phát sinh của nó, tới các phần tử thiết kế, mã nguồn, các test cases thực thi và kiểm tra sự đúng đắn trong việc thi công các yêu cầu. Các yêu cầu có thể theo vết được gán nhãn duy nhất và được viết theo một cách có cấu trúc, chi tiết và được thuyết minh đầy đủ. Chương 18 sẽ tập trung trình bày nội dung này.

VI. PHÁT TRIỂN VÀ QUẢN LÝ YÊU CẦU (REQUIREMENTS DEVELOPMENT AND MANAGEMENT)

Sự tối nghĩa của thuật ngữ yêu cầu khiến cho có nhiều cách hiểu khác nhau và cách làm khác nhau đối với việc thu thập và xử lý yêu cầu. Một số tác giả gọi toàn bộ quy trình yêu cầu là “công nghệ yêu cầu” (requirement engineering), một số khác lại gọi là “quản lý yêu cầu” (requirement management). Thường thì người ta chia nhỏ toàn bộ quy trình này ra thành *phát triển yêu cầu* (*requirements development*) (trình bày trong Phần II) và *quản lý yêu cầu* (*requirements management*) (trình bày trong Phần III), như thể hiện ở Hình 1-2.



Phát triển yêu cầu cũng có thể được chia nhỏ thành *suy luận, phân tích, đặc tả, kiểm tra* (*elicitation, analysis, specification, verification*) (Thayer and Dorfman 1997). Tất cả các nhánh công việc con đó bao trùn toàn bộ các hoạt động thu thập, đánh giá, tài liệu hoá yêu cầu của một phần mềm. Các hoạt động phát triển yêu cầu gồm:

- Xác định các lớp người dùng kỳ vọng (expected user classes) của sản phẩm.
- Suy luận nhu cầu (needs) từ các cá nhân đại diện cho mỗi lớp người dùng đó.
- Tìm hiểu tác vụ của người dùng hiện tại, các mục tiêu và nhu cầu kinh doanh được các tác vụ đó trợ giúp.
- Phân tích thông tin nhận được từ người dùng để phân loại các nhu cầu tác vụ của họ với các yêu cầu chức năng, các quy tắc nghiệp vụ (business rules), các thuộc tính chất lượng, các giải pháp được đề nghị, thông tin xa lạ khác.
- Phân chia (partitioning) các yêu cầu mức hệ thống thành các hệ thống con cơ bản (major subsystems) và phân bổ (allocating) một số yêu cầu thành các software components.
- Tìm hiểu về tầm quan trọng tương đối của các thuộc tính chất lượng.
- Đàm phán về các ưu tiên trong thi công hệ thống.
- Biến đổi (translate) các nhu cầu người dùng đã tập hợp được thành các đặc tả và mô hình (specifications and models).

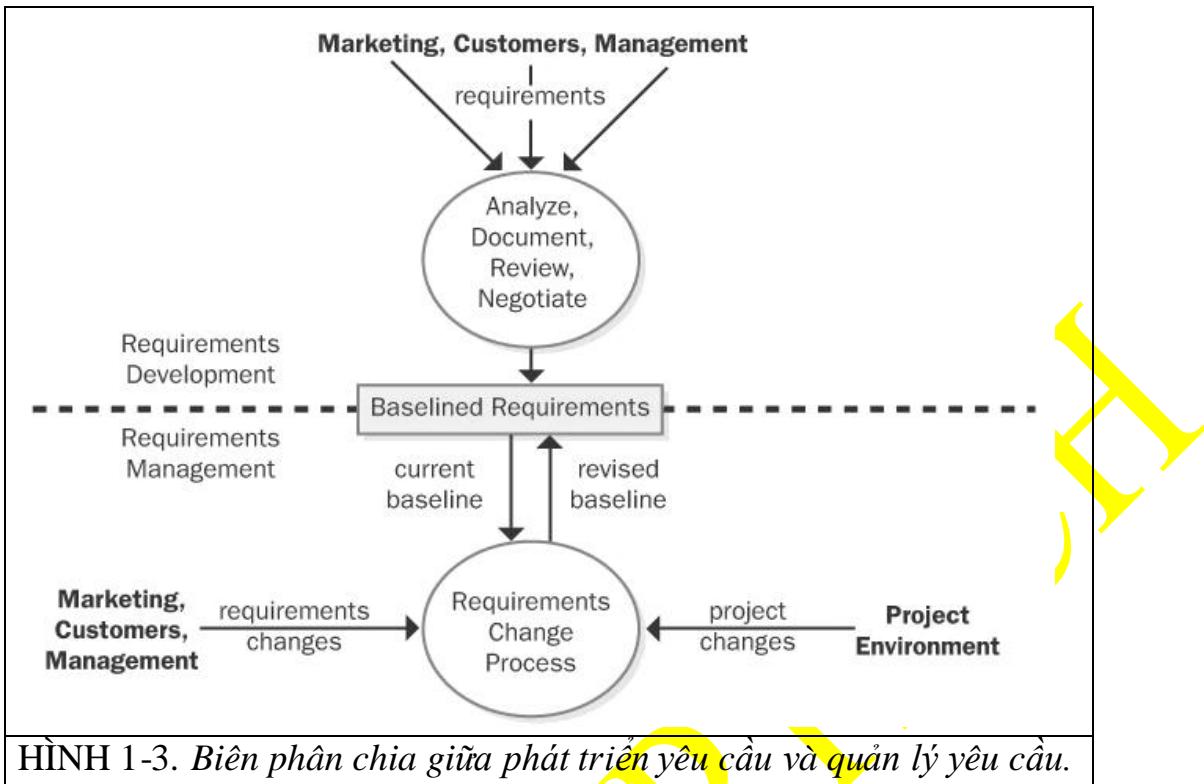
- Soát xét các đặc tả yêu cầu để đảm bảo một cách hiểu chung về các yêu cầu đã được định vị (stated requirements) và sửa chữa bất cứ vấn đề gì trước khi nhóm phát triển chấp nhận chúng.

Quản lý yêu cầu được hiểu là “thiết lập và duy trì một thoả thuận với khách hàng về các yêu cầu của dự án phần mềm” (CMU/SEI 1995). Tuy nhiên sự chấp nhận của khách hàng cũng mới chỉ chiếm một nửa điều kiện của thông qua yêu cầu. Các nhà phát triển cũng phải thoả thuận để chấp nhận chúng và xây dựng sản phẩm tương ứng. Thông thường các hoạt động quản lý yêu cầu bao gồm:

- Định nghĩa ranh giới (baseline) của hoạt động xử lý yêu cầu (ranh giới giữa phát triển và quản lý yêu cầu).
- Soát xét các thay đổi yêu cầu được đề nghị và đánh giá các ảnh hưởng của mỗi thay đổi đó trước khi quyết định có chấp nhận sự thay đổi hay không.
- Kết hợp các thay đổi yêu cầu đã được chấp nhận đó vào dự án theo một cách được kiểm soát.
- Bám sát kế hoạch dự án với sự thay đổi của các yêu cầu.
- Đàm phán về các cam kết mới căn cứ vào các ảnh hưởng được ước lượng của các yêu cầu thay đổi.
- Lần vét mỗi yêu cầu bị thay đổi đến các thiết kế tương ứng, mã nguồn, kịch bản kiểm thử.
- Hiệu chỉnh (tracking) trạng thái của các yêu cầu và các hoạt động thay đổi trên toàn bộ dự án.

Hình 1-3 thể hiện sự khác nhau giữa phát triển yêu cầu và quản lý yêu cầu (Xem hình này hiểu kỹ hơn khái niệm requirements baseline).





HÌNH 1-3. Biên phân chia giữa phát triển yêu cầu và quản lý yêu cầu.

Các bước tiếp theo

- Viết ra bất cứ vấn đề gì liên quan đến yêu cầu mà bạn đang phải đối mặt trong các dự án hiện tại và trước đây của bạn. Hãy chỉ ra vấn đề đó thuộc loại phát triển hay quản lý yêu cầu. Hãy xác định các ảnh hưởng gây ủa bởi mỗi vấn đề và nguyên nhân gốc của mỗi vấn đề.
- Hãy tổ chức một cuộc thảo luận với các thành viên trong nhóm của bạn và những người liên quan khác (khách hàng, marketing, các nhà quản trị dự án) về các vấn đề liên quan đến yêu cầu từ các dự án hiện tại và trước đây của bạn, về các ảnh hưởng và nguyên nhân gốc của chúng. Hãy giải thích với những người tham gia rằng họ phải đổi mới với các khó khăn nếu họ muốn nắm bắt chúng. Họ sẵn sàng chứ?
- Hãy sắp xếp một lớp học trong 1 ngày về yêu cầu cho nhóm của bạn. Nhóm đó gồm khách hàng chính, nhân viên marketing, các nhà quản lý, hãy làm bất cứ cái gì cần thiết để họ ngồi lại cùng nhau và học lớp này. Lớp học sẽ cung cấp cho họ một danh sách từ vựng chung, một cách hiểu chung về các vấn đề của việc phát triển và quản lý yêu cầu, nhờ vậy họ có thể chỉ mặt, đặt tên các thách thức mà họ cần vượt qua.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

CHƯƠNG 2

YÊU CẦU TỪ GÓC NHÌN CỦA KHÁCH HÀNG

Gerhard là một nhà quản lý cấp cao của Consoto Pharmaceuticals, anh có một cuộc gặp làm việc với Cynthia, nhà quản lý mới của nhóm phát triển hệ thống thông tin của Contoso. “Chúng ta cần xây dựng một hệ thống thông tin giám sát hoá chất (chemical-tracking information system, CTIS) cho Contoso,” Gerhard bắt đầu. “Hệ thống phải cho phép giám sát tất cả các chai hóa chất mà chúng ta có trong kho hoặc trong các phòng thí nghiệm. Như vậy, các nhà hóa học có thể biết hoá chất họ muốn dùng còn không và còn ở đâu thay vì khi nhu cầu xuất hiện thì lại đi mua chai mới. Văn phòng Sức khoẻ và An toàn cũng phải thực hiện các báo cáo về việc sử dụng hoá chất để gửi chính phủ liên bang. Nhóm của cô có thể giúp chúng tôi xây dựng phần mềm này trong 5 tháng chứ, nghĩa là phần mềm đã sẵn sàng hoạt động ngay khi chúng ta có cuộc kiểm toán thủ đầu tiên?”

“Tôi nghĩ đó chính là lý do tại sao dự án này quan trọng, Gerhard,” Cynthia trả lời. “Nhưng trước khi tôi có thể cam kết một lịch biểu, tôi cần thu thập một số yêu cầu về hệ thống này đã.”

Gerhard lúng túng. “Cô nói sao? Tôi vừa chẳng nói yêu cầu với cô đó thôi.”

“Thật sự là anh chỉ vừa nói về một ý tưởng và một số mục tiêu của dự án này,” Cynthia giải thích. “Các yêu cầu kinh doanh mức cao đó (high-level business requirements) không giúp tôi hình dung đủ chi tiết về cái gì cần làm và ước tính thời gian làm trong bao lâu. Tôi muốn có một nhóm kết hợp giữa các phân tích viên hệ thống và các nhà hóa học để hiểu rõ về nhu cầu đối với hệ thống này. Khi đó chúng tôi có thể phác ra chức năng nào là cần thiết nhằm đáp ứng cả mục tiêu kinh doanh và nhu cầu người dùng. Thậm chí anh có vẻ như không hề cần biết hệ thống này giúp anh như thế nào trong việc tiết kiệm được chi phí hoạt động.”

Gerhard không hề phải đổi mặt với sự phản ứng như thế này từ người phát triển hệ thống trước đây. “Các nhà hóa học rất bận rộn,” anh ta phản đối, “họ không có

thời gian để vẽ ra các chi tiết trước khi cô bắt đầu xây dựng hệ thống của mình. Người của cô không làm được việc này sao?"

Cynthia cố gắng giải thích lý lẽ của cô trong việc thu thập yêu cầu từ những người sẽ sử dụng hệ thống mới, "Nếu chúng tôi làm những gì mà chúng tôi cho rằng người dùng cần thì chúng tôi không thể làm ra một hệ thống tốt được. Chúng tôi là những nhà phát triển phần mềm, vì vậy chúng tôi không thật sự biết các nhà hóa học cần gì để làm ra hệ thống giám sát hóa chất cho họ được. Tôi đã học được rằng nếu chúng tôi không mất thời gian để tìm hiểu vấn đề trước khi viết mã nguồn thì sẽ không ai hài lòng về sản phẩm phần mềm làm ra."

"Thôi nào, chúng ta không có thời gian cho những việc ấy," Gerhard khẳng định ý kiến của mình, "Tôi vừa cho cô yêu cầu rồi. Nào bắt đầu xây dựng hệ thống của cô đi. Hãy nhớ tiến độ tôi yêu cầu đấy."

Cuộc trao đổi trên là điển hình trong ngành công nghiệp phần mềm. Khách hàng đề nghị xây dựng một hệ thống thông tin mới nhưng thường không hiểu tầm quan trọng của việc có được đầu vào từ những người dùng tương lai của hệ thống. Không gì có thể thay thế cho việc thu thập yêu cầu trực tiếp từ những người sẽ sử dụng hệ thống. Một nghiên cứu trên 8380 dự án đã chỉ ra rằng hai nguyên nhân chính khiến dự án bị trục trặc là thiếu thông tin đầu vào từ người dùng và các yêu cầu, đặc tả không đầy đủ (Standish 1995).

Nguyên nhân của sự bối rối đối với đa số những người liên quan đến việc xây dựng một hệ thống thông tin là phân biệt sự khác nhau giữa các mức yêu cầu: kinh doanh, người dùng, chức năng (business, user, functional). Gerhard đã xác định một số yêu cầu kinh doanh, nhưng anh ta không thể mô tả được yêu cầu người dùng vì anh ta không phải là người sử dụng của hệ thống này. Những người dùng tiềm năng mới có thể mô tả các tác vụ mà họ cần phải làm với sự trợ giúp của hệ thống, nhưng họ lại không thể xác định tất cả các yêu cầu chức năng cụ thể cần phải được cài đặt để cho phép họ hoàn thành các tác vụ đó.

Chương này làm rõ mối quan hệ giữa khách hàng và nhà phát triển, quan hệ đóng vai trò then chốt trong sự thành công của một dự án phần mềm. Tôi đề xuất một Cuốn sách này thuộc "Tủ sách Công nghệ thông tin", tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Dự luật Yêu cầu về Quyền của Khách hàng Phần mềm (Requirements Bill of Rights for Software Customers) và một Dự luật Yêu cầu về Trách nhiệm của Khách hàng Phần mềm (Requirements Bill of Responsibilities for Software Customers) tương ứng nhằm nhấn mạnh tầm quan trọng của khách hàng (và người dùng nói riêng) trong quy trình phát triển yêu cầu.

I. AI LÀ KHÁCH HÀNG? (WHO IS THE CUSTOMER?)

Trong nghĩa rộng nhất của nó, một khách hàng là một cá nhân hoặc một tổ chức được hưởng trực tiếp hoặc gián tiếp lợi ích từ việc sử dụng một sản phẩm. Khách hàng phần mềm gồm tất cả những người liên quan đến dự án, đó là những người thanh toán tiền mua phần mềm, chọn lựa, đặc tả, sử dụng phần mềm, những người sử dụng thông tin đầu ra của phần mềm.

Gerhard đại diện cho kiểu khách hàng thanh toán, mua sắm hoặc tài trợ một dự án phần mềm. Các khách hàng mức cao như Gerhard chịu trách nhiệm đặc tả các yêu cầu kinh doanh (business requirements). Họ cung cấp các ý tưởng ở mức cao về sản phẩm và lý do, sự hợp lý để khởi động dự án sản xuất sản phẩm đó. (*Theo kinh nghiệm của tôi, đây là loại khách hàng quan trọng nhất, họ quyết định lý do về mặt kinh doanh tại sao lại có dự án này và đó cũng là lý do tại sao lại phải tiêu vốn cho dự án này. Một thay đổi quyết định của họ sau này có thể làm ngưng lại cả một dự án đang tiến hành, dù tiền chưa được chuyển vào tài khoản của nhà phát triển phần mềm – ND*).

Như thảo luận trong Chương 6, các yêu cầu kinh doanh (business requirements) mô tả mục tiêu (objectives) mà khách hàng, doanh nghiệp, hoặc những người liên quan khác mong muốn đạt được, hoặc các giá trị mà họ muốn nhận được từ hệ thống. Business requirements tạo ra định hướng cho dự án. Sau đó thì bắt cứ cái gì được đặc tả như một yêu cầu đều cần nhất quán với các business requirements, cũng phải nhất quán như vậy đối với các tính năng của phần mềm. Tuy nhiên, business requirements không cung cấp các chi tiết đủ để các nhà phát triển biết cái họ cần làm là gì (what to build).

Mức yêu cầu tiếp theo – user requirements - cần được thu thập từ những người sẽ trực tiếp sử dụng hệ thống. Những người dùng đó (thường được gọi là người dùng Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

cuối – end users) là một kiểu khách hàng khác của hệ thống. Họ có thể mô tả cả các tác vụ (tasks) mà họ cần để thực thi với sự giúp đỡ của hệ thống và các đặc tính phi chức năng quan trọng để hệ thống có thể được chấp nhận.

Trong số các kiểu khách hàng thì:

- Các khách hàng là người trả tiền cung cấp các thông tin về lý do cần xây dựng hệ thống, hợp đồng, hoặc phát triển ứng dụng tùy biến (custom application), tức là các business requirements.
- Các khách hàng là người ấn vào các phím máy tính hàng ngày để sử dụng hệ thống cung cấp các thông tin gọi là user requirements.

Không may là cả hai kiểu khách hàng đều cảm thấy họ không có đủ thời gian để làm việc với các nhà phân tích yêu cầu, những người thu thập, phân tích và tài liệu hoá các yêu cầu. Đôi khi khách hàng kỳ vọng các nhà phân tích hoặc các nhà phát triển phác ra cái người dùng cần mà không phải mất nhiều thời gian để thảo luận và tài liệu hoá. Nếu chỉ thế thôi thì dễ. Nếu doanh nghiệp của bạn phụ thuộc nhiều vào sự thành công của việc ứng dụng một phần mềm nào đó thì bạn phải chấp nhận mất nhiều thời gian để làm việc với các nhà phân tích hệ thống.

Tình trạng sẽ hơi khác đối với việc phát triển các phần mềm thương mại (các phần mềm đóng gói), khi đó khách hàng (customer) và người dùng (user) chỉ là một người. Các đại diện cho khách hàng, ví dụ bộ phận marketing của chính doanh nghiệp của bạn, sẽ phải cố gắng xác định cái gì khiến cho những người thanh toán tiền cảm thấy thích thú từ sản phẩm phần mềm doanh nghiệp bạn làm ra. Thậm chí đối với các phần mềm thương mại, chính bạn phải trở thành người dùng và Chương 7 sẽ nói chi tiết về việc này, nếu không thì bạn hãy chuẩn bị tinh thần đọc những đánh giá trên các tạp chí chuyên ngành về các hạn chế của phần mềm của bạn.

II. SỰ CỘNG TÁC KHÁCH HÀNG – NHÀ PHÁT TRIỂN (THE CUSTOMER – DEVELOPER PARTNERSHIP)

Các sản phẩm phần mềm chất lượng cao thường là kết quả của các thiết kế được thực thi tốt trên cơ sở các yêu cầu tuyệt hảo. Các yêu cầu như vậy là kết quả của việc cộng tác và truyền thông tốt giữa nhà phát triển và khách hàng.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Dự luật Yêu cầu về Quyền của Khách hàng Phần mềm (Requirements Bill of Rights for Software Customers) liệt kê ra 10 kỳ vọng của khách hàng đối với nhà phân tích và nhà phát triển trong các hoạt động của quy trình yêu cầu. Mỗi trong số các quyền đó ngụ ý một trách nhiệm tương ứng về phía nhà phát triển và nhà phân tích. **Dự luật Yêu cầu về Trách nhiệm của Khách hàng Phần mềm (Requirements Bill of Responsibilities for Software Customers)** liệt kê ra 10 trách nhiệm của khách hàng đối với nhà phát triển trong quy trình yêu cầu. Nếu bạn thích, bạn có thể gọi đó là dự luật về quyền của nhà phát triển.

DỰ LUẬT YÊU CẦU VỀ QUYỀN CỦA KHÁCH HÀNG PHẦN MỀM	
Nếu bạn là Khách hàng Phần mềm thì bạn có quyền:	
1.	Kỳ vọng nhà phân tích nói bằng ngôn ngữ của bạn.
2.	Kỳ vọng nhà phân tích nắm được nghiệp vụ kinh doanh và mục tiêu mà bạn đặt ra cho hệ thống.
3.	Kỳ vọng nhà phân tích cấu trúc hoá được thông tin mà bạn cung cấp thành một bản đặc tả yêu cầu phần mềm thành văn.
4.	Đề nghị nhà phát triển diễn giải cho bạn tất cả các bán thành phẩm được tạo ra trong quy trình yêu cầu.
5.	Kỳ vọng nhà phát triển có thái độ tôn trọng và duy trì sự hợp tác với bạn trong suốt quá trình làm việc chung.
6.	Đề nghị nhà phát triển cung cấp cho bạn tất cả các ý tưởng và các lựa chọn mà anh ta có thể có về yêu cầu và sự thi công hệ thống từ các yêu cầu đó.
7.	Mô tả các đặc tính của sản phẩm khiến cho nó dễ sử dụng và thân thiện hơn với người dùng.
8.	Có cơ hội điều chỉnh các yêu cầu nhằm có thể sử dụng lại các software components mà bạn đã có.
9.	Có được các ước lượng tương đối tốt về chi phí, ảnh hưởng và các đánh đổi (trade-offs) khi bạn đề nghị một thay đổi trong số các yêu cầu.
10.	Nhận được một hệ thống đáp ứng các nhu cầu của bạn về chức năng và chất lượng, đáp ứng sự mở rộng các nhu cầu đã được trao đổi và thoả thuận với nhà phát triển.

DỰ LUẬT YÊU CẦU VỀ TRÁCH NHIỆM CỦA KHÁCH HÀNG PHẦN MỀM

Nếu bạn là Khách hàng Phần mềm thì bạn có trách nhiệm:

1. Giúp nhà phân tích hiểu về nghiệp vụ kinh doanh của bạn và định nghĩa các biệt ngữ nghiệp vụ (business jargon) cho họ hiểu.
2. Sẵn sàng tiêu tốn thời gian để cung cấp yêu cầu, làm sáng tỏ chúng và bổ sung chúng thường xuyên.
3. Cụ thể và chính xác khi cung cấp nguồn gốc của yêu cầu.
4. Ra quyết định đúng lúc về các công việc liên quan đến yêu cầu khi có đề nghị.
5. Tôn trọng sự đánh giá của nhà phát triển về chi phí và tính khả thi của yêu cầu.
6. Thiết lập ưu tiên cho các yêu cầu riêng lẻ, các tính năng hệ thống hoặc các tình huống sử dụng (use cases).
7. Soát xét các tài liệu yêu cầu và các nguyên mẫu (prototypes).
8. Truyền thông cho các bên liên quan về các thay đổi của yêu cầu ngay khi bạn biết được các thay đổi đó.
9. Tuân thủ quy trình đã được định nghĩa của công ty phát triển sản phẩm khi đề xuất các thay đổi yêu cầu.
10. Tôn trọng các quy trình mà nhà phát triển sử dụng cho công nghệ yêu cầu.

Các quyền và trách nhiệm trên được áp dụng trực tiếp đối với khách hàng trong các hợp đồng làm phần mềm đặt hàng riêng, với các phần mềm loại này, công ty sản xuất phần mềm đã biết chính xác khách hàng là ai. Còn trường hợp phát triển các sản phẩm cho thị trường đại chúng (mass market) thì quyền và trách nhiệm được áp dụng cho các đại diện của khách hàng như bộ phận marketing chẳng hạn.

Khi lập kế hoạch dự án, khách hàng và nhà phát triển cần phải soát xét kỹ 2 danh sách này để định ra một quy tắc làm việc chung. Các khách hàng bận rộn có thể không thích bị cuốn vào quy trình công nghệ yêu cầu (như quy định của Trách nhiệm 2). Thiếu sự tham gia của khách hàng nhất định sẽ tăng nguy cơ sản xuất một sản phẩm sai. Hãy đảm bảo rằng tất cả những thành viên chính trong quy trình công nghệ yêu cầu đều hiểu và chấp nhận trách nhiệm của họ. Nếu phải đối mặt với một số điểm gây căng thẳng thì hãy đàm phán để đi đến hiểu biết chung. Sự *Cuốn sách* này thuộc “*Tủ sách Công nghệ thông tin*”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

hiểu biết chung này sẽ giảm các xích mích không đáng có khi một bên kỳ vọng vào một cái gì đó mà bên kia không mong muốn hoặc không thể đáp ứng.

Sau đây chúng ta sẽ xem xét kỹ hai dự luật trên.

1. DỰ LUẬT YÊU CẦU VỀ QUYỀN CỦA KHÁCH HÀNG PHẦN MỀM

Quyền 1: Kỳ vọng nhà phân tích nói bằng ngôn ngữ của bạn

Các thảo luận về yêu cầu cần tập trung vào các nghiệp vụ kinh doanh và các tác vụ của người dùng, trong khi thảo luận thì phải sử dụng từ vựng của chuyên ngành của bạn, bạn cần truyền đạt đến nhà phát triển đề nghị này. Bạn không cần phải hiểu về các biệt ngữ công nghệ thông tin khi trao đổi với nhà phân tích.

Quyền 2: Kỳ vọng nhà phân tích nắm được nghiệp vụ kinh doanh và mục tiêu mà bạn đặt ra cho hệ thống

Bằng cách tương tác, trao đổi với người dùng khi suy luận yêu cầu, nhà phân tích có thể hiểu tốt hơn nghiệp vụ của bạn và biết cách làm thế nào để làm ra sản phẩm thích hợp với bạn, đáp ứng tốt nhu cầu và sự kỳ vọng của bạn. Để giúp các nhà phát triển nắm vững nghiệp vụ của bạn, hãy mời họ đến quan sát những gì bạn và các đồng nghiệp của mình làm. Nếu hệ thống được xây dựng để thay thế ứng dụng đã có thì nhà phát triển phải sử dụng hệ thống hiện có như bạn đã sử dụng nó, điều đó giúp họ rút ra các kết luận bổ ích.

Quyền 3: Kỳ vọng nhà phân tích cấu trúc hóa được thông tin mà bạn cung cấp thành một bản đặc tả yêu cầu phần mềm thành văn

Nhà phân tích sẽ sắp xếp tất cả thông tin bạn và các khách hàng khác cung cấp nhằm phân loại nhu cầu người dùng thành các business requirements và các quy tắc (rules), các functional requirements, các mục tiêu chất lượng, các ý tưởng giải pháp và các thông tin khác. Sản phẩm cuối từ sự phân tích này là một **đặc tả yêu cầu phần mềm (Software Requirement Specification, SRS)**. SRS cấu thành trên thoả thuận giữa nhà phát triển và khách hàng về nội dung của sản phẩm sẽ được xây dựng. SRS cần phải được cấu trúc và viết ra dưới dạng mà bạn có thể dễ dàng đọc và hiểu. Bạn có thể soát xét các đặc tả thành văn đó để đảm bảo chắc chắn rằng nó biểu diễn chính xác và đầy đủ các yêu cầu của bạn. Một SRS chất lượng cao thúc đẩy mạnh mẽ cơ hội xây dựng sản phẩm bạn thực sự muốn.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Quyền 4: Đề nghị nhà phát triển diễn giải cho bạn tất cả các bán thành phẩm được tạo ra trong quy trình yêu cầu

Nhà phân tích cần phải biểu diễn các yêu cầu bằng cách sử dụng nhiều sơ đồ khác nhau nhằm làm sáng tỏ hơn SRS. Các sơ đồ đó thật sự có giá trị trong việc biểu diễn một số hành vi của hệ thống, ví dụ các luồng công việc (workflows). Có thể bạn không quen thuộc lắm với các sơ đồ nhưng bạn cũng không khó khăn lắm để hiểu nó. Bạn có thể đề nghị nhà phân tích giải thích mục đích của mỗi sơ đồ hoặc các bán thành phẩm (work products) khác dẫn xuất từ yêu cầu, ý nghĩa của các ký hiệu (notation) và làm thế nào để kiểm tra các sơ đồ nhằm tìm kiếm và loại bỏ lỗi và sự thiếu nhất quán.

Quyền 5: Kỳ vọng nhà phát triển có thái độ tôn trọng và duy trì sự hợp tác với bạn trong suốt quá trình làm việc chung

Các thảo luận về yêu cầu có thể được làm tốt dần nếu người dùng và nhà phát triển chưa nắm bắt được các yêu cầu ngay từ đầu. Làm việc nhóm cùng nhau sẽ khiến cả khách hàng và nhà phát triển đều hiểu vấn đề sâu hơn.

Quyền 6: Đề nghị nhà phát triển cung cấp cho bạn tất cả các ý tưởng và các lựa chọn mà anh ta có thể có về yêu cầu và sự thi công hệ thống từ các yêu cầu đó

Thông thường khách hàng đề xuất các ý tưởng về “yêu cầu” (“requirements” ideas) là các giải pháp có thể thực thi (possible implementation solutions). Các nhà phân tích sẽ cố gắng tìm hiểu dần sau các giải pháp đó để hiểu được các vấn đề nghiệp vụ thực sự và nhu cầu cần được chỉ mặt, đặt tên (*nghĩa là cố gắng tìm hiểu mục đích nào mà khách hàng muốn đạt được sau khi thực hiện giải pháp đó - ND*). Các nhà phân tích cần duyệt qua tất cả những gì mà hệ thống hiện tại không đáp ứng được các quy trình kinh doanh hiện thời, nhằm đảm bảo rằng sản phẩm làm ra vượt qua được các nhược điểm đó. Nhà phân tích là người hiểu được lĩnh vực nghiệp vụ (business domain, *thuật ngữ chỉ miền ứng dụng của sản phẩm phần mềm, ví dụ tài chính là một business domain - ND*) và có thể đề xuất các cải tiến cần thiết, đồng thời anh ta cũng có thể thêm vào phần mềm mới các tính năng có giá trị mà người dùng cũng chưa mường tượng ra được.

Quyền 7: Mô tả các đặc tính của sản phẩm khiến cho nó dễ sử dụng và thân thiện hơn với người dùng

Bạn có thể kỳ vọng các nhà phân tích hỏi bạn các đặc tính mà phần mềm cần có để nó dễ sử dụng hơn. Các đặc tính đó, hoặc các thuộc tính chất lượng, khiến cho phần mềm thân thiện hơn và giúp bạn hoàn thành các tác vụ (tasks) của mình chính xác và hiệu quả hơn. Ví dụ, khách hàng đôi khi hay nói rằng sản phẩm phải “thân thiện người dùng” (user-friendly) hoặc “ổn định” (robust), “hiệu quả” (efficient), nhưng những yêu cầu như vậy không giúp gì nhiều cho nhà phát triển vì chúng khá cảm tính. Thay vì vậy, nhà phân tích cần cụ thể hoá thế nào thì được gọi là “thân thiện người dùng” hoặc “ổn định”, “hiệu quả” đối với khách hàng (Chương 11).

Quyền 8: Có cơ hội điều chỉnh các yêu cầu nhằm có thể sử dụng lại các software components mà bạn đã có

Yêu cầu thường cần một mức độ mềm dẻo nào đó. Nhà phân tích cần nhận thức được rằng các software components đang có đã đáp ứng một số nhu cầu hiện tại của bạn. Trong trường hợp đó, nhà phân tích cần đề xuất lựa chọn sửa đổi yêu cầu của bạn sao cho có thể sử dụng lại các components đã có trong hệ thống mới. Nếu bạn có thể điều chỉnh yêu cầu của mình sao cho nhu cầu vẫn được đáp ứng và sử dụng lại được các components đã có thì bạn sẽ tiết kiệm được nhiều thời gian và tiền bạc hơn. Một số yêu cầu phải khá mềm dẻo để bạn có thể sử dụng được các COTS components (*là software components được các công ty phần mềm sản xuất sẵn và được thương mại hóa, ví dụ đối với lĩnh vực xây dựng thì gạch lát nền nhà là một dạng tương tự COST component - ND*).

Quyền 9: Có được các ước lượng tương đối tốt về chi phí, ảnh hưởng và các đánh đổi (trade-offs) khi bạn đề nghị một thay đổi trong số các yêu cầu

Đôi khi người ta thực hiện các lựa chọn khác nhau khi chi phí giữa các lựa chọn là khác nhau. Cần có ước lượng về ảnh hưởng và chi phí của mỗi thay đổi được đề xuất đối với các yêu cầu để ra quyết định kinh doanh về các thay đổi này, nghĩa là nên lựa chọn thay đổi nào. Bạn có quyền kỳ vọng nhà phát triển đưa ra các ước lượng về ảnh hưởng, chi phí, và đánh đổi cho mỗi thay đổi. Nhà phát triển không được thổi phồng các chi phí được ước tính của một thay đổi vì lý do họ không muốn thực hiện sự thay đổi đó.

Quyền 10: Nhận được một hệ thống đáp ứng các nhu cầu của bạn về chức năng và chất lượng, đáp ứng sự mở rộng các nhu cầu đã được trao đổi và thỏa thuận với nhà phát triển

Bất cứ ai cũng mong muốn kết quả này cho dự án. Kết quả này chỉ có thể hiện thực nếu nhà phát triển được biết tất cả thông tin về các lựa chọn và ràng buộc cho phép họ làm ra sản phẩm đáp ứng tốt nhu cầu của bạn.

2. DỰ LUẬT YÊU CẦU VỀ TRÁCH NHIỆM CỦA KHÁCH HÀNG PHẦN MỀM

Trách nhiệm 1: Giúp nhà phân tích hiểu về nghiệp vụ kinh doanh của bạn và định nghĩa các biệt ngữ nghiệp vụ cho họ hiểu

Nhà phân tích kỳ vọng bạn đào tạo cho anh ta về nghiệp vụ kinh doanh và các thuật ngữ bạn dùng. Việc này không hề giúp anh ta trở thành chuyên gia trong lĩnh vực của bạn, nó chỉ giúp anh ta hiểu được những gì bạn nói, hiểu được mục tiêu của bạn. Đừng kỳ vọng nhà phân tích có thể nắm bắt sâu sắc các khía cạnh đa dạng trong nghiệp vụ của bạn.

Trách nhiệm 2: Sẵn sàng tiêu tốn thời gian để cung cấp yêu cầu, làm sáng tỏ chúng và bổ sung chúng thường xuyên

Khách hàng là những người bận rộn và thường những ai liên quan nhiều nhất đến quy trình yêu cầu lại là người bận rộn nhất. Bạn hãy cố gắng dành thời gian tham gia các hội thảo, các phiên họp tập kích não (brainstorming), các buổi phỏng vấn, các hoạt động khác liên quan đến yêu cầu. Đôi khi nhà phân tích cho rằng anh ta hiểu một điểm nào đó trong số các công việc của bạn nhưng chỉ khi triển khai sâu hơn công việc thì anh ta lại có nhu cầu làm sáng sủa vấn đề hơn. Hãy kiên nhẫn với cách tiếp cận lặp trong việc phát triển và định nghĩa yêu cầu, cũng vì bản chất của việc trao đổi thông tin qua lại giữa bạn và nhà phân tích là phức tạp, và vì sự thành công của dự án là quan trọng nhất.

Trách nhiệm 3: Cụ thể và chính xác khi cung cấp nguồn gốc của yêu cầu

Trình bày rõ, chính xác các yêu cầu là việc khó khăn, vì vậy bất cứ lúc nào nhà phân tích cũng cần gặp ai đó trong số khách hàng để được giúp đỡ phân giải các yêu cầu nhập nhằng và thiếu chính xác. Nên tạm đánh dấu TBD (cần được xác định – to be determined) trên một yêu cầu nào đó trong bản đặc tả yêu cầu, nghĩa là Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

ta cần các nghiên cứu, phân tích bổ sung, thêm các thông tin cần thiết đối với yêu cầu đó – yêu cầu chưa được hiểu rõ tại thời điểm ấy. Đôi khi, TBD được sử dụng trên một yêu cầu nào đó vì nó đã được xác định là rất khó khăn để giải quyết và không ai muốn tiếp tục xử lý nó nữa.

Trách nhiệm 4: Ra quyết định đúng lúc về các công việc liên quan đến yêu cầu khi có đề nghị

Khi nhà phân tích phân giải yêu cầu thì anh ta phải đối mặt với nhiều lựa chọn và quyết định (choices and decisions), anh ta chỉ được chọn một với một chi phí tương ứng. Mỗi quyết định bao gồm phân giải các đề nghị thiếu nhất quán từ nhiều người dùng, thực hiện các đánh đổi giữa các thuộc tính chất lượng xung đột nhau, đánh giá sự chính xác của thông tin. Khách hàng chính là người ra quyết định tại mỗi thời điểm cần sự lựa chọn đó. Nhà phân tích thường phải chờ đợi quyết định của bạn tại mỗi thời điểm như vậy, tiến độ của dự án phụ thuộc một phần vào việc ra quyết định của bạn.

Trách nhiệm 5: Tôn trọng sự đánh giá của các nhà phát triển về chi phí và tính khả thi của yêu cầu

Tất cả chức năng của phần mềm đều có giá (price) và nhà phát triển là người ước lượng tốt nhất chi phí xây dựng chức năng đó (mặc dù nhiều nhà phát triển không phải là một người ước lượng có kỹ năng). Một số tính năng mà bạn muốn thêm vào sản phẩm nhưng việc đó không khả thi về mặt kỹ thuật hoặc quá đắt để thực thi. Một số yêu cầu là phi thực tế trong điều kiện hiện tại của doanh nghiệp của bạn. Nhà phát triển có thể là kẻ đưa tin xấu về tính khả thi hoặc chi phí của một yêu cầu nào đó, nhưng bạn nên tôn trọng lời nói của anh ta.

Đôi khi bạn có thể viết lại các yêu cầu theo một cách khiến chúng khả thi hoặc rẻ hơn để thực hiện. Ví dụ, yêu cầu “hệ thống phản ứng ngay lập tức” không khả thi nhưng nếu bạn viết yêu cầu “hệ thống phản ứng trong vòng 50 mili giây” thì lại khả thi.

Trách nhiệm 6: Thiết lập ưu tiên cho các yêu cầu riêng lẻ, các tính năng hệ thống hoặc các tình huống sử dụng (use cases)

Phần lớn các dự án không có thời gian hoặc tài nguyên để thực thi tất cả các yêu cầu cùng một lúc. Hãy xác định tính năng nào là quan trọng, cơ bản để xác định thứ tự ưu tiên trong việc xây dựng hệ thống. Bạn chính là người đưa ra thứ tự ưu tiên này vì nhà phát triển không thể làm điều đó từ góc độ của anh ta được. Nhà phát triển sẽ cung cấp thông tin chi phí và rủi ro của mỗi yêu cầu nhằm giúp bạn định nghĩa thứ tự này. Khi bạn đã thiết lập thứ tự ưu tiên này, bạn đã đảm bảo nhà phát triển chuyển giao cho bạn sản phẩm với giá trị lớn nhất, chi phí thấp nhất và tại thời điểm hợp lý nhất.

Thứ tự ưu tiên này giúp bạn dễ co giãn được phạm vi dự án tại từng thời điểm nhất định căn cứ trên nhu cầu thực tế của doanh nghiệp, trên thời gian và ngân sách của doanh nghiệp.

Trách nhiệm 7: Soát xét các tài liệu yêu cầu và các nguyên mẫu (prototypes)

Như bạn sẽ thấy trong Chương 14, hoạt động soát xét (reviews) tài liệu yêu cầu là một trong số các hoạt động có giá trị nhất ảnh hưởng đến chất lượng phần mềm. Khách hàng cần tham gia vào hoạt động này vì đó là cách duy nhất để đánh giá liệu bản mô tả yêu cầu đã xác định các đặc tính của phần mềm một cách đầy đủ, đúng đắn và cần thiết hay chưa. Mỗi phiên soát xét sẽ là một cơ hội cho các đại diện của khách hàng phản hồi cho các nhà phân tích về công việc của họ, liệu họ đã đáp ứng đúng nhu cầu dự án chưa. Nếu bạn không tin tài liệu yêu cầu là chính xác, hãy nói với người có trách nhiệm càng sớm càng tốt và đưa ra các đề nghị cải tiến.

Rất khó khăn để có thể phác ra một bức tranh sống động về hoạt động của phần mềm chỉ bằng cách đọc một đặc tả yêu cầu. Để hiểu tốt hơn nhu cầu của bạn và đề xuất những cách tốt nhất để đáp ứng các nhu cầu đó, nhà phát triển thường xây dựng các nguyên mẫu (*tương tự bây giờ khi bán căn hộ chung cư cho khách hàng, công ty xây dựng cho khách hàng xem căn hộ mẫu trước - ND*) cho sản phẩm mong muốn. Phản hồi của bạn về nguyên mẫu sẽ cung cấp các thông tin rất giá trị cho nhà phát triển và giúp họ hiểu tốt hơn yêu cầu. Nguyên mẫu không phải là một sản phẩm thực sự và nhà phát triển cũng không biến đổi nguyên mẫu thành hệ thống đầy đủ được.

Trách nhiệm 8: Truyền thông cho các bên liên quan các thay đổi về yêu cầu ngay khi bạn biết được các thay đổi đó

Sự thay đổi liên tục của yêu cầu gây ra rủi ro nghiêm trọng đối với nhà phát triển trong việc chuyển giao một sản phẩm chất lượng cao cho khách hàng với một lịch biểu đã định. Thay đổi là không thể tránh khỏi, nhưng càng về cuối chu trình phát triển thì một thay đổi càng ảnh hưởng nhiều, chúng khiến nhiều công việc phải làm lại và chi phí phát sinh là rất lớn. Vậy hãy thông báo cho nhà phát triển ngay khi bạn thấy cần thay đổi yêu cầu, thông báo càng sớm càng tốt.

Trách nhiệm 9: Tuân thủ quy trình đã được định nghĩa của công ty phát triển sản phẩm khi đề xuất các thay đổi yêu cầu

Để tối thiểu hoá các ảnh hưởng tiêu cực của thay đổi, tất cả những người tham gia dự án đều phải tuân theo quy trình kiểm soát thay đổi đã được định nghĩa của dự án. Điều này đảm bảo các thay đổi đề xuất không bị mất tích, các ảnh hưởng của thay đổi được phân tích đầy đủ và được cân nhắc theo một cách nhất quán. (*Trong các dự án phần mềm, các thay đổi yêu cầu rất hay bị mất tích, tuần trước rõ ràng đã sửa chữa năng đó theo yêu cầu được thay đổi rồi nhưng tuần sau cập nhật phiên bản mới lại thấy chức năng đó vẫn như cũ, chưa hề được sửa –ND*).

Trách nhiệm 10: Tôn trọng các quy trình mà nhà phát triển sử dụng cho công nghệ yêu cầu

Thu thập yêu cầu và đảm bảo chúng đúng đắn là thách thức lớn nhất của phát triển phần mềm. Có một sự hợp lý ẩn đằng sau cách tiếp cận xây dựng yêu cầu của nhà phân tích. Mặc dù bạn vẫn có thể bổ sung và làm mịn các yêu cầu trong giai đoạn sau nhưng thời gian tiêu tốn cho giai đoạn phát triển yêu cầu thật sự là một khoản đầu tư hữu ích. Quy trình thu thập yêu cầu sẽ bớt chông gai hơn nếu bạn hiểu và tôn trọng các kỹ thuật nhà phân tích dùng để thu thập, tài liệu hoá và đảm bảo chất lượng yêu cầu. Hãy cứ đề nghị nhà phân tích diễn giải cặn kẽ cho bạn tất cả những gì bạn chưa rõ trong quy trình yêu cầu đó.

III. DẤU HIỆU DÙNG LẠI LÀ GÌ? (WHAT ABOUT SIGH-OFF?)

Đạt được thoả thuận về yêu cầu là một phần quan trọng trong sự cộng tác giữa khách hàng – nhà phát triển. Nhiều tổ chức sử dụng khái niệm “signing off” để chỉ việc khách hàng chấp nhận và thông qua văn bản yêu cầu. Điều quan trọng là tất cả Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

những người tham gia quy trình chấp thuận yêu cầu (requirements approval process) phải biết chính xác sign-off nghĩa là gì.

Quá trình thu thập và phân tích yêu cầu tạm thời dừng lại khi nhà phát triển và khách hàng thống nhất được một ranh giới (baseline) của bản đặc tả yêu cầu. Khi ký xác nhận ranh giới đó thì bạn cần thêm vào SRS đoạn văn sau: “Tôi xác nhận rằng tài liệu mô tả yêu cầu này là hiểu biết tốt nhất của chúng tôi về các yêu cầu phần mềm cho dự án tính đến ngày hôm nay. Các thay đổi tương lai đối với ranh giới này sẽ được thực hiện thông qua quy trình thay đổi đã được định nghĩa của dự án. Tôi nhận thức rằng các thay đổi đã được chấp thuận này có thể vẫn khiến chúng tôi đàm phán lại các cam kết về chi phí, nguồn lực, lịch biểu của dự án.”

Các bước tiếp theo

- Xác định các khách hàng cá nhân có trách nhiệm cung cấp business requirements và user requirements trong dự án của bạn. Mỗi mục nào trong Bill of Rights và Bill of Responsibilities đã được chấp thuận, được hiểu, được thực hiện bởi các khách hàng đó? Mục nào không?
- Hãy thảo luận về Bill of Rights và Bill of Responsibilities với các khách hàng chính để đạt được sự thỏa thuận trách nhiệm nào sẽ được chấp thuận và họ cảm thấy gì khi không nhận được một quyền nào đó. Dựa vào đó hãy đề ra các biện pháp cải thiện tốt hơn quan hệ cộng tác Khách hàng – Nhà phát triển.
- Nếu bạn là một khách hàng tham gia vào một dự án phần mềm và bạn không cảm thấy các quyền của bạn được tôn trọng thì hãy thảo luận về Bill of Rights với các nhà phát triển. Hãy xác nhận trách nhiệm của mình trong Bill of Responsibilities để từ đó đề nghị họ thực hiện trách nhiệm của họ.

CHƯƠNG 3

GOOD PRACTICES CHO CÔNG NGHỆ YÊU CẦU

Mười năm trước đây, tôi rất hâm mộ các phương pháp luận phát triển phần mềm - tập hợp các mô hình và kỹ thuật nhằm giúp tôi giải quyết các thách thức của dự án phần mềm. Ngày nay, tôi quan tâm hơn đến việc xác định và ứng dụng cái được gọi là “best practices”. Thuật ngữ “best practices”, mặc dù còn gây nhiều tranh cãi, nhưng nó có ý nghĩa là: anh nhận được một lời khuyên gọi là “best” và làm theo lời khuyên đó anh đạt được kết quả tốt nhất cho công việc của mình. Cách tiếp cận này là sự tổng kết thành công và thất bại của các chuyên gia trong rất nhiều dự án ở rất nhiều tổ chức khác nhau (Brown 1996). Từ đó họ rút ra được các yếu tố chung, tổng quát mang lại thành công cho nhiều dự án khác nhau, các yếu tố đó gọi là “best practices” (hướng dẫn thực hành tốt nhất). Tuy nhiên, đầu đề của chương này là “**Good practices cho công nghệ yêu cầu**” chứ không phải “best practices”. Chương này sẽ trình bày hơn 40 practices trong 7 nhóm khác nhau.

Bảng 3-1: GOOD PRACTICES CHO CÔNG NGHỆ YÊU CẦU

Tri thức (Knowledge)	Quản lý yêu cầu (Requirements Management)	Quản lý dự án (Project Management)
<ul style="list-style-type: none">Đào tạo các nhà phân tích yêu cầu (Train requirements analysis)Giáo dục cho đại diện người dùng và các nhà quản lý về yêu cầu (Educate user reps and managers about requirements)Đào tạo các nhà phát triển trong	<ul style="list-style-type: none">Định nghĩa quy trình kiểm soát thay đổi (Define change control process)Thành lập ban kiểm soát thay đổi (Establish change control board)Thực hiện các phân tích về ảnh hưởng của thay đổi (Perform change impact analysis)Lần vết mỗi thay đổi đối với tất cả các bản thành	<ul style="list-style-type: none">Lựa chọn một vòng đời thích hợp (Select appropriate life cycle)Lập kế hoạch dự án dựa trên yêu cầu (Base plans on requirements)Đàm phán nhiều lần về các cam kết (Renegotiate commitments)Quản lý rủi ro của yêu cầu (manage

miền ứng dụng (Train developers in application domain)	phẩm bị ảnh hưởng (Trace each change to all affected work products)	requirements risks)
<ul style="list-style-type: none"> Tạo ra một bảng thuật ngữ (Create a glossary) 	<ul style="list-style-type: none"> Vạch ranh giới và kiểm soát các phiên bản của tài liệu yêu cầu (Baseline and control versions of requirements documents) Duy trì lịch sử thay đổi (Maintain change history) Giám sát tình trạng yêu cầu (Track requirements status) Đo lường sự ổn định của yêu cầu (Measure requirements stability) Sử dụng một công cụ quản lý yêu cầu (Use a requirements management tool) 	<ul style="list-style-type: none"> Giám sát nhân lực thực hiện yêu cầu (Track requirements effort)

Bảng 3-1: GOOD PRACTICES CHO CÔNG NGHỆ YÊU CẦU (Tiếp)
Phát triển yêu cầu (Requirements Development)

Suy luận (Elicitation)	Phân tích (Analysis)	Đặc tả (Specification)	Kiểm tra (Verification)
<ul style="list-style-type: none"> Viết tầm nhìn và phạm vi (Write vision and scope) 	<ul style="list-style-type: none"> Vẽ sơ đồ bối cảnh của bài toán (Draw context) 	<ul style="list-style-type: none"> Thông qua mẫu SRS (Adopt SRS template) 	<ul style="list-style-type: none"> Thanh tra tài liệu yêu cầu (Inspect requirements)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

<ul style="list-style-type: none"> Định nghĩa thủ tục phát triển yêu cầu (Define requirements development procedure) Xác định các lớp người dùng (Identify user classes) Lựa chọn sự trợ giúp sản phẩm (Select product champions) Thành lập nhóm tập trung (Establish focus groups) Xác định các use cases (Identify use cases) Tổ chức các phiên JAD (Hold JAD sessions) Phân tích luồng công việc người dùng (Analyze user workflows) Định nghĩa các thuộc tính chất 	<ul style="list-style-type: none"> diagram) Tạo ra các nguyên mẫu (Create prototypes) Phân tích tính khả thi (Analyze feasibility) Xếp thứ tự ưu tiên các yêu cầu (Prioritize requirements) Mô hình hóa các yêu cầu (Model the requirements) Tạo một từ điển dữ liệu (Create a data dictionary) Ứng dụng Quality Function Deployment (Apply Quality Function Deployment) 	<ul style="list-style-type: none"> Xác định nguồn gốc yêu cầu (Identify sources of requirements) Gán nhãn mỗi yêu cầu (Label each requirement) Ghi lại các quy tắc nghiệp vụ (Record business rules) Tạo ra ma trận có thể lắn vết yêu cầu (Create requirements traceability matrix) 	<ul style="list-style-type: none"> document) Viết các test cases từ các yêu cầu (Write test cases from requirements) Viết tài liệu hướng dẫn người dùng (Write a user manual) Định nghĩa tiêu chuẩn chấp nhận (Define acceptance criteria)
--	---	--	--

Cuốn sách này thuộc "Tủ sách Công nghệ thông tin", tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

lượng (Define quality attributes) <ul style="list-style-type: none"> • Kiểm tra các báo cáo vấn đề (Examine problem reports) • Sử dụng lại yêu cầu (Reuse requirements) 			
--	--	--	--

Không phải tất cả các mục trên đều được tán thành là các best practices trong công nghiệp phần mềm (industry best practices). Tôi không cho rằng tất cả các mục đó đều được đánh giá một cách hệ thống vì mục đích này. Tuy nhiên, tôi và nhiều đồng nghiệp khác đã thấy các kỹ thuật này hiệu quả (Sommerville and Sawyer 1997). Mỗi practice sẽ được mô tả ngắn gọn và chỉ ra chương nói về nó trong cuốn sách này hoặc các nguồn tham khảo khác.

Bảng 3-2 nhóm các practices đó theo một thứ tự ưu tiên tương đối khi thi công dự án và độ khó tương đối khi ứng dụng các practices đó. Khi tất cả các practices đó phát huy tác dụng thì bạn có thể gặt hái kết quả - xác suất sự thành công của dự án sẽ lớn hơn.

Bảng 3-2: THỰC THI CÔNG NGHỆ YÊU CẦU (IMPLEMENTING REQUIREMENTS ENGINEERING)			
Ưu tiên (Priority)	Good Practices		
	Độ khó (Difficulty)	High	Medium
High	<ul style="list-style-type: none"> • Định nghĩa thủ tục phát triển yêu cầu (Define requirements) 	<ul style="list-style-type: none"> • Xác định các use cases (Identify use cases) • Định nghĩa các 	<ul style="list-style-type: none"> • Đào tạo các nhà phát triển trong miền ứng dụng (Train developers in

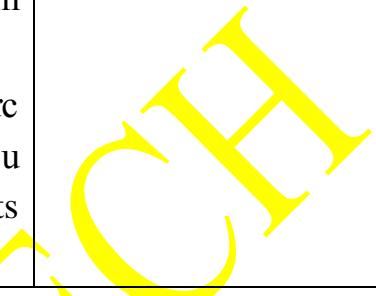
Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

	<p>development procedure)</p> <ul style="list-style-type: none"> • Lập kế hoạch dự án dựa trên yêu cầu (Base plans on requirements) • Đàm phán nhiều lần về các cam kết (Renegotiate commitments) 	<p>thuộc tính chất lượng (Define quality attributes)</p> <ul style="list-style-type: none"> • Xếp thứ tự ưu tiên các yêu cầu (Prioritize requirements) • Thông qua mẫu SRS (Adopt SRS template) • Định nghĩa quy trình kiểm soát thay đổi (Define change control process) • Thành lập ban kiểm soát thay đổi (Establish change control board) • Thanh tra tài liệu yêu cầu (Inspect requirements document) 	<p>application domain)</p> <ul style="list-style-type: none"> • Viết tầm nhìn và phạm vi (Write vision and scope) • Xác định các lớp người dùng (Identify user classes) • Vẽ sơ đồ bối cảnh của bài toán (Draw context diagram) • Xác định nguồn gốc yêu cầu (Identify sources of requirements) • Gán nhãn mỗi yêu cầu (Label each requirement) • Vạch ranh giới và kiểm soát các phiên bản của tài liệu yêu cầu (Baseline and control versions of requirements documents)
Medium	<ul style="list-style-type: none"> • Giáo dục cho đại diện người dùng và các nhà quản lý về yêu cầu (Educate user reps and managers about requirements) 	<ul style="list-style-type: none"> • Đào tạo các nhà phân tích yêu cầu (Train requirements analysis) • Thành lập nhóm tập trung (Establish focus groups) • Tạo ra các nguyên 	<ul style="list-style-type: none"> • Tạo ra một bảng thuật ngữ (Create a glossary) • Lựa chọn sự trợ giúp sản phẩm (Select product champions) • Tạo một từ điển dữ liệu (Create a data

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

	<ul style="list-style-type: none"> Mô hình hoá các yêu cầu (Model the requirements) Quản lý rủi ro của yêu cầu (manage requirements risks) Sử dụng một công cụ quản lý yêu cầu (Use a requirements management tool) Tạo ra ma trận có thể lằn vết yêu cầu (Create requirements traceability matrix) 	<ul style="list-style-type: none"> mẫu (Create prototypes) Phân tích tính khả thi (Analyze feasibility) Định nghĩa tiêu chuẩn chấp nhận (Define acceptance criteria) Thực hiện các phân tích về ảnh hưởng của thay đổi (Perform change impact analysis) Lần vết mỗi thay đổi đối với tất cả các sản phẩm bị ảnh hưởng (Trace each change to all affected work products) Lựa chọn một vòng đời thích hợp (Select appropriate life cycle) 	<ul style="list-style-type: none"> dictionary) Ghi lại các quy tắc nghiệp vụ (Record business rules) Viết các test cases từ các yêu cầu (Write test cases from requirements) Giám sát tình trạng yêu cầu (Track requirements status)
Low	<ul style="list-style-type: none"> Tổ chức các phiên JAD (Hold JAD sessions) Sử dụng lại yêu cầu (Reuse requirements) Ứng dụng 	<ul style="list-style-type: none"> Phân tích luồng công việc người dùng (Analyze user workflows) Kiểm tra các báo cáo vấn đề (Examine problem reports) 	

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

	<p>Quality Function Deployment (Apply Quality Function Deployment)</p> <ul style="list-style-type: none"> Đo lường sự ổn định của yêu cầu (Measure requirements stability) 	<ul style="list-style-type: none"> Viết tài liệu hướng dẫn người dùng (Write a user manual) Duy trì lịch sử thay đổi (Maintain change history) Giám sát nhân lực thực hiện yêu cầu (Track requirements effort) 	
--	---	---	---

Đừng cố gắng ứng dụng tất cả các kỹ thuật trên vào dự án sắp tới của bạn. Thay vì vậy, hãy suy nghĩ về các good practices đã mô tả ở đây như là các mục mới để thêm vào requirements tool kit của bạn. Bạn có thể bắt đầu bằng cách ứng dụng ngay một số practices, ví dụ các practices về quản trị thay đổi mà không cần quan tâm gì đến việc dự án của bạn tiến hành đến đâu.

Chương 4 giới thiệu các cách tiếp cận bạn có thể sử dụng để đánh giá các practices bạn đang dùng trong dự án hiện tại liên quan đến công nghệ yêu cầu và sáng tạo ra một lộ trình (road map) để thực hiện cải tiến quy trình yêu cầu được lựa chọn ra từ các practices ở đây (Chương 3) và ở Chương 4.

I. TRI THỨC (KNOWLEDGE)

Chỉ một số ít nhà phát triển phần mềm được đào tạo một cách chính thức các kỹ năng và kỹ thuật cần thiết cho công nghệ yêu cầu. Tuy vậy, trên thực tế một số nhà phát triển vẫn phải đóng vai trò phân tích viên yêu cầu khi phải làm việc với khách hàng để thu thập, phân tích và tài liệu hóa yêu cầu. Một số khoá đào tạo có thể giúp các nhà phát triển bổ sung các kỹ năng còn thiếu này, giúp họ làm tốt vai trò phân tích viên yêu cầu.

Do quy trình yêu cầu là một quy trình chủ chốt đối với sự thành công của dự án, nên tất cả những người liên quan đến dự án (stakeholders) cần phải có một hiểu biết cơ bản về sự hợp lý, tầm quan trọng, các practices của công nghệ yêu cầu. *Tổ Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

chức một khoá đào tạo ngắn trong 1 hoặc 2 ngày cho những người liên quan (nhà phát triển, người làm marketing, khách hàng, kiểm thử viên, các nhân viên quản lý) về quy trình yêu cầu khái quát có thể là một hoạt động xây dựng nhóm hiệu quả. Tất cả những người tham gia sẽ có một hiểu biết tốt hơn về những thách thức mà các đồng nghiệp của họ phải đối mặt, họ sẽ biết cách phối hợp tốt hơn với các đồng nghiệp vì sự thành công của dự án. Tương tự, các nhà phát triển sẽ có được các khái niệm và thuật ngữ của miền ứng dụng.

Đào tạo các nhà phân tích yêu cầu (Train requirements analysts)

Tất cả các nhà phát triển sẽ được đào tạo các kiến thức cơ bản trong công nghệ yêu cầu, nhưng những người trong số họ chịu trách nhiệm chính về nắm bắt, tài liệu hoá, phân tích các yêu cầu người dùng cần được đào tạo kỹ hơn về các hoạt động này. Các nhà phân tích yêu cầu có kỹ năng được tập hợp lại, họ cần có kỹ năng giao tiếp tốt, hiểu biết miền ứng dụng và có thể lựa chọn các công cụ xử lý yêu cầu (tool kit) trợ giúp cho công việc của mình.

Giáo dục đại diện người dùng và các nhà quản lý về yêu cầu phần mềm (Educate user representatives and managers about software requirements)

Đại diện người dùng sẽ tham gia vào các hoạt động triển phần mềm, họ sẽ được giáo dục 1 ngày về công nghệ yêu cầu cùng các nhà quản lý bên phát triển và bên khách hàng. Họ sẽ hiểu được tầm quan trọng của yêu cầu, các hoạt động và các sản phẩm cần chuyển giao cho nhóm yêu cầu, các rủi ro xảy ra nếu sao nhãng quy trình yêu cầu.

Đào tạo các nhà phát triển về các khái niệm của miền ứng dụng (Train developers in application domain concepts)

Để giúp các nhà phát triển có được hiểu biết về miền ứng dụng, hãy thu xếp cho họ các khoá học ngắn về các hoạt động nghiệp vụ của khách hàng, về các thuật ngữ, các mục tiêu của sản phẩm cần được sản xuất. Việc này sẽ làm giảm sự hỗn độn, giảm sự rối loạn truyền thông (miscommunication) giữa các bên liên quan và giảm các công việc phải làm lại sau này. Bạn cũng có thể giải thích thêm với các nhà phát triển về các biệt ngữ và các khái niệm nghiệp vụ quan trọng trong diễn biến của dự án sau này. Người trợ giúp sản phẩm (product champion) sẽ đóng vai trò này.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Tạo ra một bảng thuật ngữ của dự án (Create a project glossary)

Để giảm bớt các vấn đề truyền thông, hãy viết một bảng thuật ngữ định nghĩa tất cả các khái niệm chuyên môn trong miền ứng dụng của dự án. Trong đó gồm cả các thuật ngữ có nhiều cách sử dụng, cũng như các nghĩa cụ thể và thông dụng của nó.

II. SUY LUẬN YÊU CẦU (REQUIREMENTS ELICITATION)

(Mục này nói về các nguồn có thể phát sinh yêu cầu – ND thêm vào)

Chương 1 đã thảo luận về 3 mức yêu cầu: yêu cầu kinh doanh, yêu cầu người dùng, yêu cầu chức năng (business, user, functional). Chúng xuất hiện từ các nguồn khác nhau, trong những thời điểm khác nhau của dự án, nhằm đáp ứng các mục đích khác nhau vì vậy cần được tài liệu hóa theo những cách khác nhau. Yêu cầu kinh doanh (business requirements) (hoặc còn gọi là tài liệu về tầm nhìn và phạm vi của sản phẩm) phải bao hàm bất cứ yêu cầu người dùng nào (hoặc use – cases nào), và tất cả các yêu cầu chức năng phải được lẩn vét tới yêu cầu người dùng. Bạn cũng cần phải luận ra các yêu cầu phi chức năng, ví dụ các thuộc tính chất lượng, từ những nguồn thích hợp. Bạn có thể tìm thông tin bổ sung về các chủ đề này trong các chương sau:

- Chương 4 – Định nghĩa một thủ tục phát triển yêu cầu (Define a requirements development procedure)
- Chương 6 - Viết tài liệu tầm nhìn và phạm vi dự án (Write a project vision and scope document)
- Chương 7 – Xác định các lớp người dùng và các đặc tính của họ, lựa chọn người hỗ trợ sản phẩm cho mỗi lớp người dùng (Identify user classes and their characteristics; select product champions for each user class)
- Chương 8 - Đề nghị đại diện người dùng xác định các use cases (Have user representatives identify use cases)
- Chương 11 – Xác định các thuộc tính chất lượng và các yêu cầu phi chức năng khác (Determine quality attributes and other nonfunctional requirements)

Định nghĩa một thủ tục phát triển yêu cầu (Define a requirements development procedure)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Định nghĩa và tài liệu hoá các bước mà tổ chức của bạn dùng để thu thập, phân tích, đặc tả (specify) và kiểm tra yêu cầu. Hướng dẫn thực hiện theo các bước sẽ khiến nhà phân tích thực hiện công việc một cách nhất quán, khiến hoạt động thu thập yêu cầu và lập lịch biểu dễ dàng hơn.

Viết tài liệu tầm nhìn và phạm vi dự án (Write a project vision and scope document)

Tài liệu tầm nhìn và phạm vi dự án phải chứa các mục tiêu kinh doanh (business objectives) ở tầm cao của doanh nghiệp đối với sản phẩm. Tất cả các use cases và các yêu cầu chức năng cần phải song song với tài liệu này. Báo cáo tầm nhìn (vision statement) cấp cho tất cả những người tham gia dự án một hiểu biết chung về các mục tiêu của dự án. Định nghĩa phạm vi của dự án quyết định tính năng hoặc yêu cầu nào được đưa vào dự án.

Xác định các lớp người dùng và các đặc tính của họ (Identify user classes and their characteristics)

Để tránh bỏ qua nhu cầu của bất kỳ nhóm người dùng nào (user community), hãy xác định các nhóm khách hàng khác nhau đối với sản phẩm của bạn. Họ có thể khác nhau về tần suất sử dụng, các tính năng sử dụng, các mức ưu tiên. Hãy mô tả các khía cạnh công việc của họ hoặc các đặc tính cá nhân có thể ảnh hưởng đến thiết kế của sản phẩm.

Lựa chọn người hỗ trợ sản phẩm cho mỗi lớp người dùng (Select product champions for each user class)

Xác định ít nhất một người có thể giới thiệu chính xác các nhu cầu của mỗi lớp người dùng, người được hiểu như là người phát ngôn của lớp người dùng đó, người thay mặt lớp người dùng đó ra các quyết định liên quan đến yêu cầu. Việc này rất dễ khi phát triển các hệ thống thông tin nội bộ (internal information systems), nơi mà người dùng là những người bạn có thể quan hệ thân thiết. Nếu bạn phát triển các hệ thống thông tin thương mại, hãy xây dựng quan hệ tốt đẹp với các khách hàng chính hoặc các bên kiểm thử beta (beta test sites) để xác định những người hỗ trợ sản phẩm thích hợp. Người hỗ trợ sản phẩm phải liên tục tham gia dự án và ra quyết định ngay khi cần thiết.

Thành lập các nhóm tập trung của người dùng tiêu biểu (Establish focus groups of typical users)

Gặp gỡ các nhóm nhỏ đại diện cho người dùng đã sử dụng các phiên bản trước đây của sản phẩm hoặc đã sử dụng các sản phẩm tương tự, tập hợp các yêu cầu chức năng và phi chức năng của họ về sản phẩm hiện thời. Hãy tập trung làm việc với các nhóm có giá trị đối với sự phát triển thương mại của sản phẩm. Khác với người hỗ trợ sản phẩm, các thành viên của nhóm tập trung thường không phải là chủ thể ra quyết định.

Đề nghị đại diện người dùng xác định các use cases (Have user representatives identify use cases)

Tập hợp các mô tả của đại diện người dùng về các tác vụ (tasks) họ cần hoàn thành bằng phần mềm – các use cases. Hãy thảo luận về các tương tác và đối thoại giữa người dùng và hệ thống nhằm giúp người dùng hoàn thành mỗi tác vụ (task) của họ. Thông qua một mẫu tiêu chuẩn (standard template) để tài liệu hoá các use cases và trích xuất các yêu cầu chức năng từ use cases.

Tổ chức các phiên Phát triển ứng dụng chung (Hold Joint Application Development sessions)

Một phiên JAD là một hội thảo mở rộng, được tổ chức để trao đổi về sự cộng tác giữa nhà phân tích và các đại diện khách hàng để sản sinh ra các bản thảo tài liệu yêu cầu.

Phân tích workflow của người dùng (Analyze user workflow)

Quan sát người dùng thực hiện các tác vụ (tasks) của họ. Tạo ra các sơ đồ đơn giản (DFD...) phác thảo *khi nào (when)* thì người dùng có *dữ liệu gì (what data)* và họ xử lý dữ liệu *như thế nào (how)*. Tài liệu hoá luồng quy trình nghiệp vụ (business process flow) sẽ giúp bạn xác định các use cases và các yêu cầu chức năng cho sản phẩm. Thậm chí bạn có thể xác định khách hàng có thật sự cần một hệ thống phần mềm mới để đáp ứng các mục tiêu nghiệp vụ của họ (business objectives) hay không (McGraw and Harbison 1997).

Xác định các thuộc tính chất lượng và các yêu cầu phi chức năng khác (Determine quality attributes and other nonfunctional requirements)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Xác định các thuộc tính chất lượng (yêu cầu phi chức năng) sẽ giúp sản phẩm của bạn đáp ứng, thậm chí đáp ứng quá mức các kỳ vọng của khách hàng. Các đặc tính đó gồm: hiệu suất, độ tin cậy, khả năng dễ sử dụng và nhiều thứ khác nữa (performance, reliability, usability, ...). Tầm quan trọng tương đối của các thuộc tính chất lượng được xác định bởi người dùng.

Kiểm tra các báo cáo vấn đề của hệ thống hiện tại để tìm kiếm các ý tưởng yêu cầu (Examine problem reports of current systems for requirement ideas)

Các báo cáo vấn đề và đề xuất mở rộng từ khách hàng là nguồn cung cấp phong phú các ý tưởng về tính năng và cải tiến (features and improvements) sẽ được đưa vào một phiên bản nào đó của hệ thống mới. Người hỗ trợ hệ thống hiện tại cũng có thể cung cấp các gợi ý có giá trị đối với quy trình thu thập yêu cầu.

Sử dụng lại yêu cầu trong dự án (Reuse requirements across projects)

Nếu khách hàng đề xuất tính năng tương tự với tính năng đã có trong hệ thống cũ thì hãy cân nhắc liệu có thể sử dụng lại components đã có liên quan đến tính năng đó hay không.

III. PHÂN TÍCH YÊU CẦU (REQUIREMENTS ANALYSIS)

Phân tích yêu cầu bao gồm việc làm mịn, phân tích, nghiên cứu kỹ lưỡng các yêu cầu đã thu thập được để đảm bảo chắc chắn tất cả stakeholders (những người có liên quan) hiểu điều họ muốn, để tìm kiếm các lỗi, các thiếu sót và thiếu hụt khác. Công việc phân tích yêu cầu cũng đánh giá liệu các yêu cầu và SRS có đáp ứng đầy đủ các đặc tính về yêu cầu tuyệt vời được viết ở Chương 1 hay không. Mục đích của bạn là phát triển các yêu cầu chất lượng và chi tiết đủ để có thể xây dựng các ước lượng dự án thực tế và thực hiện thiết kế, xây dựng, kiểm thử.

Thường sẽ rất hữu ích nếu biểu diễn các yêu cầu dưới nhiều hình thức khác nhau, ví dụ mô tả bằng lời (textual form) và bằng hình ảnh (graphical form). Phân tích yêu cầu từ các góc nhìn khác nhau sẽ thấy lộ ra các vấn đề bên trong mà ta sẽ không thể thấy được nếu chỉ nhìn hệ thống từ một góc độ (Davis 1995). Phân tích cũng là việc tương tác với khách hàng để làm sáng tỏ các điểm còn chưa rõ và biết yêu cầu nào là quan trọng hơn yêu cầu nào. Mục đích cũng là đảm bảo các stakeholders sớm thống nhất được một cách hiểu chung - một tầm nhìn chung

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

(shared vision) - về sản phẩm sẽ được sản xuất. Các chương sau sẽ thảo luận về phân tích yêu cầu:

- Chương 6 - Vẽ một sơ đồ bối cảnh của hệ thống (Draw a context diagram of system)
- Chương 9 - Tạo một từ điển dữ liệu (Create a data dictionary)
- Chương 10 – Mô hình hóa yêu cầu (Model the requirements)
- Chương 12 - Tạo các nguyên mẫu giao diện người dùng (Create user interface prototypes)
- Chương 13 – Xếp thứ tự ưu tiên các yêu cầu (Prioritize the requirements)

Vẽ một sơ đồ bối cảnh của hệ thống (Draw a context diagram of system)

Sơ đồ bối cảnh là một mô hình đơn giản định nghĩa các đường biên và các giao diện giữa hệ thống đang được xây dựng với các thực thể bên ngoài tức môi trường của hệ thống. Nó cũng định nghĩa luồng thông tin và các đầu vào (materials) thông qua các giao diện.

Tạo các nguyên mẫu giao diện người dùng (Create user interface prototypes)

Khi các nhà phát triển và người dùng không chắc chắn được về yêu cầu thì có thể xây dựng một nguyên mẫu giao diện người dùng để làm cho các ý tưởng và các khả năng lựa chọn (concepts and possibilities) dễ hình dung hơn. Người dùng có thể đánh giá nguyên mẫu để giúp những người tham gia dự án có được một sự hiểu biết tốt hơn về bài toán cần giải quyết. Hãy cố tìm kiếm sự không nhất quán giữa các yêu cầu đã được viết ra và các nguyên mẫu.

Phân tích tính khả thi của yêu cầu (Analyze requirement feasibility)

Đánh giá về tính khả thi của việc cài đặt mỗi yêu cầu ở mức hiệu quả và chi phí có thể chấp nhận trong môi trường chuyển giao dự kiến (*môi trường mà hệ thống dự định sẽ hoạt động* – ND). Tìm hiểu các rủi ro liên quan tới việc cài đặt mỗi yêu cầu như xung đột giữa các yêu cầu, sự phụ thuộc của yêu cầu vào các yếu tố bên ngoài, các trở ngại kỹ thuật.

Xếp thứ tự ưu tiên các yêu cầu (Prioritize the requirements)

Căn cứ trên nhu cầu tất cả các stakeholders để xác định mức ưu tiên tương đối của các use cases, các tính năng sản phẩm, các yêu cầu cá nhân. Dựa trên mức ưu tiên *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

đã xác lập, xác định phiên bản nào của sản phẩm sẽ chứa những tính năng nào hoặc chứa tập hợp các yêu cầu nào. Nếu các thay đổi của yêu cầu được chấp thuận, hãy phân bổ các thay đổi này vào một phiên bản tương lai cụ thể, xác định các thay đổi tương ứng trong việc lập kế hoạch phát hành các phiên bản.

Mô hình hoá các yêu cầu (Model the requirements)

Các mô hình phân tích bằng hình ảnh của yêu cầu có thể là hỗ trợ có giá trị cho SRS. Chúng thể hiện các thông tin và mối quan hệ khác nhau, chúng giúp tìm kiếm các yêu cầu không đúng đắn, không nhất quán, các yêu cầu còn thiếu. Các mô hình như vậy gồm DFDs, ERDs, state-transition diagrams, dialog maps, object class và interaction diagrams.

Tạo một từ điển dữ liệu (Create a data dictionary)

Từ điển dữ liệu là một kho chứa trung tâm các định nghĩa của tất cả các mục dữ liệu (data items) và các cấu trúc được hệ thống sử dụng. Nó đảm bảo tất cả các nhà phát triển liên quan đến dự án đều hiểu nhất quán các định nghĩa dữ liệu. Ở mức yêu cầu, từ điển dữ liệu ít nhất phải định nghĩa các mục dữ liệu khách hàng (customer data items) sao cho khách hàng và nhóm phát triển sử dụng chung các định nghĩa và thuật ngữ. Các công cụ phân tích và thiết kế thường bao gồm một data dictionary component.

IV. ĐẶC TẢ YÊU CẦU (REQUIREMENTS SPECIFICATION)

Dù yêu cầu có thể đến từ bất cứ nguồn nào và bạn có thể thu thập chúng bằng bất cứ cách nào, thì bạn cũng phải tài liệu hóa chúng theo một cách nhất quán, dễ truy nhập và có thể soát xét (consistent, accessible, reviewable). Yêu cầu kinh doanh (business requirements) có thể được ghi nhận ở tài liệu tầm nhìn và phạm vi (vision and scope) của dự án. Yêu cầu người dùng (user requirements) được tài liệu hóa trong một use case template chuẩn. SRS chứa các yêu cầu chức năng (functional requirements) và phi chức năng. Bạn cũng phải thiết lập một quy ước chuẩn để định danh duy nhất mỗi yêu cầu. Bất cứ quy ước nào được sử dụng trong SRS cũng phải được định nghĩa để đảm bảo SRS được viết theo một phong cách nhất quán và độc giả biết được cách diễn giải nội dung trong đó như thế nào. Cụ thể về đặc tả yêu cầu được thảo luận trong các chương sau:

- Chương 8 – Ghi nhận các quy tắc nghiệp vụ (Record business rules)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- Chương 9 – Thông qua một SRS template; Gán nhãn mỗi yêu cầu (Adopt a SRS template; Label each requirement)
- Chương 18 – Xác định nguồn của yêu cầu; Tạo một ma trận có thể lẩn vết yêu cầu (Identify the sources of requirements; Create a requirements traceability matrix)

Thông qua một SRS template; Gán nhãn mỗi yêu cầu (Adopt a SRS template; Label each requirement)

Hãy định nghĩa một template chuẩn để tài liệu hoá yêu cầu. Template tạo ra một cấu trúc nhất quán để ghi lại cả yêu cầu chức năng (functional requirements) và nhiều thông tin quan trọng khác liên quan đến yêu cầu. Thay vì tạo ra một template mới, hãy thích ứng (adapt) một mẫu đã có cho dự án của bạn. Nhiều tổ chức đã sử dụng SRS template được mô tả trong IEEE Standard 830-1998 (IEEE 1998).

Xác định nguồn của yêu cầu (Identify the sources of requirements)

Để đảm bảo tất cả các stakeholders đều biết tại sao mỗi yêu cầu chức năng (functional requirement) có mặt trong SRS, hãy ghi lại nguồn phát sinh yêu cầu đó. Nguồn đó có thể là một tình huống sử dụng (use case), hoặc một đầu vào của khách hàng, một yêu cầu hệ thống cấp cao hơn, một quy tắc kinh doanh (business rule), một quy định của chính phủ (government regulation), một tiêu chuẩn, hoặc một nguồn bên ngoài khác.

Gán nhãn mỗi yêu cầu (Label each requirement)

Đặt ra một quy ước (convention) để định danh riêng rẽ mỗi yêu cầu trong SRS bằng một nhãn (label or tag). Quy ước này phải đủ bao quát được tất cả các tình huống liên quan đến yêu cầu trong toàn bộ dự án như chỉnh sửa, xoá bỏ, thêm mới. Gán nhãn (labeling) yêu cầu là cho phép lẩn vết yêu cầu (requirements traceability). Mỗi khi thay đổi yêu cầu, cần ghi nhận lại nội dung trước và cả sau khi thay đổi, cần thiết lập các độ đo (metrics) xác định tình trạng yêu cầu (requirements status).

Ghi nhận các quy tắc kinh doanh (Record business rules)

Các quy tắc kinh doanh (business rule) là các nguyên tắc hoạt động của sản phẩm, như ai có thể thực hiện hành động gì và dưới hoàn cảnh (circumstance) nào. Tài *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

liệu hoá các quy tắc này trong một phần đặc biệt (special section) của SRS hoặc trong một tài liệu quy tắc kinh doanh (business rule) riêng biệt. Một số quy tắc kinh doanh (business rule) dẫn tới các yêu cầu chức năng (functional requirements) cần thiết (để làm cho nó trở nên hiệu lực), các yêu cầu chức năng (functional requirements) này cần được lặp vết quay ngược trở lại quy tắc kinh doanh (business rule) đã sinh ra nó.

Tạo một ma trận có thể lặp vết yêu cầu (Create a requirements traceability matrix)

Tạo một ma trận kết nối tất cả các yêu cầu riêng biệt tới các phần tử thiết kế, mã nguồn và kiểm thử (design, code, test elements) yêu cầu này. Ma trận lặp vết yêu cầu cũng kết nối các yêu cầu chức năng (functional requirements) tới các yêu cầu cấp cao hơn mà từ đó nó được sinh ra. Hãy làm ma trận này song song với quá trình phát triển dự án.

V. KIỂM TRA YÊU CẦU (REQUIREMENTS VERIFICATION)

Các hoạt động kiểm tra đảm bảo các lời thề hiện yêu cầu là chính xác, đầy đủ, và mô tả được các đặc tính chất lượng mong muốn. Các yêu cầu dường như tốt đẹp nếu chỉ đọc chúng từ các SRS, nhưng khi bạn thực sự làm việc với chúng thì có thể nảy sinh các vấn đề. Nếu bạn viết các test cases từ các yêu cầu, bạn có thể phát hiện các nhập nhằng và sự không chắc chắn trong một số yêu cầu. Những cái không rõ ràng này phải được loại bỏ nếu yêu cầu được coi như là một nền tảng tin cậy (reliable foundation) cho thiết kế và kiểm tra hệ thống cuối cùng. Sự tham gia của khách hàng là yếu tố cơ bản cần thiết cho hoạt động kiểm tra yêu cầu, chúng sẽ được mô tả trong Chương 14.

Thanh tra tài liệu yêu cầu (Inspect requirement documents)

Thanh tra chính thức (formal inspection) tài liệu yêu cầu là một trong những practices có khả năng mang lại giá trị cao nhất cho chất lượng phần mềm. Tập hợp một nhóm nhỏ các thanh tra viên (inspectors) đại diện cho nhiều quan điểm khác nhau (gồm phân tích viên, khách hàng, thiết kế viên, kiểm thử viên) kiểm tra cẩn thận SRS và các mô hình liên quan nhằm tìm kiếm các khiếm khuyết (defects). Các soát xét sơ bộ không chính thức (informal preliminary reviews) trong giai đoạn phát triển yêu cầu cũng rất có giá trị.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Viết các test cases từ các yêu cầu (Write test cases from requirements)

Chúng ta thu được các black-box test cases (functional test) từ các use cases tài liệu hoá hành vi được mong muốn của sản phẩm trong các điều kiện xác định (specified conditions). Duyệt qua (walk through) các test cases cùng khách hàng để đảm bảo chúng phản ánh đúng hành vi được mong đợi. Từ các test cases lần lượt ngược lại các yêu cầu chức năng (functional requirements) để đảm bảo chắc chắn không yêu cầu nào bị bỏ qua (overlooked) và tất cả đều có các test cases tương ứng. Sử dụng các test cases để kiểm tra tính đúng đắn của mô hình yêu cầu như dialog maps, prototypes (nguyên mẫu).

Viết một sổ tay người dùng (Write a user manual)

Phác thảo sổ tay người dùng ngay từ sớm trong quy trình phát triển yêu cầu và dùng nó như là tài liệu đặc tả yêu cầu hoặc như một trợ giúp cho phân tích yêu cầu. Một tài liệu sổ tay người dùng tốt sẽ mô tả tất cả các chức năng mà người dùng thấy được (user – visible functionality) bằng một ngôn ngữ dễ hiểu. Các yêu cầu khác như các thuộc tính chất lượng, yêu cầu hiệu suất, chức năng không thấy được đối với người dùng (not visible to users) sẽ được tài liệu hoá trong SRS.

Định nghĩa tiêu chuẩn chấp nhận (Define acceptance criteria)

Đề nghị người dùng mô tả họ xác định một sản phẩm như thế nào thì đáp ứng nhu cầu sử dụng của họ và mô tả lại các tiêu chuẩn ấy (Hsia, Kung and Sell 1997).

VI. QUẢN LÝ YÊU CẦU (REQUIREMENTS MANAGEMENT)

Khi bạn đã có yêu cầu thì bạn cũng phải đối mặt với các thay đổi không thể tránh được của yêu cầu như là thể hiện của sự tiến hoá của dự án. Quản lý thay đổi hiệu quả đòi hỏi một quy trình để xuất các thay đổi và đánh giá các chi phí và ảnh hưởng tiềm tàng của thay đổi trên toàn bộ dự án. Một ban kiểm soát thay đổi (change control board) phối hợp với các stakeholders quan trọng phải ra các quyết định chấp nhận hoặc từ chối các thay đổi này.

Các practices quản lý cấu hình là điều kiện tiên quyết để quản lý yêu cầu một cách hiệu quả. Nhiều tổ chức phát triển sử dụng cách kiểm soát phiên bản (version control) và các kỹ thuật quản lý cấu hình khác để kiểm soát code base, nhưng bạn Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

cũng có thể sử dụng các practices đó để quản lý tài liệu yêu cầu. Cải tiến quy trình quản lý yêu cầu có thể là một cách để đưa các practices quản lý cấu hình mới (new configuration management practices) vào tổ chức của bạn. Các kỹ thuật liên quan trong quản lý yêu cầu được mở rộng trong các chương sau:

- Chương 16 – Thiết lập một ranh giới và kiểm soát các phiên bản của tài liệu yêu cầu (Establish a baseline and control versions of requirements documents).
- Chương 17 – Định nghĩa một quy trình kiểm soát thay đổi yêu cầu; Thiết lập một ban kiểm soát thay đổi (Define a requirements change control process; Establish a change control board).
- Chương 18 - Thực hiện phân tích ảnh hưởng thay đổi yêu cầu; Lần vết một thay đổi yêu cầu tới tất cả các sản phẩm liên quan (Perform a requirements change impact analysis; Trace a requirements change to all affected work products)
- Chương 19 – Sử dụng một công cụ quản lý yêu cầu (Use a requirements management tool).

Định nghĩa một quy trình kiểm soát yêu cầu (Define a requirements change control process)

Thiết lập một quy trình đề xuất, phân tích và ra quyết định về các thay đổi yêu cầu. Tất cả các thay đổi được đề xuất phải tuân theo quy trình này.

Thành lập một ban kiểm soát thay đổi (Establish a change control board)

Một nhóm nhỏ các stakeholders được tập hợp lại như một ban kiểm soát thay đổi để tiếp nhận các đề xuất thay đổi yêu cầu, xác định liệu các thay đổi đó còn trong phạm vi dự án, đánh giá và ra quyết định chấp nhận hoặc từ chối, nếu chấp nhận thì xác định thứ tự ưu tiên thi công đề xuất thay đổi này.

Thực hiện phân tích ảnh hưởng của thay đổi yêu cầu (Perform requirements change impact analysis)

Mỗi thay đổi được chấp nhận đều được đánh giá để xác định mức độ ảnh hưởng đến lịch biểu và các yêu cầu khác. Xác định các thay đổi tương ứng với thiết kế và thi công các tác vụ liên quan, xác định nhân lực cần thiết để hoàn thành các thay đổi.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Lần vết một thay đổi yêu cầu tới tất cả các bản thành phẩm liên quan (Trace a requirements change to all affected work products)

Khi một đề xuất thay đổi trong một yêu cầu được chấp nhận, hãy tham chiếu ma trận lần vết yêu cầu để xác định các yêu cầu bị ảnh hưởng, các thiết kế của components, mã nguồn, các test cases.

Thiết lập một ranh giới và kiểm soát các phiên bản của tài liệu yêu cầu (Establish a baseline and control versions of requirements documents)

Định nghĩa một ranh giới yêu cầu, tại đó tất cả các thoả thuận về nội dung các yêu cầu đã được chấp thuận cho đến thời điểm đó. Sau khi ranh giới được thiết lập, các thay đổi chỉ được chấp nhận bởi ban kiểm soát thay đổi thông qua quy trình kiểm soát thay đổi đã định nghĩa. Mỗi phiên bản của tài liệu đặc tả yêu cầu được định danh duy nhất. Có thể quản lý các phiên bản này bằng các công cụ quản lý cấu hình thích hợp.

Duy trì một lịch sử thay đổi yêu cầu (Maintain a history of requirements changes)

Hãy ghi lại ngày tháng mà các thay đổi xảy ra và ngày tháng phát sinh các phiên bản, lý do mỗi thay đổi đó, các thay đổi đó được thực hiện như thế nào, ai cập nhật tài liệu, số hiệu mỗi phiên bản. Một công cụ kiểm soát phiên bản có thể tự động làm việc này.

Giám sát tình trạng mỗi yêu cầu (Track status of each requirement)

Thiết lập một CSDL mà mỗi bản ghi là một yêu cầu chức năng (functional requirements) riêng biệt. Lưu giữ các thuộc tính quan trọng của mỗi yêu cầu như **tình trạng** của yêu cầu (được đề xuất, được chấp thuận, được thực thi, được kiểm tra), sao cho số lượng mỗi loại yêu cầu theo từng trạng thái có thể được biết bất cứ lúc nào.

Đo lường độ ổn định của yêu cầu (Measure requirement stability)

Ghi lại số lượng các yêu cầu được đã được định ranh giới (baselined requirements) và số lượng các thay đổi được đề xuất và chấp thuận (proposed and approved changes) (hiệu chỉnh, sửa chữa, xoá bỏ) đối với các yêu cầu đó trong mỗi tuần. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

hoặc mỗi tháng. Các thay đổi yêu cầu quá nhiều là một dấu hiệu chỉ ra rằng bài toán đã chưa được hiểu đúng, phạm vi dự án không được định nghĩa tốt, hoặc tình hình tổ chức không ổn định.

Sử dụng một công cụ quản lý yêu cầu (Use a requirements management tool)

Các công cụ quản lý yêu cầu thường mại cho phép bạn lưu trữ các yêu cầu riêng biệt theo từng loại khác nhau trong một CSDL, định nghĩa các thuộc tính cho mỗi yêu cầu, giám sát tình trạng mỗi yêu cầu, định nghĩa khả năng lặp vết giữa yêu cầu và tất cả các bùn thành phẩm liên quan.

VII. QUẢN LÝ DỰ ÁN (PROJECT MANAGEMENT)

Các cách tiếp cận quản lý dự án liên quan mật thiết tới các quy trình yêu cầu (requirements processes) của dự án. Các kế hoạch của dự án cần phải được thiết lập dựa trên chức năng cần xây dựng của sản phẩm phần mềm, các thay đổi yêu cầu sẽ ảnh hưởng đến các kế hoạch đó. Kế hoạch cần tiên liệu và điều chỉnh các thay đổi được chấp nhận trong phạm vi của dự án. Nếu các yêu cầu ban đầu không chắc chắn, bạn có thể chọn một cách phát triển phần mềm chấp nhận sự không chắc chắn đó và cho phép chỉ cài đặt các yêu cầu theo từng phần cùng với sự hiểu biết về yêu cầu tăng dần. Cách tiếp cận quản lý dự án căn cứ trên yêu cầu được thảo luận trong các chương sau:

- Chương 5 – Tài liệu hóa và quản lý các rủi ro liên quan đến yêu cầu (Document and manage requirements – related risks).
- Chương 15 – Thiết lập các kế hoạch dự án dựa trên yêu cầu (Base project plans on requirements).
- Chương 16 – Giám sát nguồn nhân lực dành cho phát triển và quản lý yêu cầu (Track the effort you spend on requirements development and management)

Chọn lựa một vòng đời phát triển phần mềm thích hợp (Select an appropriate software development life cycle)

Cách phát triển phần mềm thác nước cổ điển có thể chỉ thành công nếu yêu cầu được định nghĩa đầy đủ ngay từ sớm. Doanh nghiệp của bạn cần phải định nghĩa một số cách thức phát triển thích hợp với các loại hình dự án khác nhau tùy các mức định nghĩa yêu cầu khác nhau (McConnell 1996). Nếu yêu cầu và/hoặc phạm vi yêu cầu được định nghĩa chưa rõ ngay từ đầu thì hãy lập kế hoạch phát triển theo Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

từng bước tăng dần, bắt đầu với các yêu cầu đã được hiểu rõ bằng một kiến trúc vững chắc và có thể chỉnh sửa (robust and modifiable architecture).

Thiết lập các kế hoạch dự án dựa trên yêu cầu (Base project plans on requirements)

Thiết lập các kế hoạch và lịch biểu dự án bằng cách lặp theo sự mở rộng của phạm vi và các yêu cầu được làm chi tiết hơn. Bắt đầu bằng ước lượng nhân lực cần thiết để phát triển các yêu cầu chức năng (functional requirements) từ tài liệu tầm nhìn và phạm vi (vision and scope). Các ước lượng chi phí và lịch biểu ngay từ sớm dựa trên các yêu cầu được định nghĩa chưa rõ ràng sẽ có độ bất ổn cao, các ước lượng được cải tiến theo sự tốt dần hơn của yêu cầu.

Đàm phán lại các cam kết của dự án khi các yêu cầu thay đổi (Renegotiate project commitments when requirements change)

Khi các yêu cầu mới được đưa vào dự án, hãy đánh giá liệu bạn có thể hoàn thành công việc như các cam kết về chất lượng và lịch biểu với mức tiêu thụ tài nguyên hiện có hay không. Nếu không, hãy thông báo cho cấp quản lý về hiện thực của dự án, hãy đàm phán lại về tính hiện thực của các cam kết (Humphrey 1997). Nếu các cam kết của bạn không thành, hãy thông báo cho các bên biết kết quả cập nhật rủi ro mới của dự án.

Tài liệu hóa và quản lý các rủi ro liên quan đến yêu cầu (Document and manage requirements – related risks)

Tập kích não (brainstorm) để tìm kiếm các rủi ro liên quan đến yêu cầu và ghi chúng vào kế hoạch quản lý rủi ro của dự án. Hãy suy nghĩ cách tiếp cận để giảm hoặc tránh rủi ro, thực hiện các hành động giảm rủi ro, giám sát diễn biến và hiệu quả của các công việc xử lý rủi ro.

Giám sát nguồn nhân lực dành cho phát triển và quản lý yêu cầu (Track the effort you spend on requirements development and management)

Ghi chép lại tình trạng nhân lực được dành cho các hoạt động phát triển và quản lý yêu cầu. Sử dụng dữ liệu này để đánh giá liệu các hoạt động yêu cầu đã được lập kế hoạch tốt hay chưa để rút kinh nghiệm nhằm điều chỉnh dự án hiện tại tốt hơn và rút kinh nghiệm cho các dự án tương lai.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Các bước tiếp theo

- Hãy quay lại xem các vấn đề liên quan đến yêu cầu mà bạn đã xác định trong **Các bước tiếp theo** của Chương 1. Hãy xác định các good practices trong Chương này có thể giúp bạn giải quyết mỗi vấn đề đó như thế nào. Với mỗi practice bạn chọn, hãy xác định bất cứ rào cản nào về mặt tổ chức và văn hoá có thể gây khó khăn cho sự ứng dụng practice này.
- Hãy tạo một danh sách tất cả các requirement good practices mà bạn đã định danh trong bước trước. Với mỗi practice, hãy xác định khả năng của nhân lực trong dự án của bạn: chuyên gia, người thành thạo, người chưa có kinh nghiệm, người chưa biết tí gì. Nếu nhóm của bạn không thành thạo ít nhất một practice, hãy yêu cầu ai đó học các practices này và chia sẻ với các thành viên còn lại.

SATA-APTECH

CHƯƠNG 4

CẢI TIẾN QUY TRÌNH YÊU CẦU CỦA BẠN

Chương 3 đã mô tả nhiều hướng dẫn thực hành tốt (good practices) của công nghệ yêu cầu, bạn có thể ứng dụng các practices đó vào việc cải tiến quy trình yêu cầu của mình. Tuy nhiên, nếu các nỗ lực cải tiến quy trình được bắt đầu sai thì những người bị ảnh hưởng từ quy trình này sẽ kháng cự lại và có thể chương trình cải tiến sẽ thất bại.

Cải tiến quy trình phần mềm có 2 mục tiêu chính:

- Sửa chữa các vấn đề mà bạn đã gặp trong các dự án trước và hiện tại.
- Tiên liệu và ngăn ngừa các vấn đề mà bạn có thể sẽ gặp trong các dự án tương lai.

Nếu cách thức làm việc hiện nay của bạn dường như là tốt thì bạn không thấy có nhu cầu thay đổi cách làm yêu cầu của mình. Tuy nhiên, thậm chí các công ty phần mềm thành công cũng sẽ phải đối mặt với các khó khăn to lớn khi thực hiện các dự án lớn hơn, khi làm việc với một cộng đồng khách hàng khác, khi lịch biểu được siết chặt hơn, hoặc khi làm việc trong một miền ứng dụng mới. Vì vậy, bạn cũng nên biết những cách tiếp cận làm yêu cầu mới có giá trị đối với công việc của bạn.

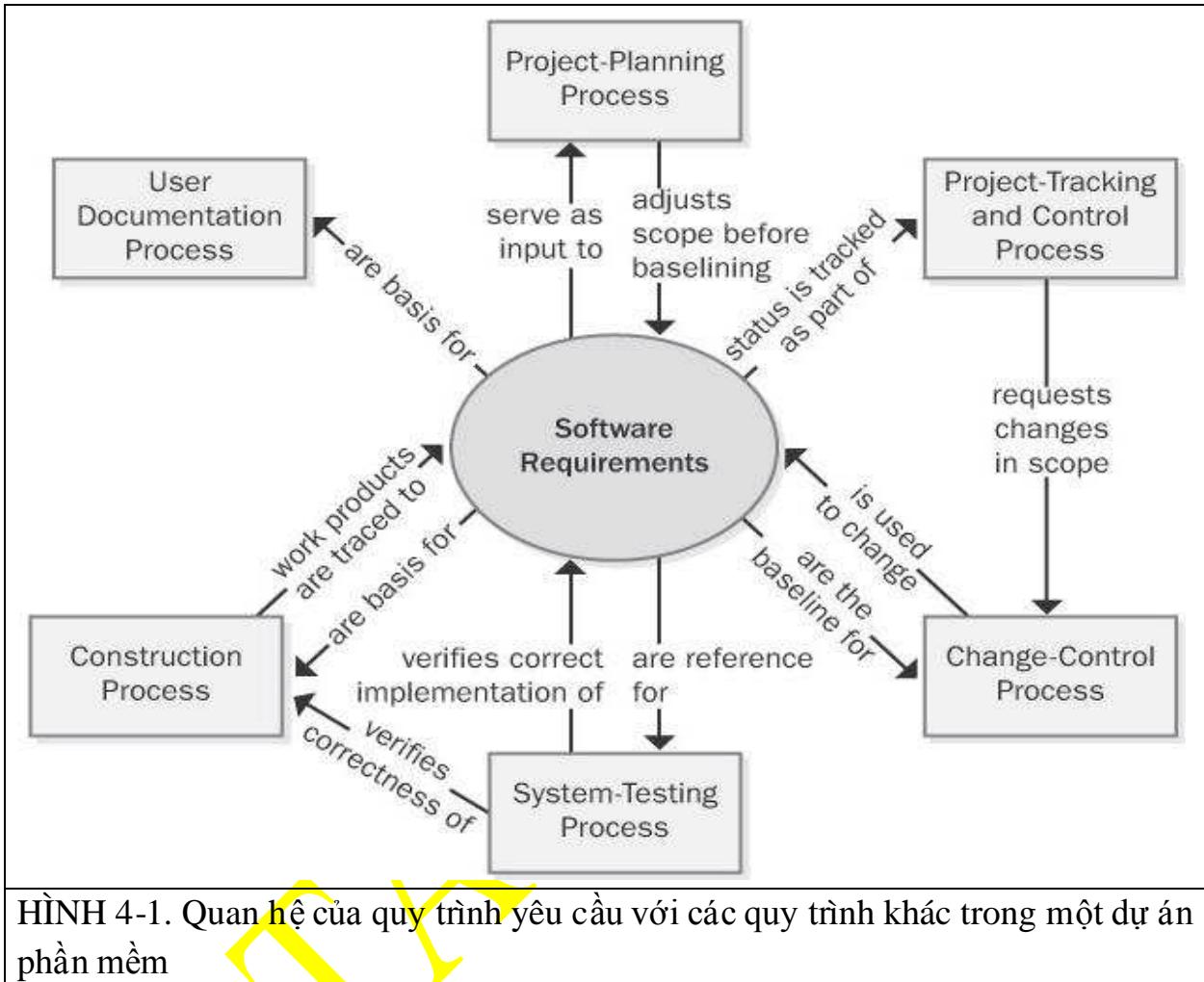
Chương này mô tả yêu cầu liên quan ra sao với các quy trình chính của dự án và những người có liên quan khác (stakeholders). Một số khái niệm cơ bản về cải tiến quy trình phần mềm và một cách cải tiến quy trình cũng sẽ được đề xuất với các bạn. Tôi sẽ liệt kê một số “tài sản quy trình” yêu cầu (requirement “process assets”) quan trọng mà tổ chức của bạn nên sử dụng. Chương này kết thúc bằng mô tả một lộ trình về cải tiến quy trình yêu cầu có sử dụng các practices trên.

I. YÊU CẦU LIÊN QUAN NHƯ THẾ NÀO ĐẾN CÁC QUY TRÌNH KHÁC CỦA DỰ ÁN (HOW REQUIREMENTS RELATE TO OTHER PROJECT PROCESSES)

Yêu cầu nằm ở trái tim của các dự án phần mềm thành công, nó trợ giúp nhiều hoạt động quản lý và kỹ thuật. Các thay đổi mà bạn thực hiện trong cách tiếp cận các

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

hoạt động phát triển và quản lý yêu cầu sẽ ảnh hưởng tới các quy trình khác và ngược lại. Hình 4-1 minh họa một số liên quan giữa quy trình yêu cầu và các quy trình khác của một dự án phần mềm;



Sự liên quan giữa quy trình yêu cầu và các quy trình khác được mô tả ngắn gọn dưới đây.

Quy trình Lập kế hoạch dự án (Project planning process)

Yêu cầu phải là cơ sở của các quy trình lập kế hoạch dự án. Các ước lượng tài nguyên và lịch biểu cần dựa trên sự hiểu biết về cái gì sẽ được xây dựng và chuyển giao cho khách hàng. Thông thường, lập kế hoạch dự án nghĩa là tính toán sao cho tất cả các tính năng mong muốn sẽ được thực hiện trong một giới hạn ngân sách và thời gian nhất định. Các quy trình lập kế hoạch có thể dẫn tới việc thu hẹp phạm vi

dự án hoặc lựa chọn một cách tiếp cận từng bước một - phát hành dần từng phiên bản của sản phẩm, mỗi phiên bản chỉ bao gồm một số tính năng.

Quy trình Giám sát và kiểm soát dự án (Project tracking and control Process)

Giám sát (monitor) trạng thái của mỗi yêu cầu được coi là một phần của việc giám sát dự án (project tracking) sao cho các nhà quản lý dự án có thể biết liệu công việc có được tiến hành như mong muốn hay không. Nếu không, cấp quản lý có thể đề nghị thu hẹp phạm vi thông qua các quy trình kiểm soát thay đổi.

Quy trình Kiểm soát thay đổi (Change control Process)

Sau khi yêu cầu đã được tài liệu hóa và vạch ranh giới (baselined), tất cả các thay đổi tiếp theo của yêu cầu cần được thực hiện thông qua quy trình kiểm soát thay đổi đã định nghĩa. Quy trình kiểm soát thay đổi đảm bảo rằng:

- Ảnh hưởng của một đề xuất thay đổi (proposed change) đã được hiểu đầy đủ.
- Tất cả những ai bị ảnh hưởng bởi thay đổi thì đều đã nhận thức được điều đó.
- Những người có thẩm quyền ra quyết định để chấp nhận thay đổi.
- Tài nguyên được điều chỉnh tương ứng.
- Các tài liệu yêu cầu được cất giữ.

Quy trình Kiểm thử hệ thống (System testing process)

Các yêu cầu người dùng (user requirements) và các yêu cầu chức năng (functional requirements) là đầu vào chính để kiểm thử hệ thống. Nếu hành vi được mong đợi của phần mềm trong các điều kiện khác nhau không được đặc tả thì người kiểm thử rất khó biết hành vi nào của hệ thống là đúng, hành vi nào là sai. Ngược lại, kiểm thử hệ thống là một phương tiện để xác nhận rằng tất cả các chức năng đã được lập kế hoạch thì đều được thực hiện và các công việc (tasks) mà người dùng mong muốn đã hoạt động một cách đúng đắn.

Quy trình Làm tài liệu người dùng (User documentation process)

Trước đây tôi đã làm việc trong một công ty với vai trò technical writer – người viết tài liệu hướng dẫn cho các sản phẩm thương mại. Tôi hỏi một trong số các writers rằng tại sao chúng ta phải làm việc nhiều như vậy. “Chúng ta ở vào điểm Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com”

cuối của dây chuyền”, cô đáp. “Chúng ta là những người phải tài liệu hóa các thay đổi cuối cùng trong giao diện người dùng hiển thị các tính năng bị xoá bỏ hay thêm vào trong phút cuối”. Các yêu cầu là đầu vào chính của quy trình làm tài liệu, vì vậy chất lượng của yêu cầu sẽ quyết định chất lượng của tài liệu.

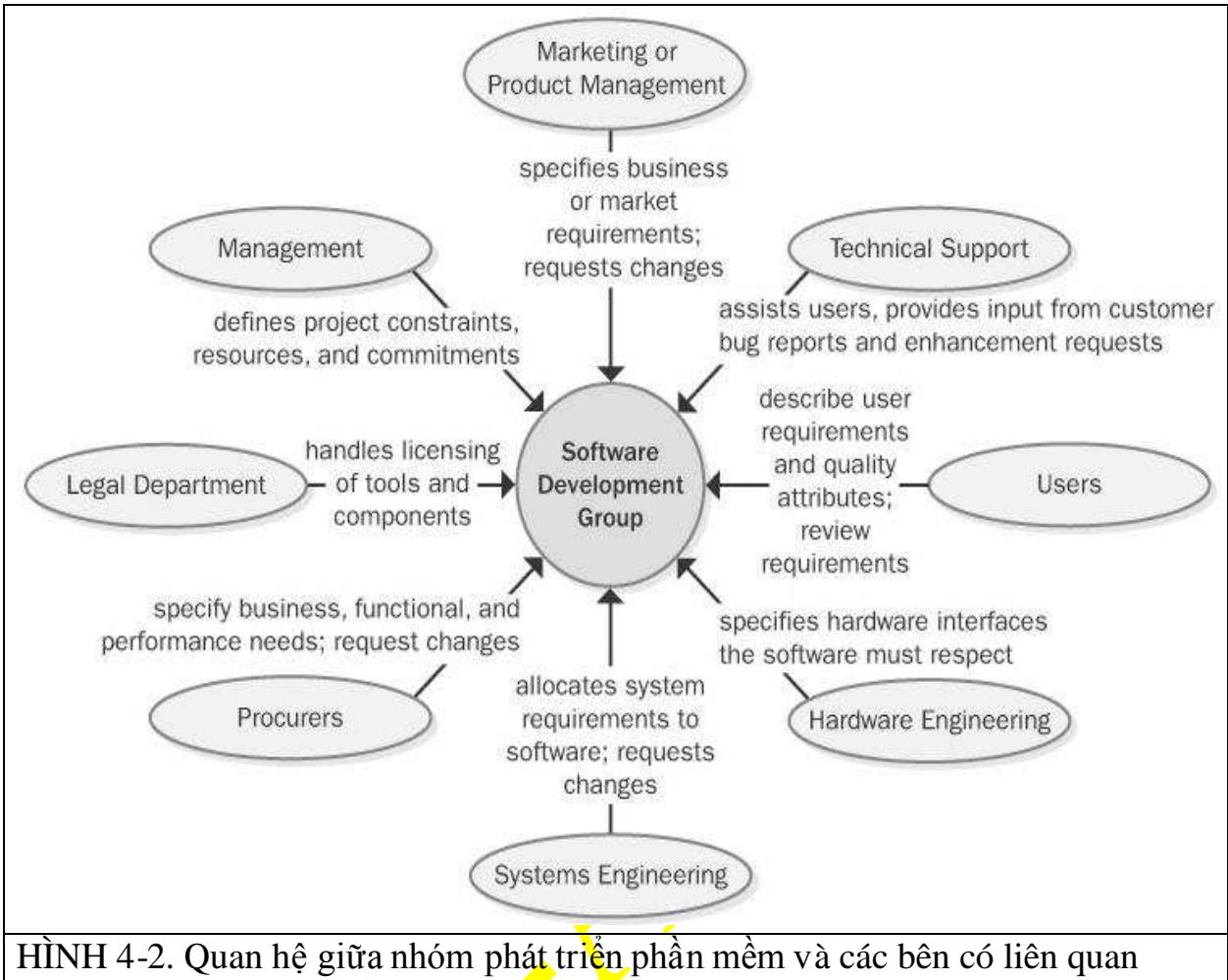
Quy trình Thi công hệ thống (Construction process)

Phần mềm có thể chạy (executable software) chứ không phải các tài liệu yêu cầu là sản phẩm phải chuyển giao cho khách hàng của một dự án phần mềm. Các yêu cầu là cơ sở để thiết kế và thi công phần mềm. Yêu cầu chức năng (functional requirements) dẫn tới các thiết kế components, chúng phục vụ như là các đặc tả cho các mã sẽ được viết. Thực hiện soát xét thiết kế để đảm bảo bản thiết kế đã chứa tất cả các yêu cầu. Kiểm thử đơn vị (unit testing) mã nguồn có thể xác định liệu nó có đáp ứng đặc tả thiết kế và yêu cầu tương ứng hay không.

II. ẢNH HƯỞNG CỦA YÊU CẦU PHẦN MỀM TỚI NHỮNG NGƯỜI CÓ LIÊN QUAN KHÁC CỦA DỰ ÁN (IMPACT OF SOFTWARE REQUIREMENTS ON OTHER STAKEHOLDERS)

Khi nhóm phát triển phần mềm thay đổi quy trình yêu cầu của họ thì sự thay đổi này sẽ ảnh hưởng tới những người có liên quan của dự án. Hình 4-2 thể hiện ảnh hưởng của yêu cầu phần mềm tới những người có liên quan.





HÌNH 4-2. Quan hệ giữa nhóm phát triển phần mềm và các bên có liên quan

Đó là:

- Nhóm marketing (Marketing or Product Management): đặc tả yêu cầu kinh doanh hoặc yêu cầu của thị trường cho nhóm phát triển; đề xuất các thay đổi đối với nhóm phát triển.
- Nhóm hỗ trợ kỹ thuật (Technical Support): hỗ trợ người dùng của khách hàng, cung cấp đầu vào cho nhóm phát triển từ việc phân tích các báo cáo lỗi của khách hàng, đề nghị các thay đổi nâng cấp phần mềm.
- Người phụ trách dùng (Users): mô tả các yêu cầu người dùng và thuộc tính chất lượng của các yêu cầu đó, soát xét các yêu cầu.
- Nhóm kỹ thuật phần cứng (Hardware Engineering): đặc tả các giao diện phần cứng mà phần mềm phải làm việc cùng.
- Nhóm kỹ thuật hệ thống (Systems Engineering): phân bổ các yêu cầu hệ thống cho phần mềm, đề xuất thay đổi.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- Nhóm mua sắm (Procuers): đặc tả các nhu cầu kinh doanh, chức năng và hiệu suất; đề xuất thay đổi.
- Nhóm pháp lý (Legal Department): xử lý các vấn đề pháp luật liên quan đến license của các tools và components.
- Cấp quản lý (Management): đề ra các ràng buộc của dự án, ràng buộc về tài nguyên và cam kết khác cho nhóm phát triển.

III. CƠ BẢN VỀ CẢI TIẾN QUY TRÌNH PHẦN MỀM (FUNDAMENTALS OF SOFTWARE PROCESS IMPROVEMENT)

Bạn đọc cuốn sách này vì mong muốn làm tốt hơn công nghệ yêu cầu, hãy ghi nhớ 4 nguyên lý sau khi thực hiện cải tiến quy trình phần mềm (Wiegers 1996a):

1. *Cải tiến quy trình cần được thực hiện theo kiểu tiến hóa, liên tục và có chu trình (Process improvement should be evolutionary, continuous, and cyclical).* Đừng nghĩ sẽ cải tiến tất cả các quy trình của bạn trong một lúc, hãy chấp nhận rằng bạn không thể làm được tất cả mọi thứ đúng ngay từ đầu, ngay khi bạn bắt đầu thực hiện sự thay đổi. Thay vì cố đạt tới sự hoàn hảo, hãy cải tiến từng chút một và thực hiện nó cẩn thận. Bạn có thể điều chỉnh cách tiếp cận khi bạn đã có được kinh nghiệm với các cải tiến trước đó.
2. *Con người và các tổ chức chỉ thay đổi khi họ bị thúc ép phải thay đổi (People and organizations change only when they have an intence to do so).* Sức thúc ép mạnh nhất chính là sự đau khổ khi làm việc theo cách cũ. Tôi không có ý nói đến các đau khổ, chẳng hạn, như các lịch biểu bị siết chặt do nhu cầu của khách hàng, tôi muốn nói đến các đau khổ mà bạn đã trải qua từ các dự án trước. Những đau khổ là động lực khiến một người quản lý nói: “Cuốn sách này nói chúng ta phải làm một số thứ, vậy nên bắt tay vào làm đi!”. Một số thống kê từ các dự án trước có thể định hướng cho việc thay đổi quy trình yêu cầu:
 - Dự án bị lỗi hẹn (missed deadlines) do các yêu cầu quá phức tạp so với sự hình dung.
 - Nhà phát triển phải làm việc vất vả nhiều giờ do các yêu cầu không được hiểu rõ hoặc các yêu cầu nhập nhằng nhưng lại được đề xuất vào phút cuối quá trình phát triển sản phẩm.

- Các nỗ lực kiểm thử hệ thống quá ít tác dụng do các kiểm thử viên không hiểu hệ thống làm gì.
- Chức năng đúng đã được thực hiện nhưng người dùng không hài lòng vì sự kém cỏi của nó như tính tiện dụng thấp chẳng hạn.
- Tổ chức phát triển đã có kinh nghiệm về các sản phẩm với chi phí bảo trì cao do khách hàng đề xuất nhiều yêu cầu nâng cấp trong giai đoạn suy luận yêu cầu.
- Tổ chức phát triển đã nhiều lần đổi mới với sự từ chối của khách hàng khi bàn giao sản phẩm.

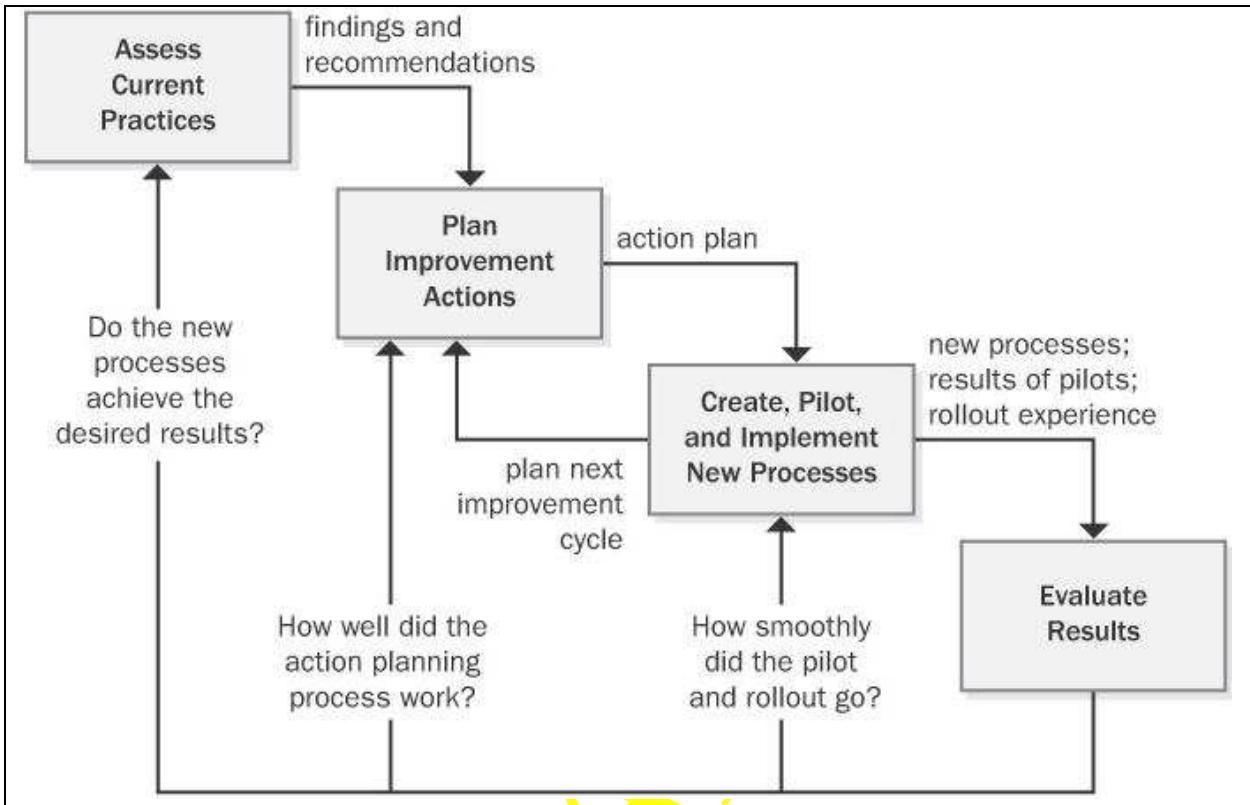
3. Các thay đổi quy trình cần phải được hướng đích (Process changes should be goal oriented). Trước khi bắt đầu hành trình thay đổi quy trình, hãy đảm bảo chắc chắn rằng bạn đã biết những gì bạn phải đổi mới. Bạn có muốn giảm bớt lượng công việc phải làm lại do các vấn đề liên quan đến yêu cầu? Bạn có muốn kiểm soát tốt hơn cách tích hợp các thay đổi yêu cầu vào dự án? Bạn có muốn chắc chắn rằng không có yêu cầu nào bị bỏ qua khi thi công? Một lộ trình vạch ra con đường bạn phải đi sẽ nâng cơ may thành công cho bạn khi cải tiến quy trình.

4. Xử lý các hoạt động cải tiến quy trình của bạn như là các tiểu dự án (Treat improvement activities as miniprojects). Nhiều sáng kiến cải tiến quy trình đã đổ vỡ vì được lập kế hoạch thực hiện một cách sơ sài, vì các cam kết nguồn lực dành cho chúng chưa bao giờ thành hiện thực. Để tránh các vấn đề đó, hãy xử lý các hoạt động cải tiến quy trình như các dự án. Hãy ghép các nguồn lực và công việc (tasks) cải tiến quy trình vào kế hoạch tổng thể của dự án. Hãy lập kế hoạch, giám sát, đo lường và tổng kết báo cáo về công việc đó như một dự án bình thường. Hãy viết một kế hoạch hành động cho mỗi hoạt động cải tiến quy trình. Hãy giám sát thời gian và chi phí mà những người tham gia cải tiến quy trình đã sử dụng nhằm nắm bắt được tốc độ cải tiến.

IV. CHU TRÌNH CẢI TIẾN QUY TRÌNH (THE PROCESS IMPROVEMENT CYCLE)

Hình 4-3 minh họa một chu trình cải tiến quy trình mà tôi thấy là rất hiệu quả. Chu trình này nói về tầm quan trọng của việc biết nơi bạn đang đứng trước khi di chuyển sang một vị trí khác, nói về sự cần thiết của việc lập kế hoạch cho các hoạt. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

động cải tiến và học hỏi từ chính kinh nghiệm của bạn như là một phần của sự cải tiến liên tục.



Assess Current Practices = Đánh giá các practices hiện tại

Plan Improvement Actions = Lập kế hoạch cho các hoạt động cải tiến

Create, Pilot and Implement New Processes = Thiết lập, thử nghiệm, cải tiến các quy trình mới

Evaluate Results = Đánh giá kết quả

HÌNH 4.3. Chu trình cải tiến quy trình phần mềm

1. ĐÁNH GIÁ CÁC PRACTICES HIỆN TẠI (ASSESS CURRENT PRACTICES)

Bước đầu tiên trong bất cứ hoạt động cải tiến quy trình nào cũng là đánh giá các practices hiện tại đang được sử dụng trong tổ chức để xác định thế mạnh và hạn chế của nó. Một bản đánh giá không tự nó gợi nên bất cứ cải tiến nào, nó chỉ cung cấp thông tin giúp chọn được những cách làm đúng đắn có những thay đổi mà bạn muốn.

Bạn có thể đánh giá các quy trình hiện tại theo nhiều cách. Nếu bạn cố gắng thực hiện bất cứ bước nào trong hộp **Các bước tiếp theo** ở cuối các chương trước thì bạn đã bắt đầu một đánh giá không chính thức về các requirements practices và các kết quả của chúng. Các bảng hỏi tự đánh giá có cấu trúc mang lại cho bạn một cách tiếp cận hệ thống hơn, bạn có thể thấy được các vấn đề bên trong của các quy trình hiện tại mà không cần phải mất quá nhiều công sức.

Cách tiếp cận chi tiết và khách quan hơn là mời các chuyên gia tư vấn bên ngoài đánh giá các software practices của bạn. Các đánh giá tốt nên dựa trên một khung đánh giá quy trình như CMMI chẳng hạn. Các đánh giá viên sẽ kiểm tra và đánh giá tất cả các quy trình quản lý và phát triển của bạn và không chỉ khuôn trong các hoạt động liên quan đến yêu cầu. Hãy chọn một cách đánh giá đáp ứng được các yêu cầu kinh doanh (business requirements) của bạn, không cần thiết nó có phù hợp CMMI hoặc một mô hình cụ thể nào khác.

Phần **Phụ lục** chứa một bảng hỏi để bạn tự đánh giá các requirements practices của mình. Sử dụng bảng hỏi này để có cái nhìn toàn cảnh về các practices trong công nghệ yêu cầu (requirements engineering practices) của bạn, bạn sẽ biết quy trình yêu cầu (requirements processes) nào cần cải tiến nhất trong tổ chức của bạn. Hãy tập trung sức lực của bạn để cải tiến những practices nào đang và sẽ gây ra cho các dự án của bạn những khó khăn và những rủi ro cao nhất. Mỗi câu hỏi trong bảng tự đánh giá tham chiếu tới một chương trong cuốn sách này. Motorola đã phát triển một bảng hỏi tương tự là “Software Requirements Quality Model” (Smith 1998).

Kết quả đạt được sau một đánh giá chính thức là các số liệu (findings) – báo cáo điểm mạnh và điểm yếu của các quy trình đang sử dụng – và các khuyến nghị (recommendations) để nhận diện các cơ hội cải tiến. Các đánh giá không chính thức, như bảng hỏi tự đánh giá, cung cấp cho bạn những hiểu biết sâu bên trong để lựa chọn những gì cần cải tiến. Bạn sẽ tìm thấy nhiều khuyến nghị tổng quan (general recommendations) trong cuốn sách này về các câu hỏi tự đánh giá. Hãy phân tích mỗi hành động cải tiến để chắc chắn về chi phí – hiệu quả của nó.

2. LẬP KẾ HOẠCH CHO CÁC HOẠT ĐỘNG CẢI TIẾN (PLAN IMPROVEMENT ACTIONS)

Nhằm tuân theo triết lý của hoạt động cải tiến quy trình là coi các hoạt động cải tiến như là các dự án, hãy viết một kế hoạch hành động sau khi bạn đã có các đánh giá. Hãy cân nhắc liệu có nên viết một kế hoạch chiến lược (strategic plan) tổng thể mô tả các sáng kiến cải tiến quy trình của doanh nghiệp của bạn hay không, cũng như các kế hoạch hành động mang tính chiến thuật (tactical action plans) xử lý các cải tiến cụ thể. Mỗi kế hoạch chiến thuật cần chỉ ra mục đích của các hoạt động cải tiến, những người tham gia, một số hành động cụ thể cần hoàn thành để thực thi kế hoạch. Hình 4-4 mô tả một template kế hoạch hành động cải tiến quy trình mà tôi đã sử dụng nhiều lần.

Kế hoạch hành động cải tiến quy trình	
Dự án: <Tên dự án>	
Ngày: <Ngày dự án được viết>	
Mục đích (Goals): <Mô tả mục đích cần đạt được của kế hoạch này. Hãy viết các mục đích định hướng đạt được các kết quả kinh doanh (business results), chứ không phải trong khuôn khổ của sự thay đổi quy trình>	
Đo lường thành công (Measures of Success): <Hãy mô tả bạn sẽ xác định như thế nào các hiệu quả mong muốn đối với dự án khi các thay đổi quy trình được thực hiện>	
Phạm vi ảnh hưởng đến tổ chức (Scope of Organizational Impact): <Mô tả tầm ảnh hưởng đối với tổ chức của các thay đổi quy trình được mô tả trong kế hoạch này>	
Những người tham gia (Staffing and Participants): <Xác định các cá nhân sẽ thực hiện kế hoạch này, vai trò của họ, cam kết thời gian tham gia của họ>	
Quy trình giám sát và báo cáo (Tracking and Reporting Process): <Hãy mô tả cách giám sát tiến độ cải tiến và tình trạng, kết quả đạt được, các vấn đề sẽ được báo cáo>	
Các phụ thuộc, rủi ro và ràng buộc (Dependencies, risks, and constraints): <Xác định bất cứ yếu tố bên ngoài nào cần thiết đối với sự thành công của dự án, hoặc ảnh hưởng xấu đến sự thành công của dự án khi thực hiện kế hoạch này>	

Ngày hoàn thành ước tính của dự án (Estimated completion date for all activites):

<Khi nào thì bạn sẽ thực hiện xong tất cả các công việc này?>

CÁC CÔNG VIỆC CẦN THỰC HIỆN (ACTION ITEMS)						
Số thứ tự (Action item)	Người thực hiện (Responsible individual)	Ngày hoàn thành (Due date)	Mục đích (Purpose)	Mô tả các hoạt động (Description of Activities)	Các kết quả đạt được (Deliverables)	Nguồn lực cần thiết (Resources needed)
	<Cá nhân chịu trách nhiệm>			<Tất cả các hoạt động sẽ được thực hiện để hoàn thành mục công việc này>	<Các thủ tục, templates, các tài sản quy trình khác sẽ được tạo ra>	<Bắt cứ nguồn lực cần thiết nào gồm vật liệu, công cụ, tài liệu, hoặc con người>

HÌNH 4-4. Template kế hoạch hành động cải tiến quy trình phần mềm

3. THIẾT LẬP, THỬ NGHIỆM, CẢI TIẾN CÁC QUY TRÌNH MỚI (CREATE, PILOT, AND IMPLEMENT NEW PROCESSES)

Bạn đã đánh giá các requirements practices của bạn và đã chuẩn bị một kế hoạch để cải tiến quy trình. Bây giờ là phần khó khăn: thực hiện quy trình. Nhiều sáng kiến cải tiến quy trình đã rơi vào tình trạng khó khăn khi đưa kế hoạch hành động vào thực tiễn.

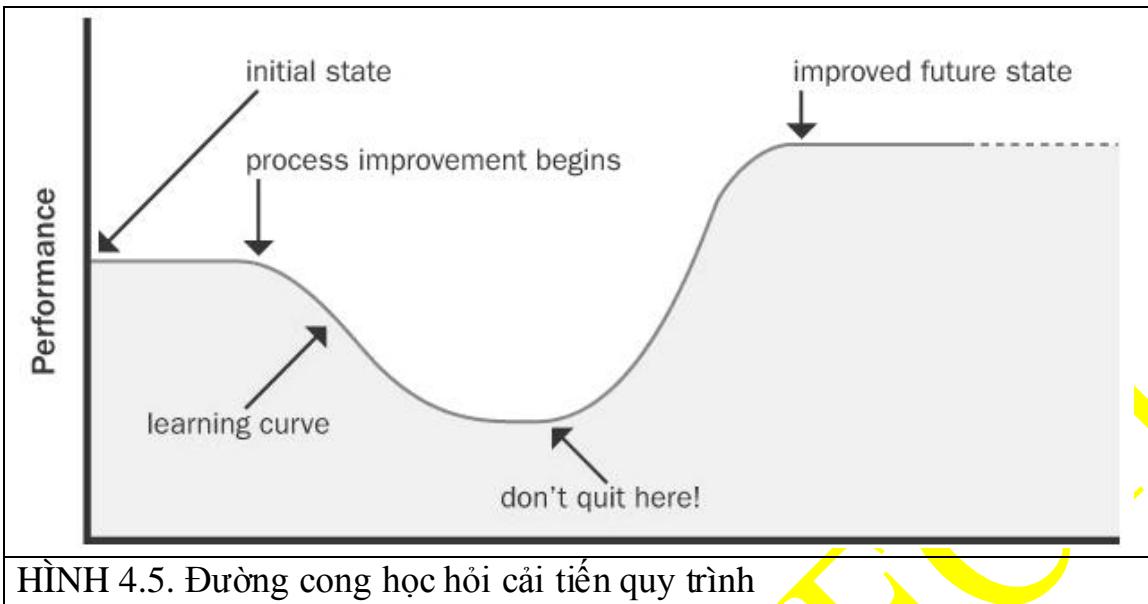
Thực hiện một kế hoạch hành động nghĩa là xây dựng các quy trình mới hoặc làm cho quy trình cũ tốt hơn. Tuy nhiên, bạn không nên hy vọng sẽ nhận được quy trình mới hoàn hảo ngay từ đầu. Nhiều cách tiếp cận dường như là một ý tưởng tốt nhưng lại kém hiệu thực hoặc không hiệu quả khi thực hiện. Vì vậy, hãy lập một kế hoạch thử nghiệm (process “pilot”) cho các thủ tục mới hoặc các templates mà bạn tạo ra. Sử dụng kiến thức, kinh nghiệm mà bạn thu thập được trong quá trình thử nghiệm để điều chỉnh các quy trình mới. Ghi nhớ các lời khuyên sau khi bạn thử nghiệm các quy trình mới:

- Lựa chọn những người tham gia thử nghiệm các quy trình mới và sau đó đưa ra đánh giá phản hồi. Họ nên là những người có tính hoài nghi cao nhưng lại không phản đối cải tiến quy trình.
- Lượng hoá các tiêu chuẩn mà bạn sử dụng để đánh giá cuộc thử nghiệm, hãy tạo ra các kết quả dễ dàng diễn giải.
- Xác định những người có liên quan đến dự án cần được thông tin về việc thử nghiệm.
- Cân nhắc việc thử nghiệm các quy trình mới trên các dự án khác nhau để tăng mức độ thử thách của quy trình.
- Như là một phần của việc đánh giá, hãy hỏi những người tham gia thử nghiệm xem họ cảm thấy thế nào nếu phải quay lại cách làm việc cũ.

4. ĐÁNH GIÁ KẾT QUẢ (EVALUATE RESULTS)

Bước cuối cùng là đánh giá các hoạt động và kết quả đạt được, việc này giúp bạn điều chỉnh các hoạt động cải tiến quy trình tiếp theo. Bạn cũng cần xem xét các quy trình mới đã được những người liên quan biết rõ để thực hiện hay chưa.

Hãy chú ý Hình 4-5 khi bạn ứng dụng các quy trình mới, sự sụt giảm năng suất làm việc sẽ diễn ra ngay sau khi ứng dụng. Nhưng sau một thời gian năng suất làm việc sẽ tăng lên và cao hơn năng suất trước khi bạn ứng dụng quy trình mới.



HÌNH 4.5. Đường cong học hỏi cải tiến quy trình

V. TÀI SẢN QUY TRÌNH YÊU CẦU (REQUIREMENTS PROCESS ASSETS)

Nếu bạn muốn các dự án được hoàn thành với các kết quả tốt hơn thì bạn cần có các quy trình hiệu quả trên toàn bộ các giai đoạn của công nghệ yêu cầu: **suy luận, phân tích, đặc tả, kiểm tra, quản lý**. Tập hợp các quy trình thực hiện các công việc trên được gọi là *tài sản quy trình*. Một quy trình hướng dẫn các hành động mà bạn cần thực hiện để có được các kết quả mà bạn cần chuyển giao; tài sản quy trình giúp các bên liên quan tới dự án thực hiện nhất quán và hiệu quả công việc của họ. Tài sản quy trình gồm các loại tài liệu sau:

Loại tài liệu	Nội dung
Danh sách kiểm tra (Checklist)	Một danh sách liệt kê các hoạt động, các kết quả chuyển giao (deliverables), hoặc các mục đích khác (items) được đánh dấu hoặc được kiểm tra (noted or verified). Checklist đảm bảo những ai thực hiện công việc không bỏ qua bất cứ chi tiết quan trọng nào.
Ví dụ (Example)	Minh họa cho một bản thành phẩm cụ thể. Tích lũy các ví dụ tốt hơn theo thời gian để minh họa tốt hơn cho những dự án sau.
Kế hoạch (Plan)	Tài liệu khái quát về việc làm thế nào để đạt được một mục đích và những gì cần làm để đạt được mục đích đó.
Chính sách (Policy)	Một nguyên tắc hướng dẫn thiết lập sự kỳ vọng về các hành vi, hành động, các kết quả chuyển giao. Các quy trình cần phải sống

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

	đối với các chính sách.
Thủ tục (Procedures)	Mô tả từng bước một về các công việc (tasks) nối tiếp nhau nhằm hoàn thành một hoạt động (activity). Mô tả các công việc (tasks) được thực hiện và chỉ ra các vai trò (roles) sẽ thực hiện chúng. Tránh kèm các thông tin hướng dẫn (tutorial information) trong một thủ tục.
Mô tả quy trình (Process description)	Một định nghĩa được ghi thành văn về một tập hợp các hoạt động cần thực hiện nhằm đạt được mục đích nào đó. Mô tả quy trình có thể bao gồm mục tiêu của quy trình, các mốc chính (key milestones), người tham gia, thời gian thích hợp để thực hiện hoạt động, các bước truyền thông, các kết quả mong muốn, dữ liệu đầu vào và đầu ra của quy trình (Caputo 1998).
Mẫu (Template)	Một mẫu được sử dụng như một hướng dẫn để sản xuất một bản thành phẩm trọn vẹn (complete work product). Các templates cho các tài liệu chính của dự án nhắc bạn không bỏ qua những gì quan trọng. Một template tốt sẽ cung cấp nhiều cách thức nắm bắt và tổ chức thông tin. Các hướng dẫn trong template giúp người dùng sử dụng hiệu quả template hơn. Hình 4-4 chính là một template.

Hình 4-6 xác định một số tài sản quy trình mà bạn cần có để phát triển và quản lý các yêu cầu hiệu quả hơn.

Tài sản quy trình phát triển yêu cầu (Requirements Development Process Assets)	Tài sản quy trình quản lý yêu cầu (Requirements Management Process Assets)
<ul style="list-style-type: none"> • Template tầm nhìn và phạm vi của dự án (Project vision and Scope Template) • Thủ tục phát triển yêu cầu (Requirements Development Procedure) • Thủ tục phân bổ yêu cầu 	<ul style="list-style-type: none"> • Thủ tục kiểm soát thay đổi (Change Control Procedure) • Thủ tục ban kiểm soát thay đổi (Change Control Board Procedure) • Checklists và template phân tích ảnh hưởng thay đổi yêu cầu (Requirements Change Impact

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

<p>(Requirements Procedure)</p> <ul style="list-style-type: none"> Template tình huống sử dụng (Use case template) Template đặc tả yêu cầu phần mềm (SRS template) Thủ tục xếp thứ tự ưu tiên các yêu cầu (Requirements Prioritization Procedure) Các checklists thanh tra SRS và Use cases (SRS and Use case Inspection Checklists) 	<p>Allocation</p> <ul style="list-style-type: none"> Analysis Checklist and Template) <ul style="list-style-type: none"> Thủ tục giám sát tình trạng yêu cầu (Requirements Status Tracking Procedure) Template ma trận lằn vết yêu cầu (Requirements Traceability Matrix Template)
<p>HÌNH 4-6. Các tài sản quy trình chính cho phát triển và quản lý yêu cầu</p>	

Các thủ tục được liệt kê trong Hình 4-6 không cần thiết được viết thành các tài liệu riêng rẽ. Ví dụ, một mô tả về quy trình quản lý yêu cầu tổng thể (overall requirements management process) có thể bao gồm thủ tục kiểm soát thay đổi, thủ tục giám sát tình trạng, và danh sách kiểm tra (checklist) phân tích ảnh hưởng. Ví dụ về một mô tả quy trình quản lý yêu cầu xem **Phụ lục J** của *CMM Implementation Guide* (Caputo 1998).

Sau đây là các mô tả ngắn gọn về mỗi tài sản quy trình trong Hình 4-6, cùng các tham chiếu tới các chương sẽ thảo luận chi tiết về chúng. Cần ghi nhớ rằng đối với mỗi dự án đều phải “may đo lại” các thủ tục để đáp ứng các nhu cầu cụ thể của tổ chức.

1. TÀI SẢN QUY TRÌNH PHÁT TRIỂN YÊU CẦU (REQUIREMENTS DEVELOPMENT PROCESS ASSETS)

Template tầm nhìn và phạm vi của dự án (Project vision and Scope Template)

Tài liệu tầm nhìn và phạm vi (vision and scope) định nghĩa nền tảng ý tưởng (conceptual foundation) của dự án và cung cấp một tham chiếu để ra quyết định về các thứ tự ưu tiên của yêu cầu và các thay đổi. Tầm nhìn và phạm vi (vision and scope) là một mô tả ở cấp độ cao và chính xác về các yêu cầu kinh doanh (business requirements) của phần mềm cần làm. Viết tài liệu này theo một cách nhất quán sẽ *Cuốn sách Công nghệ thông tin*, từ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

đảm bảo tất cả các vấn đề đúng (right issues) đều được cân nhắc khi ra quyết định thực hiện dự án. Chương 6 gợi ý một template cho tài liệu này.

Thủ tục phát triển yêu cầu (Requirements Development Procedure)

Thủ tục này mô tả làm thế nào để định danh các khách hàng và các kỹ thuật suy luận yêu cầu từ họ. Nó cũng mô tả các tài liệu yêu cầu khác nhau và mô hình phân tích thích hợp. Thủ tục có thể chỉ ra loại thông tin kèm theo mỗi yêu cầu, ví dụ mức ưu tiên, độ ổn định dự kiến của yêu cầu, hoặc số phiên bản phần mềm dự định. Thủ tục cũng xác định các bước mà dự án phải thực hiện để phân tích và kiểm tra yêu cầu. Nó cũng bao gồm các bước cần thiết để chấp nhận SRS và thiết lập ranh giới yêu cầu.

Thủ tục phân bổ yêu cầu (Requirements Allocation Procedure)

Phân bổ các yêu cầu sản phẩm ở cấp độ cao (high-level product requirements) thành các hệ thống con cụ thể là việc quan trọng khi phát triển các hệ thống bao gồm cả phần cứng và phần mềm hoặc các sản phẩm phần mềm phức hợp (complex software product) chứa nhiều hệ thống con (Nelsen 1990). Phân bổ được thực hiện sau khi các yêu cầu mức hệ thống đã được xác định và kiến trúc hệ thống đã được định nghĩa. Thủ tục này chứa thông tin về việc làm thế nào để thực hiện sự phân bổ nhằm đảm bảo các chức năng đúng (right functionality) được gán cho đúng thành phần hệ thống (system component) thích hợp. Thủ tục cũng mô tả làm thế nào các yêu cầu đã phân bổ lần lượt được về yêu cầu hệ thống gốc và các yêu cầu liên quan trong các hệ thống con khác.

Template tình huống sử dụng (Use case template)

Mẫu use case cung cấp một cách thức tiêu chuẩn để tài liệu hóa mỗi công việc (tasks) mà người dùng mong muốn thực hiện với hệ thống. Một định nghĩa use-case gồm một mô tả ngắn về công việc (task), mô tả về các hành vi có thể lựa chọn (alternative behaviors) hoặc các loại trừ đã biết (known exceptions) cần phải được xử lý, thông tin kèm theo đặc tả theo công việc (task) người dùng. Use-cases có thể được phác thảo (elaborate) theo các yêu cầu chức năng (functional requirements) riêng biệt trong SRS. Bạn cũng có thể kết hợp các template use-case và SRS thành một tài liệu duy nhất chứa các yêu cầu người dùng (user

requirements) và yêu cầu chức năng (functional requirements). Chương 8 đề xuất một khuôn dạng (format) use-case template.

Template đặc tả yêu cầu phần mềm (SRS template)

SRS template (software requirement specification template) cung cấp một cách thức có cấu trúc để tổ chức các yêu cầu chức năng (functional requirements) và yêu cầu phi chức năng. Một tổ chức thông qua một SRS template tiêu chuẩn sẽ nâng cao rất nhiều chất lượng của yêu cầu. Bạn có thể thông qua nhiều templates khác nhau để thích hợp với nhiều loại hình dự án khác nhau, tránh làm kiểu “one size fits all” (một kích thước cho tất cả). Chương 9 mô tả một SRS template.

Thủ tục xếp thứ tự ưu tiên các yêu cầu (Requirements Prioritization Procedure)

Bạn tôi Matt mô tả giai đoạn cuối của một dự án phần mềm điển hình là “giai đoạn cắt xén nhanh” khi chức năng đã được lập kế hoạch bị xoá bỏ vào phút cuối để đáp ứng hạn cuối của lịch biểu. Để cắt giảm phạm vi tại bất cứ giai đoạn nào, chúng ta cũng phải biết tính năng nào, use-cases nào, yêu cầu chức năng (functional requirements) nào có mức ưu tiên thấp nhất. Chương 13 đề xuất một thủ tục được xếp thứ tự ưu tiên để bạn tham khảo cùng nhiều thông tin khác liên quan đến rủi ro kỹ thuật, chi phí tương đối của việc cài đặt mỗi use-case, mỗi tính năng hoặc yêu cầu.

Các checklists thanh tra SRS và use-case (SRS and use -case Inspection Checklists)

Thanh tra chính thức tài liệu yêu cầu là một biện pháp quan trọng để đảm bảo chất lượng. Một inspection checklist định danh được khá nhiều trong số các lỗi thông thường được tìm thấy ở tài liệu yêu cầu. Chương 14 đề xuất các ví dụ về SRS inspection checklist và use-case inspection checklist.

2. TÀI SẢN QUY TRÌNH QUẢN LÝ YÊU CẦU (REQUIREMENTS MANAGEMENT PROCESS ASSETS)

Thủ tục kiểm soát thay đổi (Change control procedure)

Một quy trình kiểm soát thay đổi có thể làm giảm các hỗn độn trong dự án bởi các thay đổi yêu cầu không kiểm soát được và không biết bao giờ thì ngừng. Thủ tục Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

kiểm soát thay đổi định nghĩa cách một yêu cầu mới hoặc một sửa đổi yêu cầu cũ được đề xuất, được truyền thông, được đánh giá và được quyết định chấp nhận hay không. Kiểm soát thay đổi thường được trợ giúp bởi một công cụ giám sát nhưng một công cụ thì không thay thế cho quy trình được. Chương 17 chỉ ra quy trình kiểm soát thay đổi một cách chi tiết.

Thủ tục ban kiểm soát thay đổi (Change Control Board Procedure)

Ban kiểm soát thay đổi (CCB) là một nhóm bao gồm một số người liên quan đến dự án phần mềm (stakeholders) ra các quyết định liên quan đến thay đổi yêu cầu, liệu các thay đổi đó được chấp thuận hay từ chối. Thủ tục CCB mô tả cơ cấu và hoạt động của nhóm. Các hoạt động chính của CCB là phân tích ảnh hưởng của các thay đổi, ra quyết định về mỗi thay đổi, thông báo đến tất cả những ai liên quan về các thay đổi. Chương 17 định nghĩa về cơ cấu và chức năng của CCB.

Checklists và templates phân tích ảnh hưởng thay đổi yêu cầu (Requirements Change Impact Analysis Checklist and Template)

Ước lượng chi phí và ảnh hưởng của một thay đổi là việc làm quan trọng trước khi chấp nhận hoặc từ chối thay đổi đó. Như được minh họa trong Chương 18, một checklist phân tích ảnh hưởng chứa nhiều câu hỏi đòi hỏi bạn suy nghĩ về các công việc (tasks) có thể, các hiệu ứng phụ (side effects), các rủi ro tiềm tàng liên quan tới sự thực hiện một thay đổi yêu cầu cụ thể. Chương 18 cung cấp một template cho vấn đề này.

Thủ tục giám sát tình trạng yêu cầu (Requirements Status Tracking Procedure)

Quản lý yêu cầu bao gồm giám sát và báo cáo tình trạng của mỗi yêu cầu chức năng (functional requirement) và điều kiện mà theo đó trạng thái này có thể thay đổi. Bạn nên sử dụng một cơ sở dữ liệu hoặc một công cụ quản lý yêu cầu để giám sát một số lượng lớn các yêu cầu trong một hệ thống phức tạp. Thủ tục này cũng mô tả các báo cáo mà bạn cần phải có để giám sát bất cứ khi nào có thể. Chương 16 nói rõ về vấn đề này.

Template ma trận lằn vết yêu cầu (Requirements Traceability Matrix Template)

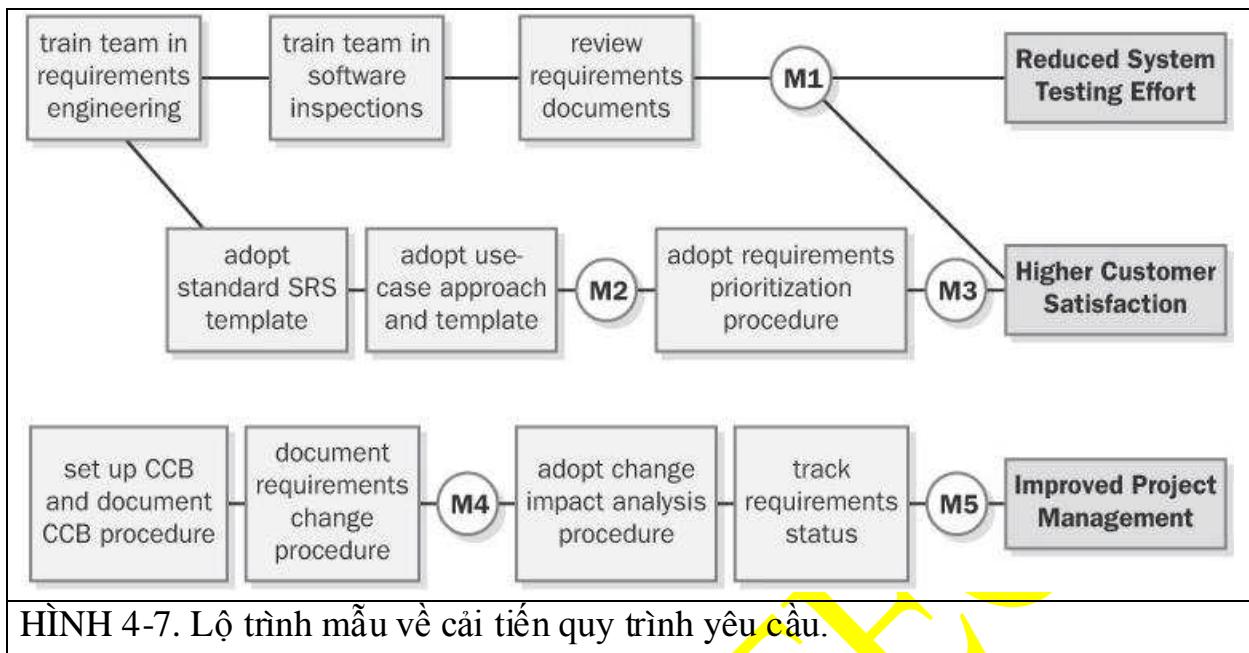
Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Ma trận lần vết yêu cầu liệt kê tất cả các yêu cầu chức năng (functional requirements) từ SRS, cùng các thành phần thiết kế (design components) của mỗi yêu cầu, các tệp nguồn (source files) và thủ tục (procedures) cài đặt các yêu cầu, các tình huống kiểm thử (test cases) kiểm tra tính đúng đắn của việc cài đặt. Ma trận lần vết yêu cầu cũng sẽ xác định người dùng hoặc yêu cầu hệ thống mà từ đó yêu cầu chức năng (functional requirements) được dẫn xuất ra. Chương 18 đề xuất một template ma trận lần vết yêu cầu.

VI. LỘ TRÌNH CẢI TIẾN QUY TRÌNH YÊU CẦU (REQUIREMENTS PROCESS IMPROVEMENT ROADMAP)

Cải tiến quy trình yêu cầu không phải là công việc dễ dàng. Những cách tiếp cận không có định hướng đối với vấn đề này sẽ không bao giờ dẫn tới thành công cả. Bạn cần phải xây dựng một lộ trình để thực thi các practices yêu cầu đã cải tiến (improved requirements practices). Lộ trình này cần phải là một phần của kế hoạch cải tiến quy trình.

Do tình trạng của mỗi doanh nghiệp là khác nhau nên tôi không thể đưa ra một lộ trình phù hợp với tất cả (one-size-fits-all). Cách tiếp cận công thức hoá không thể thay thế cho tư duy và cảm nghĩ của chính bạn. Hình 4-7 minh họa một lộ trình cải tiến quy trình của một doanh nghiệp. Các kết quả kinh doanh mong muốn (desired business results) được thể hiện trong các hộp chữ nhật in đậm bên phải. Các hoạt động cải tiến chính được thể hiện trong các hộp khác và trong một số mốc (milestones) trung gian (hình vòng tròn). Thực hiện các hoạt động cải tiến quy trình trong các hộp chữ nhật từ trái qua phải. Sau khi bạn đã tạo ra một lộ trình tương tự, hãy chuyển cho mỗi người chịu trách nhiệm về một mốc, họ sẽ viết một kế hoạch hành động để đạt được mốc đó. Công việc sau đó là đưa kế hoạch vào hành động.



Các bước tiếp theo

- Hoàn thành các **Bảng tự đánh giá Requirements Practices hiện tại** trong Phụ lục. Xác định 3 cơ hội cải tiến requirements practices cao nhất của bạn căn cứ trên các hạn chế của quy trình hiện tại.
- Xác định trong số các tài sản quy trình công nghệ yêu cầu ở Hình 4-6, tài sản quy trình nào chưa thực sự hữu ích trong tổ chức nhưng bạn lại nghĩ nó vẫn được sử dụng tốt.
- Dựa trên 2 bước trên, xây dựng một lộ trình cải tiến quy trình yêu cầu như template ở Hình 4-7. Giao cho mỗi ai đó thực hiện một mốc trên lộ trình. Yêu cầu mỗi người viết kế hoạch hành động để đạt được mốc đó, sử dụng template về kế hoạch hành động trong Hình 4-4. Giám sát tiến độ thực hiện kế hoạch cải tiến.

CHƯƠNG 5

YÊU CẦU PHẦN MỀM VÀ QUẢN LÝ RỦI RO

Dave, quản lý dự án của Chemical Tracking System tại Contoso Pharmaceuticals, đang có cuộc gặp với trưởng nhóm lập trình Helen và trưởng nhóm kiểm thử Ramesh. Tất cả đều đang vui mừng về dự án mới và họ nhớ lại một số vấn đề đã gặp trong dự án trước đây gọi là Pharm-Simulator.

“Hãy nhớ lại xem chúng ta đã không phát hiện ra người dùng ghét cái giao diện của Simulator như thế nào cho đến tận lần kiểm thử beta?” Helen hỏi. “Chúng ta đã mất 5 tuần để xây dựng lại hệ thống và kiểm thử lại. Tôi không hề muốn lặp lại việc đó một lần nữa.”

“Chả vui vẻ tí nào cả”, Dave nói. “Điều khó chịu nữa là người dùng nói họ muốn rất nhiều tính năng nhưng thực tế thì chả ai sử dụng sau này cả. Vậy hãy trao đổi với họ ít nhất 3 lần trước khi biết chúng ta có nên viết một tính năng nào đó, nếu không hãy bỏ nó đi ngay. Vớ vẩn thật!”

“Chúng ta đã xông vào Simulator quá sớm và không có đủ thời gian để viết các yêu cầu chi tiết,” Ramesh nhớ lại. “Một nửa thời gian của mấy ông kiểm thử là đì hỏi mấy ông lập trình xem chương trình có những tính năng nào để mà kiểm thử. Đến khi kiểm thử lại thấy là các chức năng mà mấy ông lập trình đã viết lại hoàn toàn không phải là cái mà người dùng mong muốn.”

“Tôi thật sự bức bối khi bà quản lý bộ phận đề nghị xây dựng Pharm-Simulator đã ký trên bản yêu cầu mà chẳng hề đọc qua xem nó thế nào”, Dave thêm vào. “Cho đến khi chúng ta nhận được cả dòng thác các đề xuất và thay đổi yêu cầu từ những người thuộc bộ phận ấy. Chả có gì ngạc nhiên khi dự án chậm đến 4 tháng và chi phí thì hầu như tăng gấp đôi so với ngân sách. Nếu điều đó xảy ra lần nữa, chắc tôi điên mất!”

Ramesh đề nghị. “Hay là chúng ta lập một danh sách các vấn đề mà ta đã gặp trong Simulator từ đó ta để tránh được các vấn đề tương tự có thể có trong dự án

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Chemical Tracking System. Tôi đã đọc một bài báo về quản lý rủi ro phần mềm, họ khuyên chúng ta nên chỉ ra các rủi ro và phác thảo kế hoạch phòng ngừa chúng."

"Tôi không biết về cái đó," Dave nói. "Chúng ta đã học được rất nhiều từ Simulator và chúng ta đều không muốn các vấn đề tương tự như vậy từ dự án này nữa. Dự án này không lớn đến mức cần phải quản lý rủi ro. Nếu chúng ta viết ra những gì có thể gây rắc rối cho Chemical Tracking System thì điều đó có vẻ như là tôi không biết làm thế nào để thực hiện dự án một cách thành công. Tôi không muốn bắt cứ những suy nghĩ trái chiều nào đối với dự án này. Chúng ta cần phải lập kế hoạch cho dự án thành công!"

Khi Dave nói ra đè xuất cuối cùng thì điều đó đã chứng tỏ rằng các kỹ sư phần mềm là những người lạc quan đáng khâm phục. Chúng ta thường hy vọng dự án tối sẽ được tiến hành suôn sẻ mặc dù chúng ta đã gặp vô số rắc rối khi thực hiện các dự án trước. Thực tế là có hàng tá các rắc rối ngăn cản dự án được tiến hành đúng như kế hoạch đã định. Vì vậy một lời khuyên hữu ích đối với các nhà quản lý là hãy định danh các rủi ro có thể có và hãy làm thế nào để phòng tránh chúng hơn là cứ cố ra vẻ lạc quan như vậy, trước hết là hãy bắt đầu với các rủi ro từ yêu cầu.

Một rủi ro là một tác động vào tiến trình dự án và có thể gây ra một số cản trở hoặc thậm chí làm thất bại dự án. Quản lý rủi ro - một best practice của công nghiệp phần mềm – là một quy trình định danh, đánh giá, kiểm soát rủi ro trước khi chúng có thể gây phiền phức cho dự án. Nếu một cái gì đó không mong muốn nhưng đã xảy ra và tác động xấu vào dự án thì cái đó không phải là rủi ro, đó là một hậu quả (issue) của rủi ro. Hãy giải quyết các vấn đề và hậu quả thông qua quy trình giám sát dự án.

Khó có thể dự đoán được độ chắc chắn xảy ra hay không xảy ra với mức độ nào của các rủi ro, nhưng quản lý rủi ro giúp bạn thực hiện các hành động để tối thiểu hóa khả năng xảy ra rủi ro. Quản lý rủi ro nghĩa là giải quyết các vấn đề tiềm tàng trước khi nó xảy ra và gây hậu quả xấu, hoặc gây ra khủng hoảng, nhằm tăng cơ may thành công của dự án. Các rủi ro nằm bên ngoài tầm kiểm soát của dự án cần được chỉ mặt, đặt tên ở mức quản lý thích hợp.

Do các yêu cầu đóng vai trò trung tâm trong các dự án phần mềm nên các nhà quản lý dự án cần định danh các rủi ro liên quan đến yêu cầu ngay từ sớm và có các kiểm soát thích hợp. Các rủi ro điển hình liên quan đến yêu cầu bao gồm không hiểu yêu cầu, không bao quát hết người dùng, sự không chắc chắn của phạm vi và mục tiêu của dự án, sự thay đổi liên tục các yêu cầu của dự án. Các nhà quản lý dự án chỉ có thể kiểm soát rủi ro yêu cầu thông qua sự hợp tác với khách hàng hoặc các đại diện của họ, ví dụ các nhân viên marketing. Các rủi ro yêu cầu cần được ghi thành tài liệu và lập kế hoạch giảm bớt hoặc loại bỏ.

Chỉ nhận biết rủi ro thì không làm mất rủi ro, vì vậy Chương này giới thiệu một số kỹ thuật quản lý rủi ro phần mềm. Một số yếu tố rủi ro có thể tăng nhanh cùng quá trình phát triển dự án và điều đó sẽ được mô tả trong phần sau của Chương này. Hãy sử dụng thông tin này để khởi động tiến trình tấn công các rủi ro yêu cầu trước khi chúng tấn công bạn.

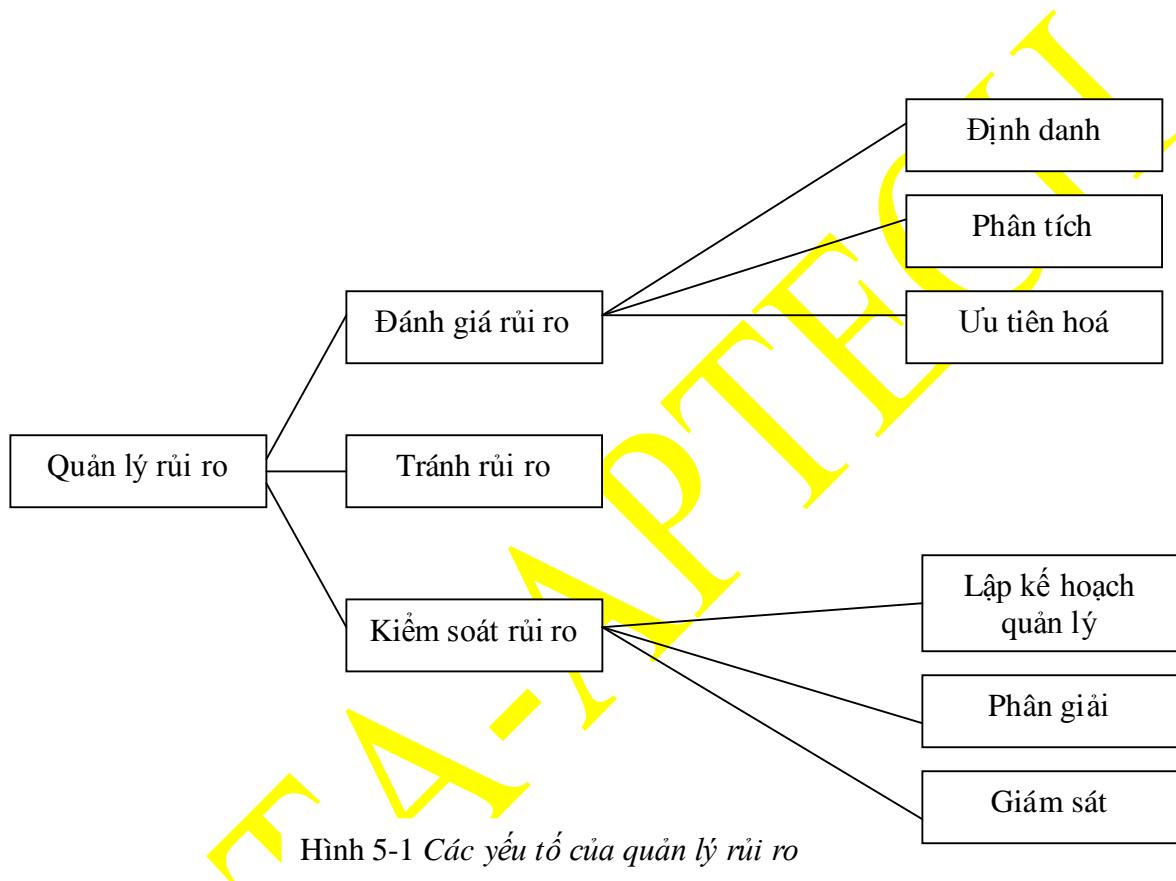
I. CƠ SỞ CỦA QUẢN LÝ RỦI RO PHẦN MỀM (FUNDAMENTALS OF SOFTWARE RISK MANAGEMENT)

Dự án phải đối mặt với nhiều loại rủi ro, chưa kể các rủi ro liên quan đến phạm vi dự án. Phải phụ thuộc vào một đối tác bên ngoài là một rủi ro hay gấp, ví dụ một nhà thầu hoặc một dự án khác sản xuất ra các components được sử dụng lại. Quản lý dự án hay phải đối mặt với các rủi ro này sinh từ sự kém chính xác của các ước lượng trước khi tiến hành dự án. Các rủi ro liên quan đến công nghệ gây ra các nguy cơ to lớn khi phát triển dự án. Sự thiếu hiểu biết về các nguồn rủi ro cũng là nguồn gốc của rủi ro... Quản lý rủi ro giống như việc bạn đi trên một con tàu và từng lúc một bạn nhìn xa về phía chân trời để xem có núi băng nào không, nếu có thì kịp thời tránh va chạm với nó hoặc va chạm nhẹ nhất có thể, điều này quan trọng hơn là khả năng hành động nhanh với một niềm tin to lớn rằng tàu của bạn sẽ không thể bị chìm. Cũng như các quy trình khác, hãy cân chỉnh các hoạt động quản lý rủi ro theo kích thước dự án của bạn. Các dự án nhỏ có thể chỉ cần lập một danh sách quản lý rủi ro đơn giản, còn các dự án lớn thì cần phải lập kế hoạch quản lý rủi ro một cách chính thức.

1. CÁC YẾU TỐ CỦA QUẢN LÝ RỦI RO (ELEMENTS OF RISK MANAGEMENT)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Quản lý rủi ro là ứng dụng các công cụ và thủ tục thích hợp để kiềm chế rủi ro dự án trong một giới hạn có thể chấp nhận. Quản lý rủi ro đề xuất một cách tiếp cận tiêu chuẩn để định danh và tài liệu hóa các yếu tố rủi ro, đánh giá độ nghiêm trọng tiềm tàng của nó, đề xuất các chiến lược để giảm bớt rủi ro đó (Williams, Walker, and Dorofee 1997). Quản lý rủi ro bao gồm các hoạt động sau trong Hình 5-1.



Đánh giá rủi ro (*risk assessment*) là quy trình khảo sát một dự án để xác định các vùng rủi ro tiềm tàng. Trước hết là *định danh rủi ro* (*risk identification*) thông qua một danh sách các rủi ro thường gặp trong một dự án phần mềm và sẽ được mô tả sau trong chương này (Carr et al. 1993, McConnell 1996). Tiếp theo là *phân tích rủi ro* (*risk analysis*), nghĩa là bạn dự đoán tất cả các hậu quả (consequences) của các rủi ro đó. Sau cùng là *ưu tiên hóa rủi ro* (*risk prioritization*), nghĩa là bạn sẽ tập trung xử lý các rủi ro nghiêm trọng nhất thông qua *khả năng phá hủy tiềm tàng của rủi ro* (*potential risk exposure*). Khả năng phá hủy tiềm tàng của rủi ro là một hàm của 2 biến: thứ nhất là khả năng gánh chịu mất mát do rủi ro, thứ hai là độ lớn của các mất mát đó.

Tránh rủi ro (*risk avoidance*) là một cách để giải quyết rủi ro: không thực hiện những gì là rủi ro (don't do the risky thing). Bạn có thể tránh rủi ro bằng cách không đảm trách các dự án rủi ro, bằng cách chỉ sử dụng các công nghệ đã được thực tế chứng minh thay vì các công nghệ mới xuất hiện, bằng cách loại trừ các tính năng sẽ đặc biệt gây khó khăn cho bạn sau này.

Nhưng thông thường bạn sẽ phải thực hiện các hoạt động *kiểm soát rủi ro* (*risk control*) để quản lý các rủi ro đã được xếp thứ tự ưu tiên. *Lập kế hoạch quản lý rủi ro* (*risk management planning*) là đưa ra một kế hoạch giải quyết mỗi rủi ro nghiêm trọng như những *cách tiếp cận giảm bớt rủi ro* (*risk mitigation approaches*), *kế hoạch liên tục nghiệp vụ* (*contingency plans*). Sau khi đã có các kế hoạch tránh rủi ro, hãy triển khai thực hiện chúng, hoạt động đó là *phân giải rủi ro* (*risk resolution*). Cuối cùng hãy giám sát tiến độ phân giải mỗi rủi ro thông qua hoạt động *giám sát rủi ro* (*risk monitoring*).

2. TÀI LIỆU HOÁ RỦI RO DỰ ÁN (DOCUMENTING PROJECT RISKS)

Bạn không chỉ đơn giản nhận biết các rủi ro đối với dự án mà còn cần phải tài liệu hoá và quản lý chúng sao cho bạn có thể thông báo về các hậu quả và trạng thái của chúng tới tất cả các stakeholders trong toàn bộ tiến trình dự án. Hình 5-2 là một mẫu tài liệu hoá một yếu tố rủi ro (*risk item*).

MẪU GIÁM SÁT YẾU TỐ RỦI RO
(RISK ITEM TRACKING TEMPLATE)

ID: <số hiệu của rủi ro>

Ngày phát hiện: <ngày rủi ro được định danh>

Ngày xử lý: <ngày rủi ro được đưa vào quản lý>

Mô tả: <mô tả rủi ro theo mẫu “điều kiện - hậu quả”>

Xác suất: <kết quả rủi ro trở thành hiện thực>

Ảnh hưởng: <sự phá hủy tiềm tàng nếu như rủi ro trở thành hiện thực>

Kế hoạch giảm bớt rủi ro: <một trong nhiều cách tiếp cận để kiểm soát, tránh, tối thiểu hóa, hoặc giảm bớt rủi ro>

Chịu trách nhiệm: <cá nhân chịu trách nhiệm phân giải rủi ro>

Ngày hoàn thành: <ngày dự kiến kế hoạch giảm bớt rủi ro được hoàn thành>

Hình 5-2 Mẫu giám sát mỗi rủi ro

Bạn có thể lưu trữ bảng này dưới dạng một bảng tính để dễ sắp xếp các rủi ro. Lưu giữ bảng này như tài liệu duy nhất về quản lý rủi ro trong toàn bộ dự án.

Dùng cách mô tả *điều kiện - hậu quả* (*condition – consequense*) khi bạn xây dựng tài liệu về các rủi ro. Nghĩa là, mô tả điều kiện mà bạn quan tâm, tiếp theo là hậu quả từ điều kiện đó – rủi ro đã trở thành một vấn đề thật sự. Thông thường, người ta chỉ mô tả các trạng thái rủi ro (risk state) như là các điều kiện (“khách hàng không thỏa thuận về yêu cầu sản phẩm”) hoặc chỉ nói về hậu quả (“chúng ta chỉ có thể đáp ứng một trong bốn khách hàng chính”). Hãy ghép các câu này thành dạng điều kiện - hậu quả: “khách hàng không thỏa thuận về các yêu cầu sản phẩm, vậy chúng ta chỉ có thể đáp ứng một trong bốn khách hàng chính.” Một điều kiện có thể sinh ra một số hậu quả, hoặc một số điều kiện có thể sinh ra một hậu quả.

Đừng cố gắng lượng hóa các rủi ro quá chính xác. Mục đích của bạn chỉ là phân biệt và xếp hạng các rủi ro theo nguy cơ của chúng mà thôi. Bạn có thể làm đơn giản hơn bằng cách ước lượng cả xác suất và ảnh hưởng theo 3 cấp *low*, *medium*, *high*.

Sử dụng Kế hoạch giảm bớt rủi ro để xác định các hành động cần thiết nhằm kiểm soát rủi ro. Một số kế hoạch chú trọng giảm xác suất xảy ra rủi ro, số khác lại tập trung giảm bớt ảnh hưởng của rủi ro. Hãy cân nhắc chi phí khi lập kế hoạch, không cần phải tiêu 20.000 USD để lập kế hoạch cho một rủi ro đáng giá 10.000 USD. Gán mỗi kế hoạch cho một người chịu trách nhiệm và định ngày hoàn thành. Các rủi ro dài hạn hoặc phức tạp cần nhiều bước tiến hành.

Hình 5-3 minh họa một rủi ro mà nhóm dự án Chemical Tracking System đã thảo luận ở đầu Chương này.

VÍ DỤ VỀ MẪU GIÁM SÁT MỘT RỦI RO CỦA CHEMICAL TRACKING SYSTEM	
ID: 1	
Ngày phát hiện: 04/05/1999	
Ngày xử lý: mở	
Mô tả: người dùng không tham gia đầy đủ ở mức cần thiết khi suy luận yêu cầu, từ đó dẫn tới có thể phải làm lại giao diện người dùng sau khi kiểm thử beta.	
Xác suất: 0,6	
Ảnh hưởng: 7	
Kế hoạch giảm bớt rủi ro:	
	<ol style="list-style-type: none">Thu thập các yêu cầu dễ nắm bắt và khả dụng ngay từ sớm trong giai đoạn 1.Tổ chức các phiên JAD với những người trợ giúp sản phẩm (product champions) để phát triển yêu cầu.Phát triển một nguyên mẫu giao diện người dùng tạm thời (throwaway user interface prototype) của các chức năng chính (core functionality) với sự giúp đỡ của những người trợ giúp sản phẩm và các chuyên gia tư vấn. Đề nghị người trợ giúp sản phẩm và những người dùng khác đánh giá nguyên mẫu.
Chịu trách nhiệm: Helen	
Ngày hoàn thành: Hoàn thành phiên JAD ngày 16/6/1999	
Hình 5-2 Mẫu giám sát mỗi rủi ro	

Nhóm ước lượng xác suất và ảnh hưởng trên cơ sở các kinh nghiệm đã có từ các dự án trước. Hai cách tiếp cận giảm rủi ro đều giảm xác suất xảy ra rủi ro bằng cách lôi kéo người dùng tham gia vào quy trình yêu cầu. Cách thứ ba là làm nguyên mẫu, cách này sẽ làm giảm bớt ảnh hưởng tiềm tàng của rủi ro bằng cách làm lại sớm giao diện người dùng.

3. LẬP KẾ HOẠCH QUẢN LÝ RỦI RO (PLANNING FOR RISK MANAGEMENT)

Một danh sách các rủi ro thì không phải là một kế hoạch quản lý rủi ro. Với một dự án nhỏ, bạn có thể bao hàm kế hoạch kiểm soát rủi ro vào kế hoạch tổng thể của dự án. Nhưng với một dự án lớn thì bạn phải có một kế hoạch quản lý rủi ro riêng. Kế hoạch này cần chỉ rõ vai trò và trách nhiệm của các hoạt động quản lý rủi ro.

Thông thường, các nhóm dự án lập kế hoạch hoạt động của họ nhưng họ lại thất bại trong việc sử dụng kế hoạch như là một hướng dẫn cách thực hiện dự án và kiểm soát các thay đổi. Đừng làm như thế với kế hoạch quản lý rủi ro!

II. CÁC RỦI RO LIÊN QUAN ĐẾN YÊU CẦU (REQUIREMENTS – RELATED RISKS)

Các yếu tố rủi ro mô tả trong các trang sau được tổ chức theo các giai đoạn của quy trình công nghệ yêu cầu: suy luận, phân tích, đặc tả, kiểm tra và quản lý (elicitation, analysis, specification, verification, management). Các kỹ thuật được đề xuất có thể giảm bớt xác suất biến rủi ro thành vấn đề, hoặc giảm bớt ảnh hưởng đến dự án nếu nó xảy ra.

1. SUY LUẬN YÊU CẦU (REQUIREMENTS ELICITATION)

Tầm nhìn và phạm vi của sản phẩm (Product vision and scope)

Vấn đề đối với phạm vi dự án là các thành viên không có một hiểu biết chung, sáng rõ về mục đích của sản phẩm là gì. Ngay từ sớm khi bắt đầu dự án, bạn đã phải viết một tài liệu về tầm nhìn và phạm vi (vision and scope) chứa đựng các yêu cầu kinh doanh (business requirements), sử dụng tài liệu này làm cơ sở cho các quyết định liên quan đến sửa đổi và bổ sung các yêu cầu mới.

Thời gian dành cho phát triển yêu cầu (Time spent on requirements development)

Lịch biểu chẽ thường gây áp lực cho các nhà quản lý khiến họ che đậy đi các yêu cầu do họ tin rằng nếu các lập trình viên không bắt đầu làm việc ngay thì họ sẽ không đáp ứng được lịch biểu. Các thông số của mỗi dự án biến đổi theo kích thước và miền ứng dụng của nó (hệ thống thông tin quản lý, phần mềm hệ thống, phần mềm quân sự...), nhưng người ta thấy thường các dự án tiêu tốn khoảng 15% nguồn lực (thời gian, con người, tiền bạc) cho các hoạt động phát triển yêu cầu (Rubin 1999). Hãy ghi chép lại thời gian mà bạn đã sử dụng cho các dự án cụ thể khác nhau để dùng nó làm tham số hiệu chỉnh các dự án tương lai của bạn.

Tính đầy đủ và tính đúng đắn của đặc tả yêu cầu (Completeness and correctness of requirements specification)

Để đảm bảo các yêu cầu mô tả đúng cái khách hàng cần, bạn hãy ứng dụng kỹ thuật use-case để suy luận các yêu cầu bằng cách tập trung vào các hành động thực hiện công việc (tasks) của người dùng. Hãy nghĩ ra các kịch bản sử dụng cụ thể, hãy viết các test cases từ các yêu cầu, hãy tạo ra các nguyên mẫu để các yêu cầu trở nên dễ nhận biết đối với người dùng và hãy suy luận dựa vào các phản hồi của họ. Hãy tranh thủ tình cảm của đại diện các khách hàng, để thanh tra (inspect) các đặc tả yêu cầu và các mô hình phân tích.

Yêu cầu cho sản phẩm mang tính sáng tạo cao (Requirements for hight innovative products)

Rất dễ dàng nhận định sai về phản ứng của thị trường đối với sản phẩm khi mới được tung ra. Hãy tập trung nhiều cho các hoạt động nghiên cứu thị trường, xây dựng các nguyên mẫu, sử dụng các nhóm định hướng khách hàng (customer focus groups) để đạt được ngay từ sớm và thường xuyên về viễn cảnh của sản phẩm mang tính sáng tạo (innovative product visions).

Định nghĩa các yêu cầu phi chức năng (Define nonfunctional requirements)

Do sự nhấn mạnh tự nhiên của nhóm phát triển vào chức năng của sản phẩm nên họ rất dễ xao lảng các yêu cầu phi chức năng. Hãy tham vấn khách hàng về các đặc tính chất lượng như hiệu năng (performance), khả năng sử dụng (usability), tính toàn vẹn (integrity) và độ tin cậy (reliability). Tài liệu hóa các yêu cầu phi chức năng này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

năng và các tiêu chuẩn chấp nhận (acceptance criteria) càng chính xác càng tốt trong SRS.

Thỏa thuận của khách hàng về các yêu cầu sản phẩm (Customer agreement on product requirements)

Nếu có những khách hàng không thỏa thuận về những gì mà bạn cần xây dựng thì sẽ có ai đó không hài lòng khi nhận được sản phẩm. Vậy hãy xác định ai là khách hàng chính, hãy sử dụng những người trợ giúp sản phẩm để đảm bảo chắc chắn bạn có được ý kiến đầy đủ của tất cả khách hàng và những người liên quan đến sản phẩm. Hãy chắc chắn bạn trao đổi đúng người khi cần ra các quyết định liên quan đến yêu cầu.

Các yêu cầu không xác định (Unstated requirements)

Khách hàng sẽ ghi nhớ trong đầu họ các mong muốn mà họ không nói ra và vì vậy không được ghi thành tài liệu. Hãy xác định và ghi nhớ bất cứ cái gì khách hàng có thể đặt ra. Sử dụng các câu hỏi mở để khuyến khích khách hàng chia sẻ nhiều hơn ý nghĩ, ý tưởng và các quan tâm của họ, bạn sẽ thu nhận được nhiều hơn những gì bạn nghe thấy.

Các sản phẩm đang sử dụng được coi như là ranh giới yêu cầu (Existing product used as requirements baseline)

Phát triển yêu cầu không được coi trọng đối với các dự án tái xây dựng quy trình (reengineering projects) tức là các dự án nâng cấp. Các nhà phát triển thường được bảo là hãy sử dụng hệ thống đã có như là nguồn yêu cầu, “hãy loại trừ các lỗi đã biết và thêm các tính năng mới”. Nhà phát triển sẽ lướm lặt các yêu cầu mới thông qua tiến trình tái xây dựng sản phẩm hiện có. Tuy nhiên, tiến trình tái xây dựng là một cách thức làm việc không hiệu quả và không đầy đủ để phát hiện các yêu cầu, và sẽ không ai ngạc nhiên nếu hệ thống mới có những hạn chế y như hệ thống cũ. Hãy tài liệu hóa các yêu cầu mà bạn phát hiện thông qua tiến trình tái cơ cấu và đề nghị khách hàng soát xét các yêu cầu đó để đảm bảo chúng vẫn còn có ích.

Giải pháp được đề xuất từ nhu cầu (Solution presented as needs)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Các giải pháp do người dùng đề xuất có thể che lấp đi các nhu cầu thật sự của người dùng và khiến cho việc tự động hóa các quy trình nghiệp vụ (business process) không hiệu quả, hoặc gây áp lực khiến nhà phát triển đưa ra các thiết kế kém cỏi. Nhà phân tích cần phải “chui sâu” để hiểu được thật sự nhu cầu của ai sau các giải pháp mà khách hàng đề xuất là gì. Nhìn chung, khách hàng chỉ nên là người đề ra nhu cầu và chính người phát triển mới là người đề xuất giải pháp.

2. PHÂN TÍCH YÊU CẦU (REQUIREMENTS ANALYSIS)

Ưu tiên hoá yêu cầu (Requirements prioritization)

Đảm bảo mỗi yêu cầu, mỗi tính năng, mỗi use-case đều được ưu tiên hoá và được phân bổ vào một phiên bản cụ thể của sản phẩm hoặc được phân bổ vào một thời điểm thi công cụ thể (implementation stage). Đánh giá sự ưu tiên của mỗi yêu cầu mới trong sự tương quan với các yêu cầu đã có trước đây đòi hỏi bạn phải có các quyết định đánh đổi thông minh.

Các tính năng khó về mặt kỹ thuật (Technically difficult features)

Đánh giá tính khả thi của mỗi yêu cầu không chỉ diễn ra khi lập kế hoạch mà còn kéo dài cho đến tận khi thi công, do có những tính năng khó thực thi về mặt kỹ thuật, vì vậy bạn phải bám sát diễn biến thi công của dự án để từ đó ra các quyết định thích hợp đối với các tính năng này, các quyết định càng được đưa ra sớm thì càng tốt.

Các công nghệ, phương pháp, ngôn ngữ, công cụ, hoặc thiết bị phần cứng không quen thuộc (Unfamiliar technologies, methods, languages, tools, or hardware)

Đừng đánh giá quá cao khả năng nắm bắt nhanh của nhóm dự án đối với các công nghệ không quen thuộc, hãy xác định mức rủi ro cao khi bắt đầu một công nghệ mới.

3. ĐẶC TẢ YÊU CẦU (REQUIREMENTS SPECIFICATION)

Hiểu yêu cầu (Understanding requirements)

Các hiểu biết khác nhau về yêu cầu giữa nhà phát triển và khách hàng có thể dẫn đến những khoảng cách lớn về kỳ vọng, nghĩa là sản phẩm chuyển giao không đáp ứng được kỳ vọng của khách hàng. Các cuộc thanh tra chính thức tài liệu yêu cầu Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

của các nhóm bao gồm nhà phát triển, kiểm thử viên, khách hàng, có thể làm giảm bớt rủi ro. Các nhà phân tích được đào tạo và có kinh nghiệm về yêu cầu sẽ đặt các câu hỏi đúng cho khách hàng và viết các đặc tả tốt hơn. Mô hình và các nguyên mẫu thể hiện yêu cầu từ nhiều góc độ khác nhau sẽ để lộ ra các điểm mờ và các yêu cầu nhập nhằng.

Gây áp lực thời gian để thực hiện các TBDs (Time pressure to proceed despite TBDs)

Các yêu cầu chưa được làm rõ thường được đánh dấu TBD (to be determined), sẽ rất rủi ro nếu tiến hành thi công trên các yêu cầu đó. Hãy giao cho các cá nhân cụ thể trách nhiệm phân giải mỗi TBD, phân giải như thế nào và thời hạn phân giải.

Các thuật ngữ nhập nhằng (Ambiguous terminology)

Lập một bảng thuật ngữ (glossary) và từ điển dữ liệu để định nghĩa tất cả các thuật ngữ nghiệp vụ và kỹ thuật (business and technical terms) có thể được diễn giải khác nhau bởi những người đọc khác nhau. Trên thực tế, một thuật ngữ có thể có nghĩa chung, nghĩa kỹ thuật, hoặc nghĩa được xác định trong một phạm vi hẹp. Soát xét SRS có thể giúp những người tham gia đạt được một hiểu biết chung về các thuật ngữ chính.

Thiết kế được bao hàm trong yêu cầu (Design included in requirements)

Cách tiếp cận thiết kế được bao hàm trong SRS có thể đặt ra các ràng buộc không cần thiết về các lựa chọn có thể có đối với các nhà phát triển và có thể ngăn cản việc sáng tạo ra các thiết kế tối ưu. Soát xét các yêu cầu để chắc chắn SRS chỉ nhấn mạnh cái cần làm thay vì làm cái đó như thế nào.

4. KIỂM TRA YÊU CẦU (REQUIREMENTS VERIFICATION)

Các yêu cầu không được kiểm tra (Unverified requirements)

Nếu bạn kiểm tra chất lượng và tính đúng đắn của các yêu cầu trước khi bắt đầu thi công thì việc lập kế hoạch kiểm thử dựa trên yêu cầu, lập các nguyên mẫu sẽ giúp bạn giảm bớt các công việc phải làm lại trong dự án. Hãy lôi kéo khách hàng tham gia các cuộc kiểm tra yêu cầu. Thực hiện các soát xét tăng dần, phi chính thức, ngoài các soát xét chính thức để phát hiện càng sớm càng tốt các vấn đề trong yêu cầu.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Sự thành thạo khi thanh tra yêu cầu (Inspection proficiency)

Nếu những người thanh tra yêu cầu không biết làm thế nào để thanh tra một cách đúng đắn các tài liệu yêu cầu và làm thế nào để góp phần vào các cuộc thanh tra hiệu quả thì các lỗi nghiêm trọng (serious defects) có thể không được phát hiện. Hãy đào tạo tất cả những người tham gia thanh tra tài liệu yêu cầu. Mời một thanh tra viên có kinh nghiệm làm việc này.

5. QUẢN LÝ YÊU CẦU (REQUIREMENTS MANAGEMENT)

Thay đổi yêu cầu (Changing requirements)

Các vấn đề phát sinh đối với phạm vi của sản phẩm thì có thể sử dụng tài liệu tầm nhìn và phạm vi (vision and scope) như là một chuẩn để chấp nhận thay đổi. Một quy trình suy luận yêu cầu mang tính hợp tác (collaborative) với sự mở rộng các thành viên tham gia có thể giảm bớt đến một nửa các vấn đề về yêu cầu (Jones 1996a). Các hướng dẫn thực hành kiểm soát chất lượng (quality control practices) có thể phát hiện ngay từ sớm các lỗi và làm giảm bớt số lượng các sửa đổi phải thực hiện sau này.

Quy trình thay đổi yêu cầu (Requirements change process)

Các rủi ro sẽ xuất hiện khi các đề xuất thay đổi xuất hiện trong SRS mà chẳng tuân theo một quy tắc nào cả. Bạn cần có một quy trình kiểm soát tất cả các thay đổi được đề xuất, quy trình như vậy bao gồm việc phân tích ảnh hưởng của các thay đổi được đề xuất, một ban kiểm soát thay đổi ra quyết định chấp thuận hay từ chối thay đổi, một công cụ hỗ trợ quản lý thay đổi được sử dụng để tăng hiệu quả làm việc.

Các yêu cầu không được thi công (Unimplemented requirements)

Má trận lằn vết yêu cầu giúp bạn không bỏ sót bất cứ yêu cầu nào khi thiết kế, thi công, kiểm thử. Nó cũng giúp bạn đảm bảo một yêu cầu sẽ không được thực hiện bởi nhiều nhà phát triển do sự thiếu thông tin đầy đủ trên dự án.

Mở rộng phạm vi dự án (Expanding project scope)

Nếu các yêu cầu được định nghĩa không đúng ngay từ đầu, thì phạm vi của dự án sẽ phải được mở rộng sau này. Các đặc tả mơ hồ về sản phẩm có thể tiêu tốn nhiều Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

nỗ lực hơn là tiên liệu của bạn. Các nguồn lực của dự án được phân bổ theo các yêu cầu ban đầu có thể phải điều chỉnh cho đúng với phạm vi đích thực của dự án mà bạn đã biết rõ hơn. Để giảm bớt các rủi ro này, hãy lập kế hoạch dự án thành từng giai đoạn và tăng trưởng dần dần. Thực hiện chức năng cốt lõi trước, các chức năng khác sẽ được thực hiện dần trong các giai đoạn sau của dự án.

III. QUẢN LÝ RỦI RO CẦN THIẾT CHO BẠN (RISK MANAGEMENT IS YOUR FRIEND)

Một nhà quản lý dự án có thể sử dụng các phương pháp quản lý rủi ro để tăng khả năng nhận thức về các nguyên nhân khiến cho dự án trở nên tồi tệ. Các nhà quản lý dự án nên cân nhắc để chọn được những người thích hợp thực hiện công việc suy luận yêu cầu. Các nhà quản lý có kinh nghiệm sẽ chỉ thực hiện dự án khi đã có danh sách các rủi ro, ước lượng khả năng xảy ra của mỗi rủi ro và ảnh hưởng của nó tới toàn dự án.

Các bước tiếp theo

- Định danh một số rủi ro liên quan đến yêu cầu mà bạn đang phải đối mặt trong dự án hiện tại. Đừng xác định các vấn đề hiện tại như là rủi ro, chỉ chú tâm vào những gì chưa xảy ra thôi. Tài liệu hoá các rủi ro theo khuôn mẫu điều kiện - hậu quả, hãy sử dụng template trong Hình 5-2. Hãy đề xuất ít nhất một cách tiếp cận giảm bớt rủi ro có thể cho mỗi rủi ro.
- Hãy tổ chức một phiên tập kích não (brainstorming) về rủi ro với sự tham gia của các stakeholders đại diện cho nhà phát triển, marketing, khách hàng, và cấp quản lý. Hãy xác định càng nhiều càng tốt các rủi ro mà bạn có thể gặp. Đánh giá mỗi rủi ro về xác suất xuất hiện và ảnh hưởng tương đối của nó. Sắp xếp các rủi ro thành danh sách giảm dần theo độ xác suất rủi ro để từ đó thấy được 5 rủi ro lớn nhất. Hãy gán mỗi rủi ro cho một cá nhân để thực hiện các hành động giảm bớt rủi ro.

PHẦN II PHÁT TRIỂN YÊU CẦU PHẦN MỀM

SATA-APTECH

CHƯƠNG 6

THIẾT LẬP TÂM NHÌN VÀ PHẠM VI CỦA DỰ ÁN

Bạn đồng nghiệp Karen của tôi đã giới thiệu thành công phương pháp thanh tra chính thức (formal inspections) tài liệu yêu cầu phần mềm trong công ty của cô. Cô nhận thấy phần nhiều các vấn đề này sinh trong các phiên thanh tra là xuất phát từ cách hiểu sai về phạm vi dự án. Những người tham gia thanh tra thường có các hiểu biết khác nhau về phạm vi hàm ý (intended scope) của dự án, và không phải bao giờ họ cũng chia sẻ cùng nhau một tầm nhìn chung đối với các mục tiêu của dự án. Hệ quả là họ sẽ gặp nhiều khó khăn khi thỏa thuận với nhau về các yêu cầu chức năng (functional requirements) của hệ thống sẽ được mô tả trong SRS.

Như đã nói trong Chương 1, yêu cầu kinh doanh (business requirements) là thể hiện mức cao nhất của sự trừu tượng hoá trong chuỗi yêu cầu: chúng định nghĩa tầm nhìn và phạm vi (vision and scope) của dự án. Yêu cầu người dùng (user requirements) và yêu cầu chức năng (functional requirements) cần phải song song với bối cảnh (context) và mục tiêu đã được yêu cầu kinh doanh (business requirements) đặt ra. Các yêu cầu mà không nhằm đạt được các mục tiêu đó thì không được đưa vào bản mô tả yêu cầu. Dự án cũng phải bao gồm các yêu cầu kinh doanh (business requirements) không liên quan trực tiếp tới phần mềm như mua sắm phần cứng, cài đặt sản phẩm, bảo trì, quảng cáo, nhưng ở đây ta chỉ lưu tâm đến các yêu cầu kinh doanh cho phần mềm cần làm mà thôi.

Một dự án mà thiếu một định hướng được thiết lập và truyền thông một cách rõ ràng thì sẽ tăng nguy cơ thất bại nên rất nhiều, những người tham gia dự án có thể mang vào đó nhiều dự định khác nhau, nhiều ưu tiên khác nhau. Yêu cầu sẽ không thể bền vững nếu các stakeholders không chia sẻ cùng nhau một hiểu biết chung về các nhu cầu kinh doanh (business needs) mà sản phẩm cần đáp ứng và lợi ích mà sản phẩm mang lại. Một tầm nhìn và phạm vi (vision and scope) sáng sủa là đặc biệt quan trọng khi dự án được phát triển ở nhiều địa điểm khác nhau, do sự đa dạng về địa điểm khiến cho giao tiếp giữa các thành viên trở nên khó khăn hơn.

Một dấu hiệu thể hiện các yêu cầu kinh doanh (business requirements) không được định nghĩa đầy đủ là một số tính năng được đưa vào từ đầu, sau đó bị xoá đi rồi lại được đưa vào. Các vấn đề thuộc tầm nhìn và phạm vi (vision and scope) của dự án cần được phân giải rõ trước khi các yêu cầu chức năng (functional requirements) chi tiết được đặc tả đầy đủ. Một tài liệu tầm nhìn và phạm vi (vision and scope) tốt sẽ cung cấp các tham chiếu cần thiết cho việc thêm, xoá bỏ, chỉnh sửa các yêu cầu trong tiến trình phát triển của dự án.

I. ĐỊNH NGHĨA TẦM NHÌN THÔNG QUA YÊU CẦU KINH DOANH (DEFINE VISION THROUGH BUSINESS REQUIREMENTS)

Tầm nhìn dự án định hướng cho các thành viên dự án về một hướng đi rõ ràng. Phạm vi dự án vạch ra ranh giới giữa cái bên trong và cái bên ngoài dự án. Các yêu cầu kinh doanh (business requirements) được mô tả trong tài liệu tầm nhìn và phạm vi (vision and scope) của dự án, chúng cần thiết để dự tính nguồn vốn cho dự án. Các tổ chức xây dựng các phần mềm thương mại thường tạo ra một tài liệu yêu cầu marketing với mục đích tương tự.

Tài liệu tầm nhìn và phạm vi (vision and scope) thường được phác thảo bởi những người tài trợ vốn cho dự án hoặc bởi một ai đó có vai trò tương tự. Yêu cầu kinh doanh (business requirements) được thu thập từ những cá nhân có thể trả lời câu hỏi tại sao cần có dự án này. Các cá nhân như vậy có thể là nhà tài trợ, khách hàng hoặc nhà quản lý cao cấp của tổ chức, những người hỗ trợ sản phẩm (product champions), các thành viên của bộ phận marketing.

Các yêu cầu kinh doanh (business requirements) được tập hợp từ nhiều nguồn có thể xung đột nhau. Ví dụ, ta xét một phần mềm nhúng quản lý máy bán hàng tự động. Các mục tiêu của phần mềm đó do người phát triển thu thập được:

- Cho thuê hoặc bán kiôt cho người bán lẻ
- Bán thực phẩm, đồ uống, sách báo
- Thu hút khách hàng tới mua hàng của một thương hiệu nhất định
- Cải thiện quan hệ nhà phát triển – khách hàng

Các mối quan tâm của người bán lẻ có thể là:

- Thu được tiền từ khách hàng sử dụng máy bán hàng tự động

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- Thu hút nhiều khách hàng hơn sử dụng máy bán hàng
- Tiết kiệm được chi phí nếu thay thế máy bán hàng bằng các hoạt động bán hàng thủ công

Nhà phát triển muốn tạo ra một thiết bị bán hàng công nghệ cao và dễ sử dụng cho khách hàng; trong khi người bán lẻ thì chỉ muốn một hệ thống đơn giản, chìa khóa trao tay; còn khách hàng thì muốn hệ thống tiện lợi và nhiều tính năng. Ba cách tiếp cận vấn đề khác nhau trên sẽ dẫn đến các mục đích khác nhau, các ràng buộc khác nhau, các chi phí khác nhau và có thể khiến yêu cầu kinh doanh (business requirements) bị xung đột. Tất cả các xung đột này cần được phân giải trước khi các yêu cầu phần mềm được làm chi tiết.

Bạn cũng có thể sử dụng các yêu cầu kinh doanh (business requirements) để thiết lập thứ tự ưu tiên thực thi các use cases và các yêu cầu chức năng (functional requirements) liên quan. Ví dụ, một yêu cầu kinh doanh có thể mang lại doanh thu nhiều nhất cho máy bán hàng thì sẽ được ưu tiên thực hiện trước nhất cùng các tính năng liên quan.

Yêu cầu kinh doanh làm rõ cả tập hợp các tình huống sử dụng (use cases) mà ứng dụng sẽ cài đặt (bề rộng của ứng dụng) và độ sâu mà mỗi use-case được thực hiện. Độ sâu có thể từ cài đặt bình thường tới tự động hóa hoàn toàn. Cả chiều rộng và chiều sâu của mỗi use-case cần phải được xác định và tài liệu hoá đầy đủ.

II. TÀI LIỆU TẦM NHÌN VÀ PHẠM VI (VISION AND SCOPE DOCUMENT)

Tài liệu bao gồm một mô tả về cơ hội kinh doanh của sản phẩm, tầm nhìn và các mục tiêu của sản phẩm, báo cáo phạm vi và các giới hạn của sản phẩm, mô tả đặc tính của khách hàng (characterization of customers), các ưu tiên của dự án, mô tả các tiêu chuẩn đánh giá sự thành công của dự án. Tài liệu cần tương đối ngắn, chỉ nên từ 3 tới 8 trang, phụ thuộc chủ yếu vào bản chất và kích thước của dự án.

Hình 6-1 là một template của tài liệu này.

- | |
|---|
| 1. Yêu cầu kinh doanh (business requirements) |
| 1.1. Dẫn nhập (Background) |
| 1.2. Cơ hội kinh doanh (Business opportunities) |
| 1.3. Mục tiêu kinh doanh (Business objectives) |
| 1.4. Yêu cầu của khách hàng hoặc yêu cầu của thị trường (Customer or Market requirements) |
| 1.5. Giá trị được cung cấp cho khách hàng (Value provided to customers) |
| 1.6. Rủi ro kinh doanh (Business risks) |
| 2. Tầm nhìn của giải pháp (Vision of solution) |
| 2.1. Báo cáo tầm nhìn (Vision statement) |
| 2.2. Các tính năng chính (Major features) |
| 2.3. Các giả định và ràng buộc (Assumptions and dependencies) |
| 3. Phạm vi và các giới hạn (Scope and Limitations) |
| 3.1. Phạm vi của phiên bản đầu tiên (Scope of Initial Release) |
| 3.2. Phạm vi các phiên bản tiếp theo (Scope of Subsequent Release) |
| 3.3. Các giới hạn và loại trừ (Limitations and Exclusions) |
| 4. Bối cảnh kinh doanh (Business context) |
| 4.1. Hồ sơ khách hàng (Customer profiles) |
| 4.2. Ưu tiên của dự án (Project Priorities) |
| 5. Các yếu tố thành công (Product success factors) |

Hình 6-1 *Template cho tài liệu tầm nhìn và phạm vi (vision and scope)*

Dưới đây là giải thích template trên.

1. YÊU CẦU KINH DOANH (BUSINESS REQUIREMENTS)

Yêu cầu kinh doanh xác định những lợi ích chính mà hệ thống mới sẽ cung cấp cho khách hàng và tổ chức phát triển sản phẩm. Nội dung này khác nhau đối với những loại hình sản phẩm khác nhau như hệ thống thông tin quản lý, gói phần mềm thương mại, hệ thống chứa các phần mềm nhúng.

1.1. Dẫn nhập (Background)

Mô tả ngắn gọn nguyên nhân xây dựng sản phẩm, mô tả quá trình hoặc tình trạng dẫn tới quyết định xây dựng sản phẩm.

1.2. Cơ hội kinh doanh (Business opportunities)

Mô tả cơ hội thị trường cho sản phẩm hoặc mô tả vấn đề kinh doanh (business problem) cần được giải quyết. Mô tả thị trường mà sản phẩm sẽ được sử dụng và tính cạnh tranh của sản phẩm là gì trên thị trường nếu đó là sản phẩm thương mại, mô tả nơi sử dụng sản phẩm nếu đó là hệ thống thông tin được đặt hàng. Hãy mô tả ngắn gọn về sản phẩm tương tự đang có trên thị trường và so sánh sản phẩm đang đề xuất xây dựng với sản phẩm này. Hãy xác định các vấn đề không thể được giải quyết nếu không sản xuất sản phẩm đề xuất, hãy mô tả sự phù hợp của sản phẩm với khuynh hướng thị trường và định hướng chiến lược kinh doanh của doanh nghiệp.

1.3. Mục tiêu kinh doanh (Business objectives)

Tổng kết ngắn gọn về các lợi ích mà sản phẩm sẽ mang lại, nếu lượng hoá được các lợi ích ấy thì sẽ thuyết phục hơn. Giá trị mà sản phẩm mang lại cho khách hàng được mô tả trong mục 1.5 tài liệu này, vì vậy ở đây chỉ nói về các giá trị sản phẩm mang lại cho chính doanh nghiệp sản xuất ra nó. Các mục tiêu này cần được mô tả trong sự ước lượng về doanh thu (revenue) và tiết kiệm chi phí, phân tích tỷ lệ hoàn vốn đầu tư (ROI), ngày phát hành các phiên bản. Nếu các thông tin như trên xuất hiện ở nhiều nơi thì hãy tham chiếu tới đó từ mục này.

1.4. Yêu cầu của khách hàng hoặc yêu cầu của thị trường (Customer or Market requirements)

Mô tả nhu cầu của các khách hàng tiềm năng, gồm cả các nhu cầu không được đáp ứng bởi các sản phẩm đang tồn tại trên thị trường. Hãy giới thiệu các vấn đề mà khách hàng đang đối mặt trong hoạt động nghiệp vụ của họ, trong số các vấn đề đó thì sản phẩm sẽ giải quyết vấn đề nào và hãy đưa ra các ví dụ về việc khách hàng sử dụng sản phẩm để giải quyết vấn đề. Hãy xác định môi trường phần cứng và phần mềm mà trên đó sản phẩm hoạt động. Hãy định nghĩa ở mức ý tưởng bắt cứ yêu cầu hiệu năng hoặc yêu cầu giao diện quan trọng nào của khách hàng, nhưng tránh chỉ ra các thiết kế và thực hiện ở mức chi tiết. Hãy viết các yêu cầu thành một danh sách có đánh số sao cho sau này các yêu cầu người dùng và yêu cầu chức năng có thể lần vết ngược trở lại.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

1.5. Giá trị được cung cấp cho khách hàng (Value provided to customers)

Định nghĩa các giá trị mà khách hàng sẽ nhận được từ sản phẩm và chỉ ra sản phẩm sẽ làm hài lòng khách hàng như thế nào. Diễn giải các giá trị được mang lại cho khách hàng như sau:

- Cải thiện năng suất hoặc giảm các công việc cần làm lại
- Tiết kiệm chi phí
- Xây dựng quy trình làm việc
- Tự động hóa các công việc vẫn được làm bằng tay như trước đây
- Có khả năng thực hiện trọn vẹn các tác vụ (tasks) mới hoặc chức năng mới
- Phù hợp với các tiêu chuẩn hoặc quy định hợp lý
- Cải thiện khả năng sử dụng so với ứng dụng hiện có

1.6. Rủi ro kinh doanh (Business risks)

Tóm tắt các rủi ro kinh doanh chính (major business risks) liên quan tới việc phát triển hoặc không phát triển sản phẩm, ví dụ sự cạnh tranh trên thị trường, các vấn đề về thời hạn, sự chấp nhận của người dùng, các vấn đề về thực thi, các ảnh hưởng không mong muốn có thể có về mặt kinh doanh. Ước lượng mức độ nghiêm trọng của rủi ro, xác định bất cứ hành động giảm bớt rủi ro nào mà bạn có thể vạch ra.

2. TẦM NHÌN CỦA GIẢI PHÁP (VISION OF SOLUTION)

Mục này thiết lập một tầm nhìn dài hạn của sản phẩm, tầm nhìn này tạo bối cảnh cho việc ra quyết định trên toàn bộ quá trình phát triển sản phẩm. Tầm nhìn không bao gồm các yêu cầu chức năng (functional requirements) hoặc thông tin lập kế hoạch dự án.

2.1. Báo cáo tầm nhìn (Vision statement)

Mô tả mục đích dài hạn của sản phẩm trong đánh giá của doanh nghiệp, đồng thời cũng mô tả sản phẩm đáp ứng khách hàng như thế nào. Dưới đây là ví dụ về báo cáo tầm nhìn của Chemical Tracking System (CTS) đã được thảo luận trong chương trước:

CTS giúp các nhà khoa học gửi đề nghị mua các công-ten-nơ hoá chất đến các nhà cung cấp. Vị trí của mỗi công-ten-nơ trong công ty, số lượng hoá chất còn trong đó, quá trình từ khi nhập công-ten-nơ vào công ty và sử dụng hoá chất trong đó được giám sát bởi hệ thống. Công ty tiết kiệm được 25% chi phí mua hoá chất bằng cách khai thác tối ưu các công-ten-nơ đang có, bằng cách loại bỏ các công-ten-nơ hết hạn và bằng cách sử dụng một quy trình mua sắm hoá chất tối ưu. CTS cũng sinh các báo cáo theo yêu cầu phù hợp với các quy định của bang và liên bang về việc sử dụng, lưu trữ và hủy bỏ hoá chất.

2.2. Các tính năng chính (Major features)

Là một danh sách có đánh số các tính năng chính mà sản phẩm cung cấp cho khách hàng, nhấn mạnh các tính năng phân biệt sản phẩm với sản phẩm cạnh tranh, hoặc phân biệt sản phẩm với phiên bản trước của nó. Các yêu cầu người dùng (user requirements) và yêu cầu chức năng (functional requirements) có thể được lùn vét tới các tính năng đó.

2.3. Các giả định và ràng buộc (Assumptions and dependencies)

Ghi lại bất cứ giả định nào có thể xuất hiện trong đầu bạn về dự án khi viết tài liệu tầm nhìn và phạm vi (vision and scope). Thường thì các giả định đó chỉ được biết bởi người nghĩ ra chứ không chia sẻ cho người khác. Nếu bạn viết ra và xem xét kỹ nó cùng người khác bạn có thể có được một tập hợp các giả định cơ bản về dự án. Ví dụ, người quản lý tài trợ CTS giả định sẽ thay thế hệ thống quản lý kho hiện tại bằng CTS và CTS sẽ giao tiếp với ứng dụng mua sắm. Hãy viết ra các giả định đó để tránh bất cứ rối loạn nào trong tương lai. Cũng vậy, hãy viết các ràng buộc chính của dự án như công nghệ cụ thể được sử dụng, nhà cung cấp thứ ba, đối tác phát triển và các quan hệ kinh doanh khác.

3. PHẠM VI VÀ CÁC GIỚI HẠN (SCOPE AND LIMITATIONS)

Phạm vi của dự án định nghĩa nội dung của giải pháp được đề xuất, giới hạn xác định những gì mà giải pháp không có. Làm sáng tỏ phạm vi và giới hạn nhằm giúp các stakeholders có được các kỳ vọng thực tế đối với giải pháp. Đôi khi khách hàng đề xuất các tính năng mà để làm ra nó cần chi phí cao hoặc các tính năng không thuộc phạm vi. Các yêu cầu được đề xuất nằm ngoài phạm vi cần phải bị từ chối, trừ khi xét thấy chúng mang lại nhiều lợi ích thì phạm vi sẽ được nới rộng ra. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

để chừa đựng các yêu cầu đó (với sự thay đổi tương ứng của ngân sách, lịch biểu, thành viên nhóm làm việc). Lưu trữ các ghi chép về các yêu cầu như vậy và lý do chúng bị từ chối vì có thể chúng sẽ xuất hiện trở lại sau này.

3.1. Phạm vi của phiên bản đầu tiên (Scope of Initial Release)

Tóm tắt các tính năng chính sẽ được đưa vào trong phiên bản đầu tiên của sản phẩm. Mô tả các đặc tính chất lượng cần thiết của sản phẩm đáp ứng nhu cầu nhiều cộng đồng khách hàng khác nhau. Hãy tập trung làm trước các tính năng mang lại giá trị cao nhất, trong chi phí phát triển khả thi nhất, đạt được diện triển khai rộng nhất.

Ví dụ, trong một dự án của Scott bạn tôi, anh ta đã quyết định đưa **nghiệp vụ chuyển giao gói hàng** vào phiên bản đầu tiên. Phiên bản 1.0 không cần phải nhanh, đẹp, hoặc dễ sử dụng, cái cần nhất là **sự tin cậy**, điều này sẽ quyết định cái gì sẽ được nhóm làm. Phiên bản đầu tiên hoàn thành các mục tiêu cơ bản của hệ thống, các phiên bản tiếp theo sẽ đưa các tính năng kém quan trọng hơn, các lựa chọn, các trợ giúp dễ sử dụng.

3.2. Phạm vi các phiên bản tiếp theo (Scope of Subsequent Release)

Mô tả các tính năng được đưa vào các phiên bản tiếp theo, thời gian dự kiến.

3.3. Các giới hạn và loại trừ (Limitations and Exclusions)

Định nghĩa biên giới giữa cái bên trong và cái bên ngoài là cách để quản lý cả những vấn đề liên quan đến phạm vi lẫn các kỳ vọng của người dùng.

4. BỐI CẢNH KINH DOANH (BUSINESS CONTEXT)

Tóm tắt các vấn đề nghiệp vụ của dự án, thêm hồ sơ phân loại khách hàng chính và các ưu tiên cho dự án của cấp quản lý.

4.1. Hồ sơ khách hàng (Customer profiles)

Hồ sơ khách hàng xác định các đặc tính (characteristics) cơ bản của các loại khách hàng khác nhau, do mỗi khách hàng có những yêu cầu khác nhau đối với sản phẩm. Hồ sơ cần các thông tin sau:

- Lợi ích chính mà mỗi loại khách hàng khác nhau sẽ nhận được từ sản phẩm
- Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- Thái độ của họ đối với sản phẩm.
- Các tính năng chính mang lại lợi ích.
- Các định hướng thành công (success drivers) của mỗi loại khách hàng.
- Bất cứ ràng buộc nào đã biết đối với khách hàng cũng cần phải đưa vào.

4.2. Ưu tiên của dự án (Project Priorities)

Trước khi các ưu tiên của dự án được thiết lập, các stakeholders và những người tham gia dự án có thể tập trung chủ yếu vào các mục tiêu chung của dự án. Một cách để tiếp cận cách xếp thứ tự ưu tiên là cân nhắc 5 chiều kích của một dự án phần mềm: tính năng, chất lượng, lịch biểu, chi phí, nhân viên (Wiegers 1996a). Trong bất cứ dự án nào thì mỗi chiều kích cũng thuộc về một trong 3 phân loại sau:

- *Một yếu tố định hướng (a driver)*: một mục tiêu được ưu tiên cao nhất.
- *Một ràng buộc (a constraint)*: một yếu tố giới hạn mà nhà quản lý dự án cần phải tuân theo.
- *Một bậc tự do (a degree of freedom)*: một yếu tố mà nhà quản lý dự án có thể cân đối giữa các chiều kích khác nhau nhằm đạt được các định hướng (drivers) trong các ràng buộc đã biết.

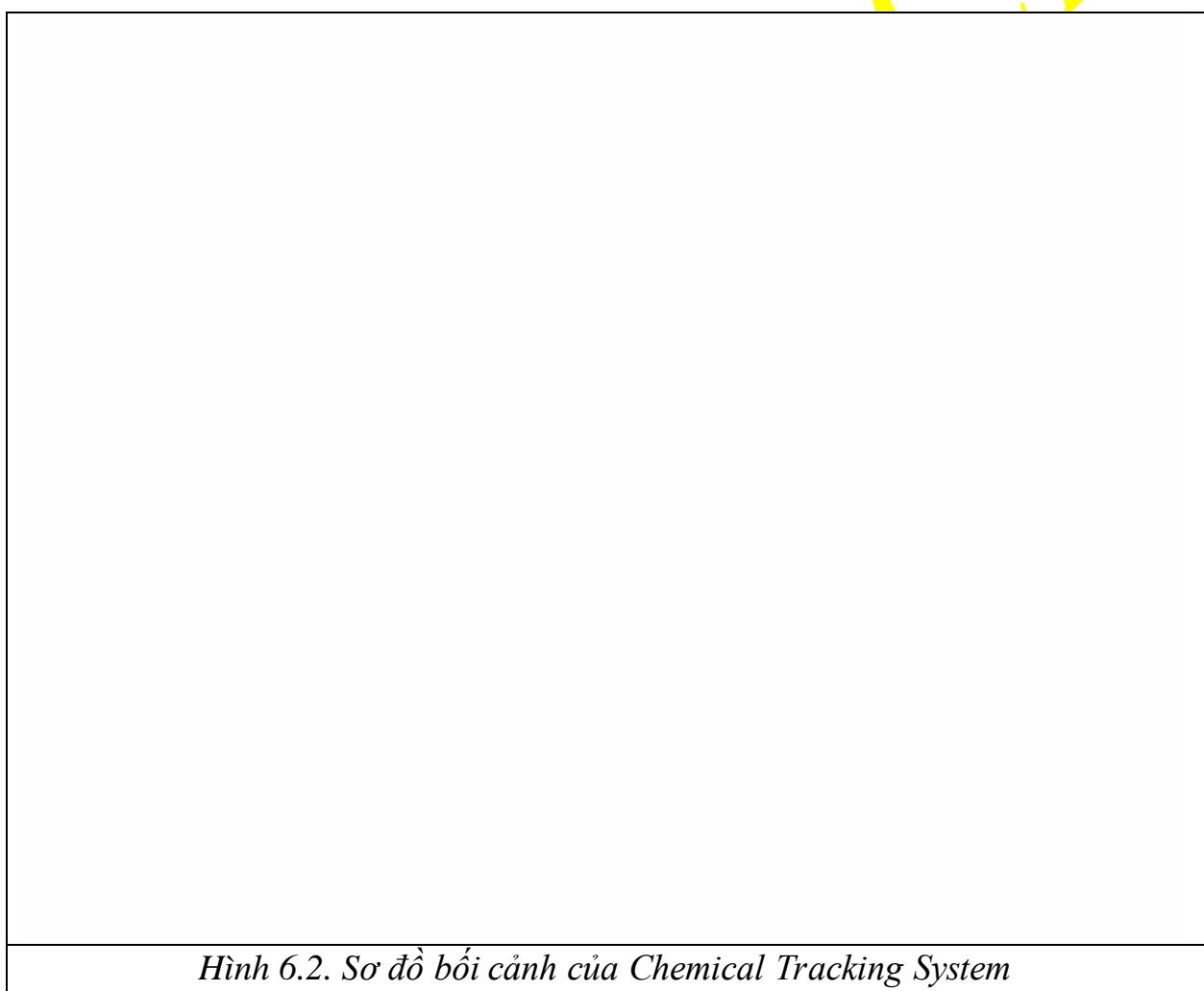
5. CÁC YẾU TỐ THÀNH CÔNG (PRODUCT SUCCESS FACTORS)

Hãy xác định các yếu tố thành công cần được định nghĩa và đo lường như thế nào. Nếu có thể hãy thiết lập các tiêu chuẩn đo lường sự đáp ứng các mục tiêu của dự án. Một số trong các tiêu chuẩn đó là thị phần, doanh thu, các chỉ số đánh giá sự hài lòng của khách hàng, lượng giao dịch xử lý.

III. SƠ ĐỒ BỐI CẢNH (CONTEXT DIAGRAM)

Mô tả phạm vi thiết lập đường biên giữa hệ thống chúng ta đang phát triển với toàn bộ những gì xung quanh nó. Sơ đồ bối cảnh thể hiện bằng hình vẽ minh họa đường biên này bằng cách thể hiện các kết nối giữa hệ thống đang được phát triển, hoặc hệ thống đã được xác định, với thế giới bên ngoài. Sơ đồ bối cảnh xác định các thực thể (entities) bên ngoài hệ thống có giao tiếp với hệ thống theo một cách nào đó (được gọi là *terminators* hoặc *external entities*), xác định luồng dữ liệu và luồng vật chất (flow of data or material) giữa mỗi thực thể bên ngoài và hệ thống. Bạn có thể in kèm sơ đồ bối cảnh trong tài liệu SRS.

Hình 6.2 minh họa một phiên bản đơn giản hóa sơ đồ bối cảnh của Chemical Tracking System. Thực thể Chemical Tracking System được vẽ như một hình tròn đơn, sơ đồ bối cảnh không cần cho người đọc biết các dữ liệu và quy trình bên trong hệ thống. Các luồng (flows) biểu diễn mỗi thông tin (“chemical request”), hoặc thực thể vật lý (“chimical container”). Terminators được thể hiện bằng các hình chữ nhật, có thể biểu diễn cho các lớp người dùng (user classes) (“Chemists”), hoặc tổ chức (“Purchasing Department”) hoặc các hệ thống máy tính khác (“Training Database”).



Hình 6.2. Sơ đồ bối cảnh của Chemical Tracking System

IV. NẮM CHẮC PHẠM VI (KEEPING THE SCOPE IN FOCUS)

Các yêu cầu kinh doanh (business requirements) được ghi lại trong tài liệu tầm nhìn và phạm vi (vision and scope) là cơ sở để xử lý tất cả các rắc rối liên quan đến Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

phạm vi của dự án. Tài liệu tầm nhìn và phạm vi (vision and scope) giúp bạn đánh giá liệu các tính năng và yêu cầu được đề xuất có nên đưa vào tài liệu này hay không. Câu hỏi đầu tiên mà bạn cần trả lời khi nghe bắt cứ ai đề xuất yêu cầu mới hoặc thay đổi yêu cầu đã có là: “Cái này có thuộc phạm vi không?”

Một số yêu cầu được đề xuất là nằm ngoài phạm vi. Chúng có thể là một ý tưởng tốt, nhưng chúng phải được xử lý trong một dự án khác hoặc trong một phiên bản khác của sản phẩm. Với các đề xuất khác thuộc phạm vi dự án thì được đưa vào tập hợp các yêu cầu đã có của dự án, được sắp xếp thứ tự ưu tiên trong mối tương quan với các yêu cầu khác, được cân nhắc “đánh đổi” với các yêu cầu khác.

Khả năng thứ ba là yêu cầu mới được đề xuất nằm bên ngoài phạm vi nhưng là một ý tưởng có giá trị cao mà rất cần mở rộng phạm vi để đưa nó vào. Bạn cần làm mới lại tài liệu tầm nhìn và phạm vi (vision and scope) trong sự điều chỉnh của ban kiểm soát thay đổi, tất nhiên cần cân nhắc về thời gian thực hiện dự án. Khi phạm vi thay đổi, bạn cần đàm phán lại với khách hàng về ngân sách, thời gian, nguồn lực khác và nhân lực nếu cần thiết (đặc biệt yêu cầu mới cần những nhân viên có một kỹ năng khác). Tốt nhất là lịch biểu và nguồn lực được thay đổi theo thích hợp, bạn nên lập ngân sách ngay từ đầu cho sự tăng trưởng yêu cầu.

Các bước tiếp theo

- Có thể bạn sắp khai trương một dự án mới, hãy viết một tài liệu tầm nhìn và phạm vi (vision and scope) theo mẫu trong Hình 6-1. Bài tập này sẽ khó thực hiện nếu nhóm của bạn không sẵn sàng chia sẻ cùng nhau các hiểu biết chung về phạm vi dự án.

CHƯƠNG 7

TÌM KIẾM TIẾNG NÓI CỦA KHÁCH HÀNG

Nếu bạn chia sẻ niềm tin của tôi rằng sự tham gia của khách hàng là yếu tố then chốt trong việc chuyển giao một sản phẩm có chất lượng tuyệt hảo thì bạn phải làm sao để khách hàng tham gia càng sớm càng tốt vào dự án. Sự thành công khi làm yêu cầu dẫn đến sự thành công khi phát triển phần mềm phụ thuộc vào việc có nghe được tiếng nói ngay từ sớm của khách hàng hay không. Chương 6 đã thảo luận về việc lấy tiếng nói của một kiểu khách hàng là các nhà tài trợ, các nhà marketing, thì chương này tập trung vào kiểu khách hàng khác - những người dùng, tương ứng là yêu cầu người dùng (user requirements). Để tìm kiếm tiếng nói của khách hàng, bạn cần thực hiện các bước sau:

- Xác định nguồn từ đó có thể lấy yêu cầu người dùng của dự án.
- Xác định các lớp khách hàng khác nhau của dự án.
- Hãy làm việc với các cá nhân đại diện cho những lớp khách hàng khác nhau.
- Thoả thuận với người ra quyết định cuối cùng đối với các vấn đề liên quan đến dự án.

Sự tham gia của khách hàng là cách duy nhất để tránh được các khoảng cách kỳ vọng giữa cái khách hàng mong muốn và sản phẩm thực sự được sản xuất ra. Tuy nhiên, điều đó không có nghĩa là hỏi một vài khách hàng một số câu hỏi về kỳ vọng của họ rồi sau đó bắt tay vào viết mã nguồn. Nếu nhà phát triển làm chính xác cái mà khách hàng đề xuất ngay từ đầu thì có thể họ sẽ phải làm lại hệ thống một lần nữa bởi khách hàng, trong đa số các trường hợp, là không thật sự biết mình cần gì, và nhà phát triển cũng vậy.

I. NGUỒN YÊU CẦU (SOURCES OF REQUIREMENTS)

Các yêu cầu phần mềm có thể đến từ nhiều nơi và điều này phụ thuộc vào bản chất sản phẩm của bạn, phụ thuộc vào môi trường phát triển. Dưới đây là các nguồn yêu cầu.

Phỏng vấn và thảo luận với khách hàng tiềm năng (Interviews and discussions with potential customers)

Hãy tìm kiếm các khách hàng tiềm năng và nói chuyện với họ. Chương này thảo luận làm thế nào để tìm các đại diện người dùng phù hợp, chương 8 chỉ mặt đặt tên các kỹ thuật suy luận yêu cầu từ các đại diện khách hàng.

Tìm kiếm tài liệu mô tả sản phẩm hiện thời khách hàng đang sử dụng hoặc các sản phẩm cạnh trang (Documents that describe current or competing products)

Tài liệu có thể mô tả các tiêu chuẩn mà sản phẩm phải tuân theo như các chuẩn công nghiệp, các quy định của chính phủ.

Các đặc tả của yêu cầu hệ thống (System requirements specifications)

Một mô tả bao gồm cả đặc tả yêu cầu hệ thống ở mức cao của phần mềm và phần cứng tương ứng. Một tập con các yêu cầu hệ thống cũng được phân bổ cho mỗi phân hệ của phần mềm (Nelsen 1990).

Các báo cáo bài toán và đề nghị nâng cấp hệ thống hiện tại (Problem reports and enhancement requests for a current system)

Nhóm hỗ trợ sản phẩm của doanh nghiệp (help desk) là một nguồn yêu cầu rất có giá trị. Họ chính là nơi thu thập thông tin về phần mềm hiện tại và các đề nghị cải tiến của người dùng.

Các điều tra marketing và các bảng hỏi người dùng (Marketing surveys and user questionnaires)

Các cuộc điều tra thường được sử dụng để thu được một lượng dữ liệu định lượng hoặc thống kê lớn từ một lượng người dùng lớn. Hãy điều tra đúng người và hãy đặt đúng câu hỏi.

Quan sát người dùng làm việc (Observing users at work)

Khi quan sát người dùng làm việc thì nhà phân tích có thể rút ra được nhiều thông tin có giá trị. Quan sát luồng công việc của người dùng (user's workflow) trong môi trường làm việc của họ cho phép nhà phân tích thấy được những vấn đề mà người dùng đang phải đối mặt khi làm việc trên hệ thống hiện tại và từ đó xác định

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

cách mà hệ thống mới có thể hỗ trợ để luồng công việc hiệu quả hơn (McGraw and Harbison 1997; Beyer and Holtzblatt 1998). Khi quan sát, nhà phân tích hiểu biết chính xác hơn về các hoạt động của người dùng thay vì chỉ đơn giản hỏi họ, hay đề nghị họ viết ra giấy các bước xử lý công việc của họ. Nhà phân tích cần trùu tượng hóa và tổng quát hóa công việc của người dùng để đảm bảo các yêu cầu nắm bắt được có thể ứng dụng cho một lớp người dùng như một tổng thể chứ không chỉ cho một cá nhân. Một nhà phân tích giỏi có thể đề xuất các ý tưởng để cải tiến quy trình nghiệp vụ hiện tại của người dùng.

Các phân tích kịch bản tác vụ người dùng (Scenario analysis of user tasks)

Bằng cách xác định các tác vụ (tasks) mà người dùng cần phải hoàn thành với hệ thống, thường thông qua những kịch bản cụ thể hoặc một dãy các hành động (đôi khi còn gọi là “stories”), nhà phân tích có thể chuyển giao các yêu cầu chức năng (functional requirements) cần thiết cho phép người dùng thực hiện các tác vụ (tasks) đó. Đây là nội dung cơ bản của cách tiếp cận use-case. (Xem Chương 8).

II. CÁC LỚP NGƯỜI DÙNG (USER CLASSES)

Một người dùng thì khác với một người dùng khác, khác từ góc nhìn khác nhau, khác về tầm xuất mà họ sử dụng sản phẩm, khác về miền ứng dụng, về kinh nghiệm sử dụng máy tính, về các tính năng được sử dụng, về quy trình nghiệp vụ được sử dụng, về vị trí địa lý, về các mức ưu tiên truy nhập. Bạn có thể nhóm những người dùng vào các lớp khác nhau căn cứ vào sự tương tự của một số người dùng về những gì phân biệt họ đã nói ở trên. Nhu cầu của các lớp người dùng khác nhau thì khác nhau về tầm quan trọng và do đó khác nhau về mức độ ưu tiên (Gause and Lawrence 1999).

Mỗi lớp người dùng sẽ có một tập các yêu cầu chức năng (functional requirements) và phi chức năng riêng. Ví dụ, một người dùng không kinh nghiệm thì quan tâm đến việc hệ thống có dễ học hay không, vì vậy mà các menus, các wizards là quan trọng. Còn những người dùng nhiều kinh nghiệm và làm việc nhiều giờ với hệ thống mỗi ngày thì họ cần thao tác nhanh trên hệ thống, vì vậy các phím tắt, các macro, ... là quan trọng.

Có một số người bị ảnh hưởng bởi phần mềm bạn đang thực hiện nhưng họ lại không phải là người dùng trực tiếp, nghĩa là không truy nhập vào dữ liệu của phần mềm, không đọc các báo cáo. Họ tạo nên một lớp người dùng phụ (additional user classes) của phần mềm.

Có những lớp người dùng không phải là con người, đó có thể là các ứng dụng khác, các hardware components có trao đổi thông tin với phần mềm của bạn.

Định danh và đặc tả các lớp người dùng khác cho sản phẩm của bạn ngay từ sớm trong vòng đời của dự án, bạn có thể suy luận các yêu cầu từ đại diện của mỗi lớp người dùng. Tôi biết một trong số các công ty phát triển các ứng dụng thương mại có 65 công ty khách hàng. Khi họ thực hiện sản phẩm, họ nhóm tất cả các khách hàng vào 6 lớp người dùng để đặc tả yêu cầu của mỗi lớp. Tài liệu hoá các lớp người dùng và yêu cầu của mỗi lớp trong SRS. Xem ví dụ Bảng 7-1 nói về yêu cầu của mỗi lớp người dùng.

BẢNG 7-1. Các lớp người dùng của Chemical Tracking System	
Nhà hoá học (Chemists)	<i>Nhà hoá học sử dụng hệ thống để gửi yêu cầu về hoá chất của mình tới các nhà cung cấp và các kho chứa hoá chất. Mỗi nhà hoá học sẽ sử dụng hệ thống một vài lần mỗi ngày, chủ yếu là để giám sát các thùng chứa hoá chất trong và ngoài phòng thí nghiệm. Các nhà hoá học có nhu cầu tìm kiếm các cấu trúc hoá học riêng biệt (specific chemical structures) trên catalog của nhà cung cấp, các cấu trúc này được vẽ ra bằng các công cụ vẽ cấu trúc hiện có của họ.</i>
Người mua (Buyers)	<i>Người mua trong bộ phận mua sắm xử lý các đề nghị mua hoá chất được đề xuất từ những người sử dụng khác, họ giao tiếp bằng hệ thống với nhà cung cấp bên ngoài để đặt hàng và giám sát đơn hàng (orders). Họ chỉ có chút ít hiểu biết về hóa học và vì vậy mà cần các thiết bị truy vấn đơn giản để tìm kiếm catalog của các nhà cung cấp. Người mua sẽ không sử dụng tính năng giám sát thùng chứa của hệ thống. Mỗi người mua sẽ sử dụng hệ thống trung bình 10 lần mỗi ngày.</i>
Nhân viên kho hoá chất (Chemical stockroom staff)	<i>Nhân viên kho hóa chất gồm 3 kỹ thuật viên quản lý một kho chứa hơn 500.000</i>

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

thùng (container) hóa chất. Họ xử lý các đề xuất từ nhà hóa học để cung cấp các thùng hóa chất đã sẵn có, đặt hàng hóa chất mới từ các nhà cung cấp. Giám sát tình hình tất cả các thùng chứa trong và ngoài kho hóa chất. Họ là những người dùng duy nhất của tính năng báo cáo việc sử dụng hóa chất và sử dụng kho. Do lượng giao dịch cao nên các chức năng được sử dụng bởi các nhân viên kho cần phải được thiết kế hiệu quả và được tự động hóa.

Nhân viên Sức khoẻ và An toàn (Health and Safety staff)

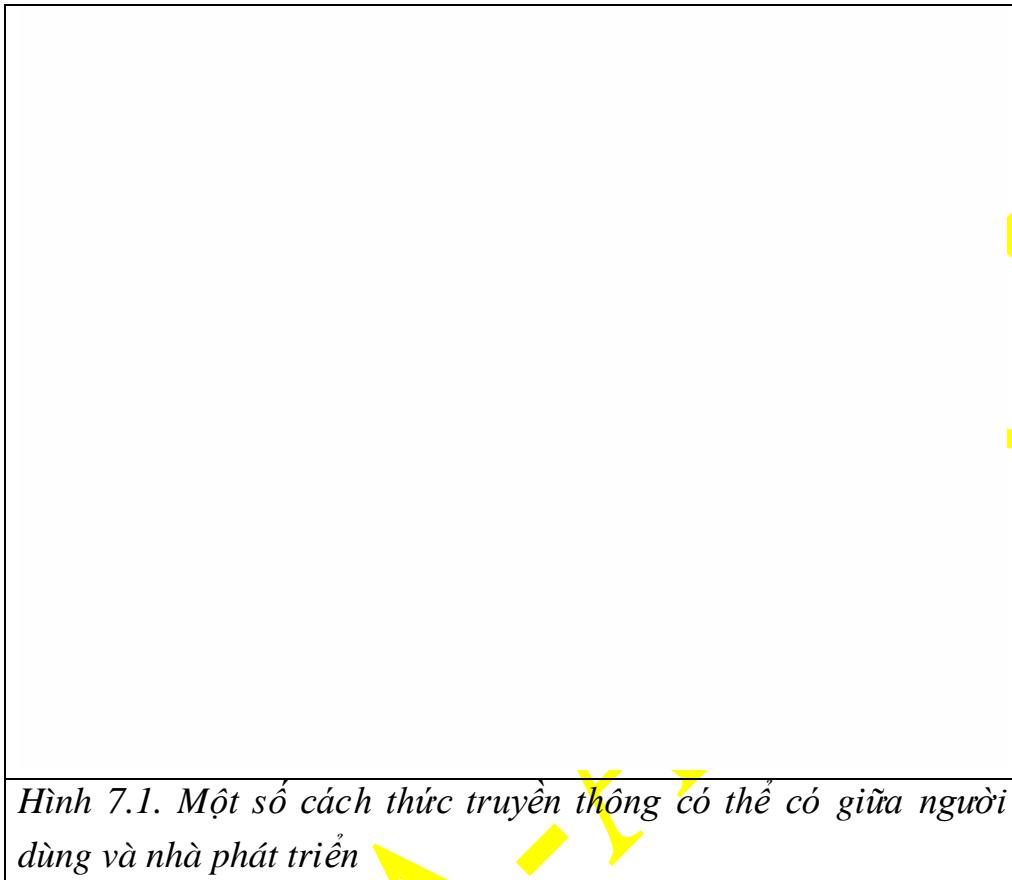
Nhân viên Sức khoẻ và An toàn sử dụng hệ thống chỉ để lập các báo cáo quý về việc tuân thủ các quy định của bang và liên bang trong việc sử dụng hóa chất. Khuôn mẫu của báo cáo cần được định nghĩa trước. Nhà quản lý Sức khoẻ và An toàn sẽ đề nghị các thay đổi trong báo cáo vài lần mỗi năm mỗi khi chính phủ có các thay đổi trong quy định. Các thay đổi của báo cáo được để mức ưu tiên cao nhất.

III. TÌM KIẾM ĐẠI DIỆN NGƯỜI DÙNG (FINDING USER REPRESENTATIVES)

Mỗi kiểu dự án – như các hệ thống thông tin doanh nghiệp, các sản phẩm chứa phần mềm nhúng, các chương trình phát triển Web, các phần mềm được đặt hàng - cần những đại diện người dùng phù hợp để phát ngôn thay cho khách hàng trong quá trình suy luận yêu cầu. Các đại diện người dùng cần được tham gia vào toàn bộ chu trình phát triển hệ thống, không chỉ trong giai đoạn làm yêu cầu. Bạn cần một nhóm đa dạng những đại diện người dùng với nhiều trình độ, kinh nghiệm khác nhau tham gia phát triển hệ thống cùng bạn, mỗi người đại diện đều là người quan trọng nhất trong mỗi cộng đồng người dùng.

Dễ dàng nhất để có được những quan hệ tốt đẹp với người dùng là khi bạn phát triển các ứng dụng được sử dụng trong chính doanh nghiệp của bạn. Tuy nhiên, nếu bạn phát triển các ứng dụng thương mại, thì bạn có thể mời những người thử nghiệm bản beta hoặc những người tại các điểm triển khai tham gia ngay từ sớm vào quy trình phát triển. Nếu bạn thiết lập một nhóm tập trung (focus group), hãy đảm bảo rằng những người tham gia thật sự đại diện cho kiểu người dùng sản phẩm của bạn, trong đó có những khách hàng hiểu biết nhiều và cả những người dùng kém kinh nghiệm.

Hình 7-1 minh họa một số cách thức truyền thông điển hình giữa khách hàng (người dùng) và nhà phát triển. Thông tin phản ánh trung thực nhất khi có sự kết nối trực tiếp giữa khách hàng và nhà phát triển.



Để đảm bảo tính chính xác của yêu cầu, một số nhà phân tích yêu cầu có kinh nghiệm làm việc với người dùng hoặc những người tham gia trung gian khác (sales, product managers) để tập hợp, đánh giá và tổ chức lại yêu cầu trước khi chuyển giao cho nhà phát triển. Hãy thật sự đảm bảo rằng bạn hiểu được các rủi ro khi sử dụng các thông tin về sản phẩm do nhân viên marketing cung cấp, tương tự bạn cũng hiểu được các rủi ro khi sử dụng thông tin của những người đại diện khác. Bất chấp các trở ngại mà bạn gặp phải khi làm việc với các đại diện lớp người dùng, bất chấp các chi phí mà bạn phải gánh chịu, sản phẩm của bạn và khách hàng sẽ vẫn trở nên tồi tệ hơn nếu bạn không chọn được những người có thể cung cấp những thông tin tốt nhất cho bạn.

IV. NGƯỜI TRỢ GIÚP SẢN PHẨM (THE PRODUCT CHAMPION)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Nhiều năm trước đây, tôi là một thành viên chính trong một nhóm phát triển phần mềm nhỏ hỗ trợ các hoạt động nghiên cứu khoa học tại một công ty lớn. Khi chúng tôi thành lập nhóm, chúng tôi quyết định rằng mỗi dự án sẽ gồm một số nhỏ các thành viên chủ chốt đến từ cộng đồng người dùng của chúng tôi nhằm cung cấp cho chúng tôi tiếng nói của khách hàng. Chúng tôi gọi những người này là *người trợ giúp sản phẩm* hoặc *người trợ giúp dự án* (*product champions* or *project champions*) (Wiegers 1996a). Cách tiếp cận này rất hiệu quả để xây dựng mối quan hệ khách hàng – nhà phát triển.

Mỗi người trợ giúp sản phẩm đại diện cho một lớp người dùng riêng và họ đóng vai trò như là người phát ngôn của lớp người dùng đó. Những người trợ giúp sản phẩm phải là những người dùng thực sự, chứ không phải là người dùng đại diện như nhà tài trợ, khách hàng mua sắm, nhân viên marketing. Người trợ giúp sản phẩm tập hợp các yêu cầu từ các thành viên khác thuộc lớp khách hàng của họ. Mỗi người trợ giúp sản phẩm chịu trách nhiệm điều hòa sự thiếu nhất quán hoặc sự thiếu tương thích trong các yêu cầu được diễn giải bởi người mà họ đại diện. Mục đích của mỗi người trợ giúp sản phẩm là làm việc với nhà phân tích để hình thành nên một tập các yêu cầu thống nhất của lớp người dùng của họ. Phát triển yêu cầu, vì vậy, là sự chia sẻ trách nhiệm giữa nhà phân tích và một số ít khách hàng, mặc dù nhà phân tích thường phải viết tài liệu yêu cầu.

Người trợ giúp sản phẩm tốt cần có một cái nhìn sáng suốt về hệ thống mới và có một sự nhiệt tình cao do, hơn ai hết, họ hiểu được lợi ích mà hệ thống mới mang lại cho họ và các đồng nghiệp. Người trợ giúp sản phẩm cũng phải là một người làm truyền thông giỏi và nhận được sự tôn trọng của các thành viên trong lớp người dùng mà họ đại diện. Họ có một hiểu biết tốt về miền ứng dụng và có đủ kinh nghiệm về phần mềm để biết yêu cầu nào của lớp người dùng là khả thi đối với công nghệ hiện tại, yêu cầu nào là hiện thực trong môi trường vận hành.

Sử dụng cách tiếp cận yêu cầu của các lớp người dùng thông qua người trợ giúp sản phẩm sẽ đạt hiệu quả tối ưu nếu người này có thể ra quyết định thay mặt cho toàn bộ người dùng mà họ đại diện. Nếu các quyết định của người trợ giúp sản phẩm thường bị gạt bỏ bởi các nhà quản lý hoặc nhóm phần mềm thì vai trò trợ giúp của người này chưa tốt. Tuy nhiên, những người trợ giúp sản phẩm cần phải
Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

hiểu rằng họ không phải là các khách hàng duy nhất của sản phẩm. Tôi đã thấy cách tiếp cận người trợ giúp sản phẩm bị phá sản khi các cá nhân đóng vai trò này không truyền thông đầy đủ vấn đề tới các đồng nghiệp của họ, họ chỉ giới thiệu nhu cầu của riêng họ và ý tưởng về sản phẩm của riêng họ.

1. TÌM KIẾM NGƯỜI TRỢ GIÚP SẢN PHẨM (FINDING PRODUCT CHAMPIONS)

Nếu bạn đang phát triển một sản phẩm thương mại chứ không phải một sản phẩm nội bộ (internal software), thì bạn sẽ gặp nhiều khó khăn để tìm kiếm một người làm việc như một người trợ giúp sản phẩm từ bên ngoài công ty bạn. Nếu bạn có những mối quan hệ làm việc khăng khít với một số khách hàng doanh nghiệp, thì có thể họ sẽ rất vui lòng (thậm chí là đề nghị) được tham gia vào quá trình suy luận yêu cầu. Khi đó bạn sẽ phải đối mặt với thách thức là làm thế nào để tránh nghe theo các yêu cầu một chiều của một khách hàng mà bỏ băng đi yêu cầu của các khách hàng khác. Nếu bạn có một lượng khách hàng đa dạng thì bạn có thể xác định các yêu cầu chính (core requirements) chung đối với mọi khách hàng và định nghĩa các yêu cầu bổ sung (additional requirements) cụ thể đối với từng loại khách hàng hoặc từng lớp người dùng.

Trừ phi bạn tham gia những hội thảo thương mại (trade shows) hoặc các sự kiện khác của giới doanh nghiệp, còn không thì người trợ giúp sản phẩm không thể trao đổi các vấn đề với các đồng nghiệp ở các công ty khác. Rủi ro của việc thảo luận như vậy là để lộ chi tiết các vấn đề kinh doanh của doanh nghiệp bạn, làm mất đi lợi thế cạnh tranh của bạn. Điều gì sẽ xảy ra nếu một người trợ giúp sản phẩm biết những gì mà bạn đã lập kế hoạch cho các sản phẩm tương lai lại chia sẻ các thông tin nội bộ này cho những người không hề nhận thức được về điều đó? Các thỏa thuận bí mật cũng không thể đảm bảo thông tin đó được giữ kín. Khả năng khác là công ty của người trợ giúp sản phẩm có thể quyết định không mua phiên bản đầu tiên của sản phẩm do phiên bản thứ hai tốt hơn đã sắp sửa phát hành. Bạn có thể chia sẻ với những người trợ giúp sản phẩm một số lợi ích kinh tế để đổi lại sự tham gia của họ, ví dụ như giảm giá hoặc thậm chí trả tiền cho thời gian họ đã tham gia làm việc về yêu cầu.

Một sự lựa chọn khác là thuê một người trợ giúp sản phẩm phù hợp có một nền tảng kiến thức đúng đắn. Một công ty phát triển một hệ thống hỗ trợ bán lẻ đã thuê làm việc toàn thời gian 3 nhà quản lý bán lẻ với tư cách người trợ giúp sản phẩm.

2. CÁC KỲ VỌNG CỦA NGƯỜI TRỢ GIÚP SẢN PHẨM (PRODUCT CHAMPION EXPECTATIONS)

Để những người trợ giúp sản phẩm làm việc thành công, hãy ghi thành tài liệu các kỳ vọng của bạn về họ. Nếu người trợ giúp sản phẩm không thực hiện tất cả các công việc mà bạn mong muốn, bạn có thể sử dụng các kỳ vọng đã được viết thành văn để đàm phán về trách nhiệm chính xác của người trợ giúp sản phẩm. Bảng 7-2 xác định một số hoạt động mà người trợ giúp sản phẩm có thể thực hiện.

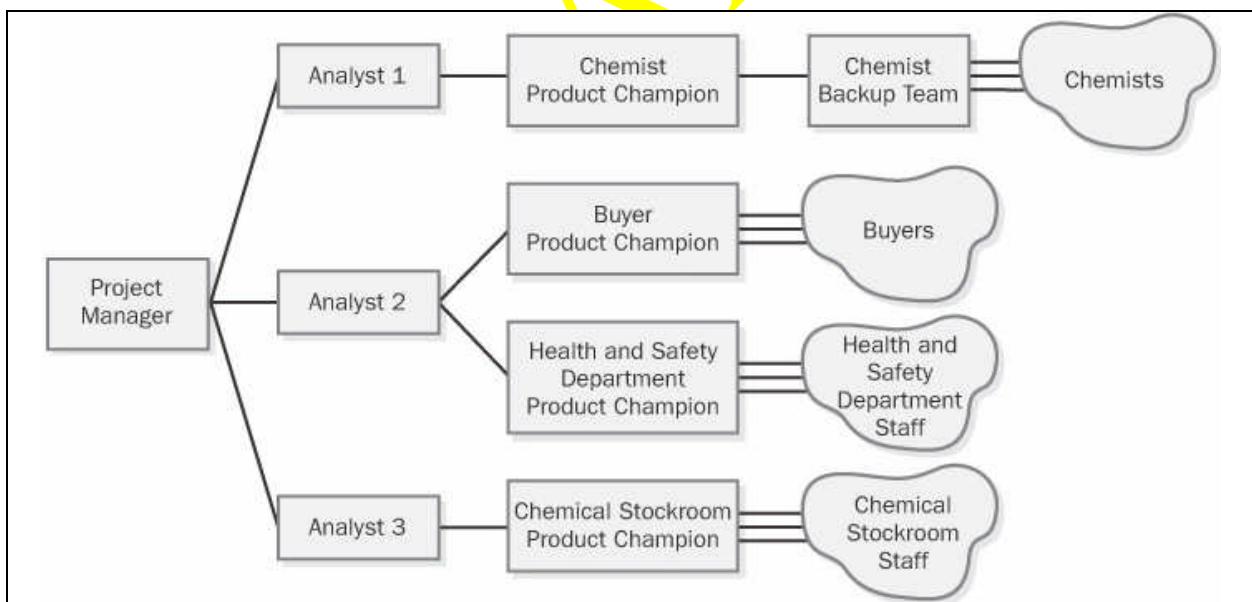
BẢNG 7-2: Các hoạt động có thể có của người trợ giúp sản phẩm	
Phân loại	Hoạt động
Lập kế hoạch	<ul style="list-style-type: none"> Định nghĩa phạm vi và giới hạn của sản phẩm Định nghĩa các giao diện ngoài với hệ thống khác Định nghĩa quá trình chuyển đổi (transition) từ hệ thống hiện tại tới hệ thống mới
Yêu cầu	<ul style="list-style-type: none"> Phỏng vấn những người dùng khác mà họ đại diện để thu thập yêu cầu của họ. Phát triển các kịch bản thường dùng và tình huống sử dụng (use-cases) Phân giải xung đột giữa các yêu cầu được đề xuất Định nghĩa các ưu tiên thực thi Đặc tả các thuộc tính chất lượng và các yêu cầu phi chức năng khác Đánh giá các nguyên mẫu giao diện người dùng
Kiểm tra và hợp lý hóa (Verification and Validation)	<ul style="list-style-type: none"> Thanh tra tài liệu yêu cầu Định nghĩa các tiêu chuẩn chấp nhận của người dùng Phát triển test cases từ các kịch bản thường dùng Thực thi beta testing Cung cấp tập dữ liệu test (test data sets)
Trợ giúp	<ul style="list-style-type: none"> Viết sách hướng dẫn người dùng và online help

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

người dùng (User Aids)	<ul style="list-style-type: none"> Chuẩn bị đào tạo Giới thiệu các mô tả sản phẩm cho các đồng nghiệp
Quản lý thay đổi	<ul style="list-style-type: none"> Đánh giá và xếp thứ tự ưu tiên các sửa chữa lỗi Đánh giá và xếp thứ tự ưu tiên các đề nghị nâng cấp (enhancement request) Đánh giá ảnh hưởng trên người dùng của các thay đổi yêu cầu được đề xuất Chỉ định ban kiểm soát thay đổi để ra các quyết định về thay đổi

3. NHIỀU NGƯỜI TRỢ GIÚP SẢN PHẨM (MULTIPLE PRODUCT CHAMPIONS)

Hệ thống giám sát hóa chất có 4 lớp người dùng, vì vậy cần nhiều người trợ giúp sản phẩm được lựa chọn từ cộng đồng người dùng nội bộ tại Contoso Pharmaceuticals. Hình 7-2 minh họa làm thế nào để nhà quản lý dự án thiết lập một nhóm những người trợ giúp sản phẩm để tập hợp yêu cầu đúng đắn (right requirements) từ các nguồn đúng đắn (right sources).



HÌNH 7-2. Mô hình người trợ giúp sản phẩm của Chemical Tracking System

Những người trợ giúp sản phẩm không làm toàn thời gian, họ chỉ dành mỗi tuần vài giờ để làm việc với nhóm dự án. 3 nhà phân tích làm việc với 4 người trợ giúp sản phẩm (lớp người dùng Buyer và Health and Safety nhỏ và có ít yêu cầu) để suy. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com.

luận, phân tích, tài liệu hoá các yêu cầu. Một nhà phân tích sẽ tổng hợp và làm nhát quán tất cả các điều vào thành SRS.

Sẽ là không thực tế nếu kỳ vọng một người có thể cung cấp tất cả các nhu cầu đa dạng của một lớp người dùng lớn gồm hàng trăm nhà hoá học ở Contoso Pharmaceuticals. Vì vậy, người trợ giúp sản phẩm của lớp người dùng các nhà hoá học sẽ được hỗ trợ bởi một nhóm 5 nhà hoá học khác từ các bộ phận khác của công ty. Các nhà hoá học khác tập hợp đầu vào từ các đồng nghiệp của họ trong một số bộ phận, thảo luận về CTS và cung cấp thông tin hiện tại của dự án cho các đồng nghiệp. Cách tiếp cận phân cấp này sẽ làm tăng số lượng người dùng tham gia vào suy luận yêu cầu, tránh phải tổ chức nhiều hội thảo, hoặc phải phỏng vấn nhiều người.

V. AI RA QUYẾT ĐỊNH? (WHO MAKES DECISIONS?)

Trước đây, tôi đã gặp trưởng dự án tại một địa điểm vệ tinh của một tổ chức phát triển phần mềm lớn. Khi tôi hỏi rằng anh ta làm việc cho ai, anh ta đưa cho tôi 4 cái tên: một nhà quản lý phát triển địa phương, một nhà quản lý phát triển hỗ trợ tại văn phòng, hai nhà quản lý đơn vị kinh doanh trong cộng đồng khách hàng của anh ta. Anh chàng trưởng dự án này bị rối trí vì sự thiếu đồng bộ, thiếu kết hợp trong việc ra quyết định của 4 người kia, bởi sự đảo ngược thường xuyên của các quyết định.

Nếu các yêu cầu đến từ nhiều phía, sẽ rất khó khăn để phân giải các xung đột, làm sáng tỏ các nhập nhằng, điều hoà sự thiếu nhất quán. Ai đó sẽ phải phân xử các vấn đề về phạm vi chắc chắn sẽ xuất hiện. Sớm hơn trong mỗi dự án, bạn cần phải thỏa thuận về việc ai là người ra các quyết định liên quan đến yêu cầu. Nếu có cái gì đó chưa sáng sủa thì ai có quyền hạn và trách nhiệm ra các quyết định.

Không có câu trả lời đúng tổng quát cho tất cả các vấn đề liên quan đến yêu cầu. Ngay cả các yêu cầu được đưa ra bởi những người trợ giúp sản phẩm khác nhau cũng có thể xung đột, vậy cần phải phân giải. Nói chung các quyết định cần phải được xuất phát từ mức thấp nhất có thể trong cơ cấu phân tầng của tổ chức, những người ở mức thấp này hiểu hơn ai hết các vấn đề của họ và được thông tin đầy đủ nhất.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Các tình huống ra quyết định sau thường gặp trong khi làm dự án và các đề xuất xử lý. Người quản lý dự án của bạn cần xác định ai sẽ ra quyết định cần làm gì khi một tình huống như vậy xuất hiện và ai là người cất tiếng nói sau cùng nếu các thoả thuận không đạt được.

- Nếu những người dùng cá nhân không thoả thuận được với nhau về yêu cầu thì người trợ giúp sản phẩm sẽ quyết định. Sử dụng cách tiếp cận người trợ giúp sản phẩm là hy vọng anh ta sẽ phân giải các xung đột yêu cầu xuất phát từ lớp người dùng anh ta làm đại diện.
- Nếu các lớp người dùng khác nhau có các nhu cầu không tương thích, hãy xác định lớp người dùng nào quan trọng hơn để đáp ứng họ trước. Hãy dựa vào các mục tiêu kinh doanh để xác định tầm quan trọng của các lớp người dùng, để xác định lớp người dùng nào cần được đáp ứng trước nhất. Một khách hàng chính có thể dẫn dắt việc phát triển các tính năng chính trong khi các yêu cầu của các khách hàng khác có thể được đưa vào trong các phiên bản sau.
- Các yêu cầu được diễn giải bởi các nhà quản lý của khách hàng đôi khi mâu thuẫn với các yêu cầu được diễn giải bởi những người dùng trong giới hạn quản lý của họ. Trong khi yêu cầu người dùng (user requirements) phải song hành với yêu cầu kinh doanh (business requirements), nhưng các nhà quản lý lại không có kinh nghiệm thực tế về việc sử dụng phần mềm, vì vậy họ nên tuân theo các nhu cầu chi tiết của người dùng (user needs) và các đặc tả chức năng của sản phẩm - những thứ mà người trợ giúp sản phẩm sẽ nêu ra với các nhà phát triển. Tránh để các nhà phát triển phân xử mâu thuẫn yêu cầu giữa các khách hàng.
- Khi sản phẩm mà nhà phát triển nói rằng họ cần phải xây dựng mâu thuẫn với điều khách hàng nói thì khách hàng thường phải ra quyết định cuối cùng. Tuy nhiên, đừng rơi vào bẫy “khách hàng luôn luôn đúng” mà làm bất cứ cái gì khách hàng muốn. Thực tế, không phải khách hàng bao giờ cũng đúng. Khách hàng có quan điểm của họ và nhà phát triển cần hiểu và tôn trọng quan điểm ấy, nhưng hãy làm sao để mọi việc hài hòa.
- Một tình trạng tương tự phát sinh là bộ phận marketing trình bày các yêu cầu mâu thuẫn với những gì nhà phát triển nghĩ họ cần phải làm. Là đại diện của khách hàng nên những gì bên marketing nói rất có trọng lượng. Tuy vậy,

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

trên thực tế tôi đã gặp những trường hợp marketing không bao giờ nói không với một đề nghị của khách hàng, họ chả phân vân gì về tính khả thi hoặc chi phí khi thực hiện đề nghị đó. Tôi cũng đã gặp các trường hợp marketing cung cấp chút ít đầu vào và nhà phát triển cần phải định nghĩa sản phẩm và tự viết các yêu cầu.

Không có câu trả lời đúng duy nhất. Hãy quyết định ai sẽ là người ra quyết định về các yêu cầu trong dự án của bạn trước khi bạn đối mặt với các rắc rối do vấn đề đó gây ra.

Các bước tiếp theo

- Hình 7-1 là cách để bạn nghe được tiếng nói của khách hàng. Bạn đã bao giờ đối mặt với các rắc rối về truyền thông với khách hàng? Hãy xác định các cách thức truyền thông ngắn nhất và hiệu quả nhất bạn có thể dùng để thu thập các yêu cầu người dùng (user requirements).
- Hãy xác định các lớp người dùng khác nhau cho dự án của bạn và chọn ra người trợ giúp sản phẩm tốt nhất cho mỗi lớp người dùng. Sử dụng Bảng 7-2 để định nghĩa các chức năng mà bạn mong muốn người trợ giúp sản phẩm thực hiện. Hãy đàm phán với những người trợ giúp sản phẩm và các nhà quản lý của họ để tăng cường sự tham gia của họ vào dự án.
- Hãy xác định ai là người ra quyết định khi các yêu cầu có mâu thuẫn. Hiện nay, cách ra quyết định trong các dự án của bạn có hiệu quả không? Nó bị hỏng hóc ở đâu không? Người ra quyết định được lựa chọn đúng không? Nếu không hãy xác định những ai liên quan đến việc ra quyết định về yêu cầu và đề xuất các quy trình mà họ cần tuân theo để đạt được các thoả thuận về yêu cầu trước khi nhà phát triển phân giải các yêu cầu.

CHƯƠNG 8

NGHE TIẾNG NÓI CỦA KHÁCH HÀNG

Suy luận yêu cầu là công đoạn quan trọng nhất của công nghệ yêu cầu. Suy luận là tiến trình xác định, tìm hiểu về các nhu cầu và ràng buộc của các lớp người dùng khác nhau đối với sản phẩm. Suy luận tập trung tìm kiếm các yêu cầu người dùng (user requirements), mức trung gian trong bộ ba yêu cầu phần mềm. Các yêu cầu kinh doanh (business requirements) từ tài liệu tầm nhìn và phạm vi (vision and scope) là điểm bắt đầu của yêu cầu người dùng (user requirements), chúng xác định các tác vụ (tasks) mà người dùng cần thực hiện thông qua hệ thống. Từ các tác vụ (tasks) này, nhà phân tích có thể suy ra các yêu cầu chức năng (functional requirements) cụ thể mô tả hành vi của hệ thống, các hành vi này hỗ trợ người dùng thực hiện các tác vụ (tasks) cần thiết. Chương này xác định các nguyên tắc của công việc suy luận yêu cầu, nhấn mạnh việc ứng dụng use cases để nắm bắt các yêu cầu người dùng (user requirements).

Suy luận yêu cầu là bước đầu tiên nhằm bắc cầu qua khoảng cách giữa miền ứng dụng (problem domain) và giải pháp cuối cùng sẽ được xây dựng. Một kết quả quan trọng của công việc suy luận yêu cầu là hiểu biết chung của các stakeholders về nhu cầu của họ đối với sản phẩm. Trước khi các nhu cầu được hiểu, nhà phân tích, nhà phát triển và khách hàng có thể duyệt qua các giải pháp khác nhau. Những người tham gia suy luận yêu cầu cần phải chống lại sự cám dỗ bắt tay vào việc thiết kế hệ thống cho đến trước khi họ nắm chắc được vấn đề; ngược lại, bạn có thể kỳ vọng khi lật đi lật lại các vấn đề thiết kế thì bạn cũng có thể định nghĩa rõ hơn các yêu cầu. Tập trung vào suy luận yêu cầu trên các tác vụ (tasks) của người dùng – chứ không phải trên giao diện người dùng - sẽ tránh cho nhóm dự án phải xét lại các vấn đề về thiết kế sau này.

Suy luận, phân tích, đặc tả, kiểm tra (verification) không diễn ra theo một thứ tự tuyến tính chặt chẽ như vậy: các hoạt động này đan dệt lẫn nhau, tăng trưởng, và lặp đi lặp lại. Khi bạn làm việc với khách hàng, bạn sẽ hỏi và nghe thông tin mà họ trình bày (suy luận). Đồng thời bạn sẽ xử lý thông tin để có thể hiểu vấn đề, phân loại thông tin thành các loại khác nhau (various categories), liên kết nhu cầu của

khách hàng với các yêu cầu phần mềm có thể có (phân tích). Bạn sẽ sắp xếp các đầu vào của khách hàng thành các tài liệu thành văn và các sơ đồ (đặc tả). Tiếp theo bạn sẽ đề nghị các đại diện khách hàng để soát xét lại những gì bạn đã viết và sửa chữa bất cứ sai lỗi nào (kiểm tra). Đó chính là 4 bước trong quy trình phát triển yêu cầu.

Do tính đa dạng của các dự án phần mềm và văn hoá tổ chức, nên sẽ không có cách duy nhất nào để phát triển yêu cầu. Tiếp sau đây là một quy trình 14 bước bạn có thể sử dụng để hướng dẫn các nhóm dự án của mình trong việc phát triển yêu cầu. Khi bạn có thể hoàn thành bước 13 để đạt được một tập con yêu cầu bất kỳ thì bạn có thể thiết kế và xây dựng phân hệ thống tương ứng với tập con các yêu cầu đã có với lòng tin rằng bạn đang xây dựng một sản phẩm đúng đắn (right product).

I. HƯỚNG DẪN SUY LUẬN YÊU CẦU (REQUIREMENTS ELICITATION GUIDELINES)

QUY TRÌNH PHÁT TRIỂN YÊU CẦU GỢI Ý

1. Định nghĩa tầm nhìn và phạm vi của dự án (Define the project's vision and scope).
2. Xác định các lớp người dùng (Identify user classes).
3. Xác định các đại diện thích hợp của mỗi lớp người dùng (Identify appropriate representatives from the user classes).
4. Xác định người ra quyết định về yêu cầu và quy trình ra quyết định của họ (Identify requirements decision makers and their decision making process).
5. Chọn các kỹ thuật suy luận mà bạn sẽ dùng (Select elicitation techniques that you will use).
6. Ứng dụng các kỹ thuật suy luận để phát triển các use cases và xếp thứ tự ưu tiên các use cases đó cho từng phần của hệ thống (Apply elicitation techniques to develop and prioritize the use cases for a portion of the system).
7. Thu thập thông tin về các thuộc tính chất lượng và các yêu cầu phi chức năng khác từ người dùng (Gather information about quality attributes and other nonfunctional requirements from users).
8. Phác thảo các use cases từ các yêu cầu chức năng cần thiết (Elaborate the

- use cases into the necessary functional requirements).
9. Soát xét các mô tả use-case và các yêu cầu chức năng (Review the use-case descriptions and the functional requirements).
 10. Phát triển các mô hình phân tích, nếu cần thiết, để làm sáng tỏ hiểu biết của những người tham gia suy luận về các phần của yêu cầu (Develop the analysis models, if needed, to clarify elicitation participants' understanding of portions of the requirements).
 11. Phát triển và đánh giá các nguyên mẫu giao diện người dùng nhằm trực quan hóa các yêu cầu chưa được hiểu kỹ (Develop and evaluate user interface prototypes to help visualize requirements that are not clearly understood).
 12. Phát triển các test cases dưới dạng ý tưởng từ các use cases (Develop conceptual test cases from use cases).
 13. Sử dụng các test cases để kiểm tra các use cases, các yêu cầu chức năng, các mô hình phân tích, các nguyên mẫu (Use the test cases to verify use cases, functional requirements, analysis models, and prototypes).
 14. Lặp lại các bước từ 6 đến 13 trước khi thực hiện thiết kế và xây dựng từng phần của hệ thống (Repeat step 6 to step 13 before proceeding design and construction of each portion of the system).

Suy luận yêu cầu có lẽ là việc khó khăn nhất, then chốt nhất, thu hút lối nhất và là khía cạnh dễ bị ảnh hưởng nhất bởi quá trình giao tiếp trong nhóm dự án khi phát triển phần mềm. Suy luận chỉ thành công khi được tiến hành trong một môi quan hệ khách hàng – nhà phát triển tốt đẹp. Nhà phân tích cần xây dựng một môi trường tạo điều kiện cho sự khảo sát tỷ mỷ các vấn đề liên quan đến sản phẩm đang được đặc tả.

Suy luận yêu cầu là một hoạt động đòi hỏi sự hợp tác cao độ chứ không chỉ là sự sao chép đơn giản lại từ những lời khách hàng nói rằng họ cần. Theo một nhà phân tích, bạn cần phải thăm dò được phía dưới bề mặt của các yêu cầu mà khách hàng đề nghị để hiểu được nhu cầu thật sự của họ. Hãy hỏi các câu hỏi mở để giúp họ hiểu tốt hơn các quy trình kinh doanh đang có của họ và suy nghĩ xem hệ thống mới sẽ cải thiện hiệu quả của các quy trình đó như thế nào. Cần suy nghĩ về những thay đổi trong tác vụ (tasks) mà người dùng thực hiện khi đưa hệ thống mới vào. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Bạn hãy đặt bản thân mình vào vị trí cần phải học hỏi công việc của người dùng. Nhiệm vụ nào mà bạn cần thực hiện? Câu hỏi nào mà bạn phải đặt ra? Hãy sử dụng cách tiếp cận này để định hướng cách duyệt qua yêu cầu của bạn.

Cũng như thế, bạn cần thăm dò về các loại trừ: Cái gì ngăn cản người dùng hoàn thành thành công một tác vụ (tasks)? Người dùng nghĩ hệ thống cần đáp ứng thế nào đối với các điều kiện sai (error conditions)? Hãy đặt các câu hỏi bắt đầu bằng “Cái gì có thể nữa...”, “Cái gì xảy ra khi...”, “Bạn có thể cần phải...”. Tập trung chú ý nguồn của mỗi yêu cầu sao cho bạn có thể lần vết được về nguồn gốc phát sinh của nó.

Cố gắng làm sáng tỏ bất cứ giả định (assumption) nào mà khách hàng dựa trên đó để đưa ra yêu cầu hoặc dựa trên đó để làm việc, đặc biệt khi các giả định đó tạo ra xung đột. Hãy cố gắng đọc giữa các dòng chữ để xác định các tính năng hoặc các đặc tính mà khách hàng kỳ vọng được đưa vào phần mềm mà họ không nói ra một cách rõ ràng. Gause và Weinberg (1989) gợi ý sử dụng “context-free questions” – các câu hỏi cho phép suy luận thông tin về các đặc tả tổng thể của cả miền ứng dụng (problem domain) và các giải pháp tiềm tàng. Trả lời của khách hàng đối với các câu hỏi như “Mức chính xác nào là cần thiết cho sản phẩm?”, hoặc “Anh có thể giúp tôi hiểu được tại sao anh không đạt được thỏa thuận với câu trả lời của ông X?” có thể khiến bạn thấu hiểu vấn đề hơn là khi đặt các câu hỏi đóng cho khách hàng.

Suy luận phác lê tất cả các nguồn có thể có để từ đó mô tả miền ứng dụng (problem domain) hoặc các đặc tính mong muốn có trong giải pháp phần mềm. Chương 7 đã mô tả một số nguồn của yêu cầu phần mềm. Một khảo cứu đã chỉ ra rằng các dự án thành công thường sử dụng nhiều kênh truyền thông giữa nhà phát triển và khách hàng hơn các dự án không thành công (Kiel và Carmel 1995). Phỏng vấn các khách hàng cá nhân hoặc nhóm khách hàng là những người dùng tiềm năng vẫn là một nguồn truyền thống của yêu cầu phần mềm đối với cả phần mềm thương mại và phần mềm đặt hàng.

Sau mỗi cuộc phỏng vấn, hãy lập danh sách các vấn đề mà bạn cần thảo luận và hỏi lại những người mà bạn đã phỏng vấn. Các cuộc soát xét được thực hiện sớm. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

và thường xuyên là yếu tố cơ bản để suy luận yêu cầu thành công vì chỉ các cá nhân cung cấp yêu cầu mới có thể chỉ ra các yêu cầu có được thu thập chính xác hay không. Sử dụng các suy luận và phân tích để phân giải bất cứ xung đột và sự thiếu nhất quán nào.

II. CÁCH TIẾP CẬN USE-CASE (THE USE-CASE APPROACH)

Trong nhiều năm, nhà phân tích đã sử dụng các kịch bản hoặc các câu chuyện để mô tả cách một người dùng tương tác với hệ thống phần mềm để suy luận yêu cầu (McGraw and Harbison 1997). Ivar Jacobson (1992) và những người khác đã hình thức hoá cách này thành cách tiếp cận use-case để suy luận yêu cầu và mô hình hóa. Mặc dù các use cases xuất hiện từ lĩnh vực phát triển hướng đối tượng, nhưng chúng cũng có thể được ứng dụng trong các dự án tuân theo một cách phát triển bất kỳ do người dùng không quan tâm bạn xây dựng hệ thống như thế nào. Một số phương pháp thiết kế bao hàm các ký pháp để mô hình hóa use cases (Regnell, Kimbler, and Wesslén 1995; Booch, Rumbaugh, and Jacobson 1999). Ưu điểm của cách tiếp cận use cases là tập trung vào cái **người dùng** cần phải làm với hệ thống, chứ không như cách suy luận truyền thống là hỏi người dùng cái họ muốn **hệ thống** làm.

Một *use case* mô tả một dãy các tương tác giữa một hệ thống và một “actor” bên ngoài, kết quả là actor hoàn thành một tác vụ (tasks) cho ai đó. Một actor là một người, một ứng dụng phần mềm khác, một thiết bị phần cứng, một thực thể khác nào đó tương tác với hệ thống để đạt được một mục đích nào đó (Cockburn 1997a, b). Các actors thể hiện vai trò mà các thành viên của các lớp người dùng cần phải thực hiện. Ví dụ, use case “Đề nghị một hoá chất” của CTS bao hàm một actor gọi là “**Requester**” (Người đề xuất). Không có lớp người dùng nào trong CTS gọi là “**Requester**” cả, cả lớp người dùng các nhà hoá học và lớp người dùng nhân viên kho đều có thể đóng vai trò Requester.

Use cases cung cấp một cách thức để biểu diễn các yêu cầu người dùng (user requirements), chúng sống đôi với các yêu cầu kinh doanh (business requirements) của hệ thống. Nhà phân tích và khách hàng cần phải kiểm tra bất cứ use case được đề xuất nào để biết liệu nó có nằm trong phạm vi đã định của dự án hay không. Mục tiêu của các tiếp cận use cases là suy luận yêu cầu nhằm mô tả tất cả các tác *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

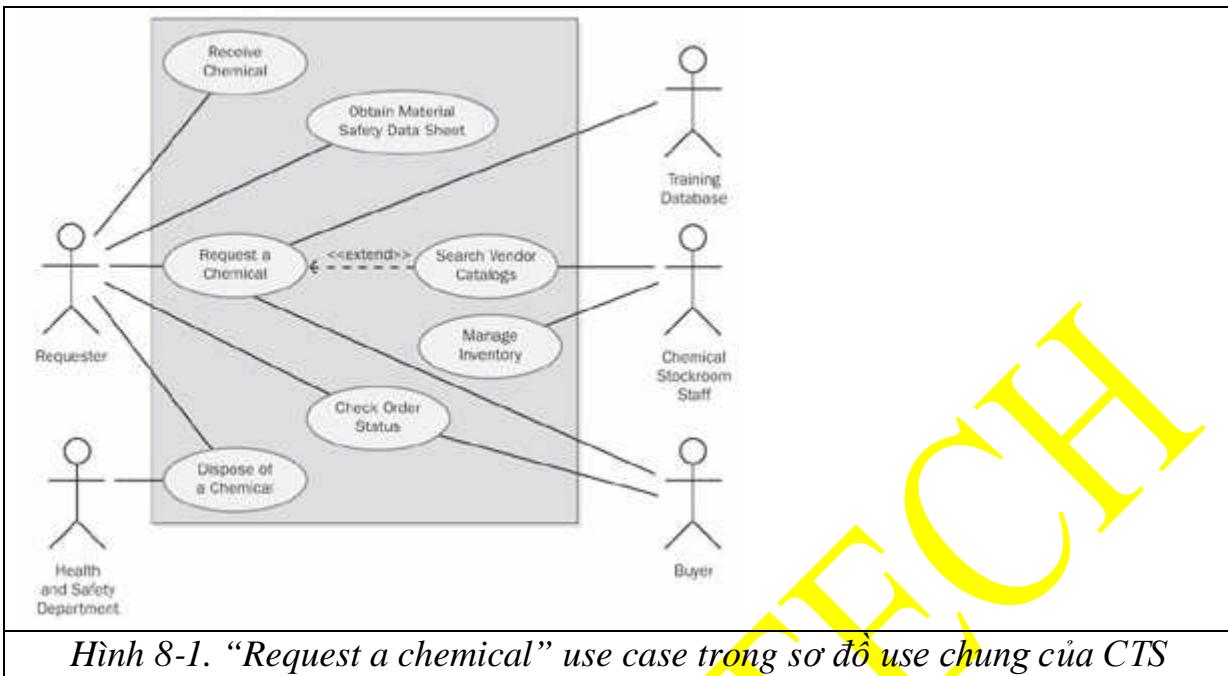
vụ (tasks) người dùng thực hiện với hệ thống. Về mặt lý thuyết, tập hợp các use cases cuối cùng sẽ bao chứa tất cả các tính năng mong muốn của hệ thống. Về mặt thực hành, bạn không chắc sẽ đạt được một bao chứa đầy đủ như vậy nhưng cách tiếp cận use cases sẽ giúp bạn suy luận yêu cầu tốt hơn bất cứ cách nào mà tôi đã biết.

1. USE CASES VÀ KỊCH BẢN SỬ DỤNG (USE CASES AND USAGE SCENARIOS)

Một use case có thể bao hàm một số các tác vụ (tasks) liên kết logic với nhau và một số chuỗi công việc tương tác lẫn nhau để hoàn thành các tác vụ này. Một use case, vì vậy, là một tập hợp các kịch bản sử dụng có liên quan đến nhau, trong đó mỗi kịch bản là một thể hiện cụ thể của một use case. Một kịch bản được gọi là một *tiến trình chuẩn* (*normal course*) của các sự kiện nối tiếp nhau tạo nên use case, nó cũng được gọi là tiến trình chính (*main course*), tiến trình cơ sở (*basic course*), luồng chuẩn (*normal flow*), hay là “đường yên lành” (*happy path*). Tiến trình chuẩn được mô tả bằng cách liệt kê một dãy đối thoại (*dialogue elements*) hoặc dãy tương tác giữa actor và hệ thống. Khi cuộc đối thoại này hoàn thành, thì actor đã đạt được mục đích dự định. Tiến trình chuẩn của “Request a Chemical” use case đưa đến kết quả là đề nghị mua hóa chất của người dùng được chuyển cho nhà cung cấp bên ngoài.

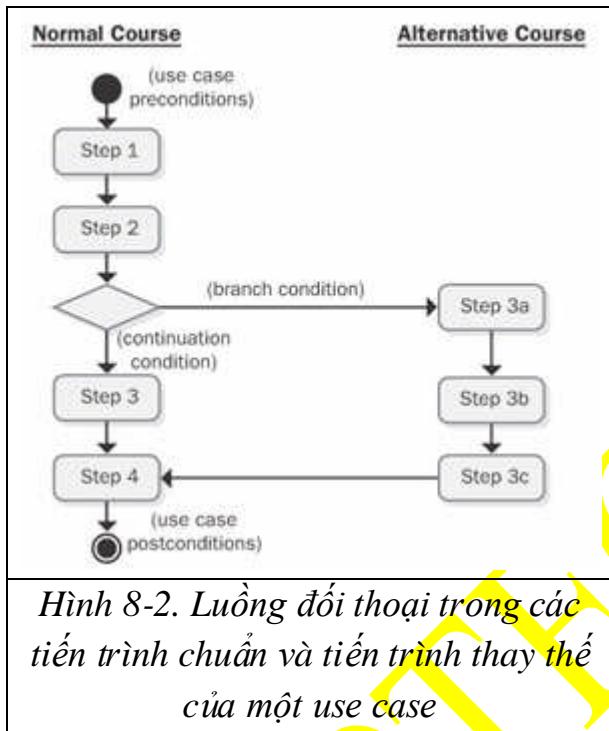
Hình 8-1 thể hiện một sơ đồ use case của “Request a Chemical”, sử dụng ngôn ngữ ký pháp UML (Booch, Rumbaugh, Jacobson 1999).





Hình 8-1. “Request a chemical” use case trong sơ đồ use case chung của CTS

Các kịch bản khác trong use case được mô tả là các *tiến trình thay thế* (*alternative courses*). Các tiến trình thay thế cũng nhằm hoàn thành tác vụ (tasks) nhưng chúng được dùng để thay thế tiến trình chuẩn khi một số điều kiện xảy ra khiến không thể thực hiện được tiến trình chuẩn. Tiến trình chuẩn có thể tách thành một tiến trình thay thế tại một điểm quyết định nào đó trên dây đối thoại (dialogue sequence), và nhập lại vào tiến trình chuẩn sau. Một tiến trình thay thế cho “Request a Chemical” use case là “Request a Chemical from the Chemical Stockroom”. Mặc dù người dùng gửi các đề nghị hóa chất theo cả 2 tiến trình chuẩn và thay thế, nhưng các hành động diễn ra giữa người dùng và hệ thống sẽ khác nhau ở một số chi tiết. Trong tiến trình thay thế, người dùng sẽ vẫn đặc tả hóa chất mong muốn nhưng họ có thể lựa chọn từ các hóa chất đã có sẵn trong các hộp ở trong kho.



*Hình 8-2. Luồng đối thoại trong các
tiến trình chuẩn và tiến trình thay thế
của một use case*

Một số các bước trong một tiến trình thay thế cũng tương tự các bước trong tiến trình chuẩn, nghĩa là cũng phải hoàn thành mục tiêu như các bước trong tiến trình chuẩn (Xem Hình 8-2). Đôi khi sẽ thích hợp hơn nếu *mở rộng* (*extend*) tiến trình chuẩn bằng cách thêm vào tiến trình chuẩn một use case định nghĩa một tiến trình thay thế. Use case mở rộng phải là một use case hoàn chỉnh (complete use case) và có thể được thực hiện riêng rẽ. Hình 8-1 thể hiện use case “See Available Stockroom Containers” mở rộng use case “Request a Chemical”. Sự mở rộng này dẫn tới tiến trình thay thế “Request a Chemical from the Chemical Stockroom”.

Một số use cases có thể dùng chung một số tính năng. Để tránh phải làm 2 lần, bạn có thể định nghĩa một use case riêng biệt (separate use case) chứa tính năng chung đó và khi cần sử dụng tính năng chung thì các use cases khác *kèm* (*include*) use case chung đó vào. Việc này cũng giống như gọi một chương trình con dùng chung trong một ngôn ngữ lập trình. Included use case đóng vai trò quan trọng để hoàn thành tác vụ (tasks), trong khi đó một use case mở rộng một use case khác là một lựa chọn (optional) để tiến trình chuẩn có thể diễn ra theo một cách riêng (Rumbaugh 1994). Một ví dụ, “Request a Chemical” use case kèm một use case gọi là “Enter Change Number” như trong Hình 8-1.

Các điều kiện khiến cho tác vụ (tasks) không thể được hoàn thành trọn vẹn được gọi là các *loại trừ* (*exceptions*), đôi khi chúng được coi như một kiểu tiền trìn thay thế. Khi định nghĩa các use cases thì điều quan trọng là mô tả các exception paths, do chúng thể hiện tầm nhìn của người dùng (user's vision) về việc hệ thống có hành vi như thế nào trong một số điều kiện cụ thể. Một exception của "Request a Chemical" use case là "Chemical is Not Commercially Available". Nếu bạn không tài liệu hóa các exceptions, thì người phát triển sẽ vượt quá các giới hạn khi thiết kế và xây dựng hệ thống, chúng có thể khiến hệ thống bị sự cố khi phải đối mặt với một điều kiện loại trừ.

2. XÁC ĐỊNH VÀ TÀI LIỆU HÓA USE CASES (IDENTIFYING AND DOCUMENTING USE CASES)

Bạn có thể xác định các use cases bằng cách sử dụng một số cách tiếp cận (Ham 1998, Larman 1998):

- Đầu tiên cần xác định các actors và vai trò của họ, sau đó xác định các quy trình nghiệp vụ (business processes) mà mỗi actor tham gia, từ đó sẽ lộ ra các use cases.
- Xác định các sự kiện bên ngoài mà hệ thống cần đáp ứng, tiếp sau xác định sự liên quan của các actors và các use cases cụ thể.
- Diễn giải các quy trình nghiệp vụ (business processes) hoặc các hoạt động hàng ngày dưới dạng các kịch bản cụ thể, dẫn xuất các use cases từ các kịch bản, xác định các actors liên quan đến mỗi use case.
- Dẫn xuất các use cases có thể có từ các phát biểu yêu cầu chức năng. Nếu một yêu cầu bất kỳ mà không song hành với một use case thì có nghĩa bạn chưa thật sự hiểu các yêu cầu.

Do các use cases biểu diễn các yêu cầu người dùng (user requirements), nên bạn phải thu thập chúng trực tiếp từ đại diện của các lớp người dùng khác nhau. Các nhà phân tích của dự án Chemical Tracking System tổ chức các hội thảo suy luận use cases với đại diện của người dùng. Để tổ chức các hội thảo, mỗi nhà phân tích cần đề nghị người dùng suy nghĩ về các tác vụ (tasks) hoặc quy trình nghiệp vụ (business processes) mà họ thấy cần thực hiện với hệ thống mới. Mỗi trong số các tác vụ (tasks) đó có thể trở thành một candidate use case (use case ứng cử viên),

mỗi use case được đánh số và đặt tên ngắn gọn thể hiện nghiệp vụ của nó, ví như “Request a Chemical”, hoặc “Print the Safety Datasheet for a Chemical”.

Hình 8-3 minh họa template của một phiên bản đơn giản của “Request a Chemical” use case.

Use case ID: USE CASES-5					
Tên use case: Request a Chemical					
Người tạo: Tim	Người cập nhật lần cuối: Janice				
Ngày tạo: 4/10	Ngày cập nhật lần cuối: 27/10				
Actor: Requester					
<p>Mô tả: Requester đặc tả hóa chất đề nghị, hoặc nhập chemical ID number của hóa chất đó, hoặc nhập cấu trúc của hóa chất từ một công cụ vẽ cấu trúc hoá chất. Hệ thống có thể đáp ứng đề nghị hoặc bằng cách đưa ra cho Requester một hộp hoá chất mới, hoặc một hộp chứa hoá chất dùng rồi mà trong kho đã có sẵn hoặc một thông báo để Requester tự đặt hàng từ một nhà cung cấp bên ngoài.</p>					
<p>Tiền tình huống (Preconditions):</p> <ol style="list-style-type: none"> 1. User ID được cấp quyền. 2. CSDL kho hóa chất được trực tuyến. 					
<p>Hậu tình huống (Postconditions):</p> <ol style="list-style-type: none"> 1. Các đề nghị được lưu trữ đầy đủ trong Chemical Tracking System 2. Đề nghị được gửi qua email tới Chemical Stockroom hoặc Bộ phận mua sắm để thực hiện. 					
Ưu tiên: Cao					
Tần suất sử dụng: Xấp xỉ 5 lần 1 tuần trên mỗi nhà hoá học, 100 lần mỗi tuần trên mỗi nhân viên quản lý kho hóa chất.					
Tiến trình chuẩn: 5.0 Request a Chemical from a Vendor					
<table border="1"> <thead> <tr> <th>Hành động của Actor</th><th>Đáp ứng của hệ thống</th></tr> </thead> <tbody> <tr> <td> 1. Nhập Chemical ID number hoặc tên của file chứa cấu trúc hoá học 4. Xác định nhà cung cấp (tiếp tục) hoặc kho hoá chất </td><td> 2. Kiểm tra tính hợp lệ của Chemical ID 3. Hỏi Requester là muốn đặt hàng từ một nhà cung cấp mới hoặc muốn một hộp hoá chất từ kho 5. <tiếp tục đói thoại cho đến khi đề nghị </td></tr> </tbody> </table>		Hành động của Actor	Đáp ứng của hệ thống	1. Nhập Chemical ID number hoặc tên của file chứa cấu trúc hoá học 4. Xác định nhà cung cấp (tiếp tục) hoặc kho hoá chất	2. Kiểm tra tính hợp lệ của Chemical ID 3. Hỏi Requester là muốn đặt hàng từ một nhà cung cấp mới hoặc muốn một hộp hoá chất từ kho 5. <tiếp tục đói thoại cho đến khi đề nghị
Hành động của Actor	Đáp ứng của hệ thống				
1. Nhập Chemical ID number hoặc tên của file chứa cấu trúc hoá học 4. Xác định nhà cung cấp (tiếp tục) hoặc kho hoá chất	2. Kiểm tra tính hợp lệ của Chemical ID 3. Hỏi Requester là muốn đặt hàng từ một nhà cung cấp mới hoặc muốn một hộp hoá chất từ kho 5. <tiếp tục đói thoại cho đến khi đề nghị				

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

	được thực hiện>
Tiến trình thay thế: 5.1 Request a Chemical from the Chemical Stockroom (rẽ nhánh kể từ 5.0.4)	
Hành động của Actor	Đáp ứng của hệ thống
<p>2. Tùy chọn (optionally), yêu cầu cho xem tình hình (history) của bất kỳ hộp hoá chất nào</p> <p>3. Chọn một hộp cự thể hoặc yêu cầu đặt hàng từ một nhà cung cấp</p>	<p>1. Hiển thị một danh sách các hộp chứa hóa chất mong muốn còn ở trong kho.</p>
Các loại trừ: 5.E.1 Chemical Is Not Commercially Available	
Hành động của Actor	Đáp ứng của hệ thống
3. Đề nghị một hoá chất khác.	<p>1. Hiển thị message: Không Nhà cung cấp.</p> <p>2. Hỏi Requester liệu có đề nghị một hoá chất khác hoặc thoát khỏi chương trình.</p> <p>4. Bắt đầu lại Tiến trình Chuẩn.</p>
Includes: UC-12 Enter Change Number	
Yêu cầu đặc biệt: Hệ thống có thể nhập một cấu trúc hoá học theo một khuôn dạng đã được mã hoá chuẩn từ bất cứ thiết bị vẽ cấu trúc hoá học nào.	
Giả định: Cấu trúc hoá học được nhập vào hệ thống được giả định hợp lệ	
Ghi chú: Tim sẽ tìm hiểu xem liệu sự chấp thuận của cấp quản lý có cần thiết khi một hoá chất trên mức nguy hiểm cấp 1 được đề nghị hay không. Hạn ngày 4/11 (Due date).	

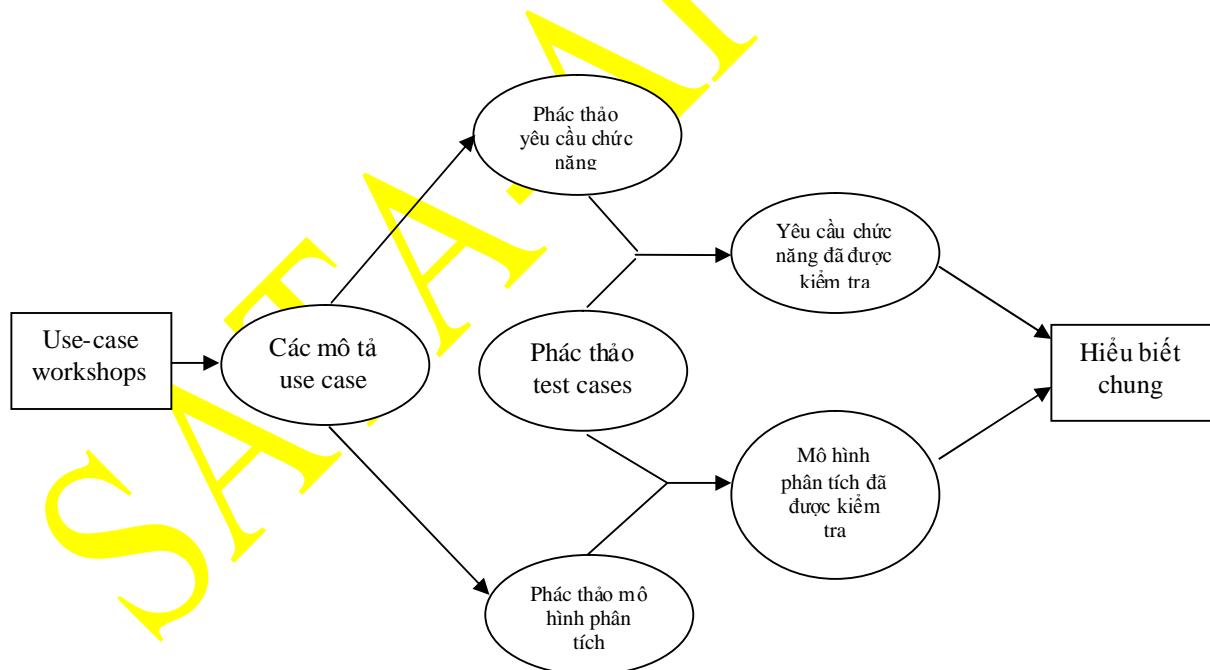
Những người tham gia bắt đầu bằng xác định actor của use case. Tiếp theo, họ định nghĩa **bất cứ** **tình huống** nào cần được đáp ứng để thực hiện use case, các **hậu tình huống** mô tả trạng thái của hệ thống sau khi use case đã được thực hiện. Tần suất ước tính của use case giúp định hình yêu cầu về công suất của hệ thống. Tiếp theo, nhà phân tích đề nghị những người tham gia mường tượng họ sẽ tương tác với hệ thống như thế nào để thực hiện tác vụ. Dãy đối thoại thể hiện thông qua các hành động của actor và đáp ứng của hệ thống tạo thành một flow được gọi là **tiến trình chuẩn** của các sự kiện. Mặc dù mỗi người tham gia mường tượng khác nhau về giao diện người dùng và cơ chế tương tác với hệ thống nhưng khi làm việc chung thì họ sẽ đạt được một cái nhìn chung về quá trình đối thoại actor-system.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Nhóm suy luận cũng phát triển các đối thoại tương tự cho các tiến trình thay thế và loại trừ mà họ đã xác định. Nhiều điều kiện loại trừ lộ ra khi nhà phân tích hỏi các câu hỏi như: “Điều gì sẽ xảy ra nếu CSDL không trực tuyến tại thời điểm đó?”, hoặc: “Điều gì sẽ xảy ra nếu hóa chất đó không được bán trên thị trường?”.

Sau khi một use case đều đã được duyệt và không thấy các biến động phụ (additional variations), các loại trừ (exception), hoặc các chi tiết khác được đề xuất thì những người tham gia nhóm suy luận sẽ chuyển sang use case khác. Họ không cần thiết phải duyệt hết tất cả các use cases ngay trong một phiên làm việc marathon, họ có thể lập kế hoạch để làm dần dần, lặp lại các phiên soát xét và làm mịn, phác thảo dần các chi tiết của yêu cầu chức năng (functional requirements) và các giao diện người dùng có thể có.

Hình 8-4 thể hiện một chuỗi các sự kiện liên quan đến phiên suy luận use case (use-case elicitation workshop) và các hoạt động tiếp theo.



HÌNH 8-4. Cách tiếp cận suy luận use cases

Sau mỗi phiên suy luận, nhà phân tích mô tả các use cases và bắt đầu dẫn xuất các yêu cầu chức năng (functional requirements) từ use cases. Một số use cases đã có các thông tin rõ ràng như “Mỗi đề nghị đã được chấp thuận cần phải được định danh bởi một số duy nhất được hệ thống sinh ra”. Một số khác lại khá tinh tế, không rõ ràng, nó biểu diễn chức năng mà người phát triển cần phải thực hiện để người dùng từng bước tương tác với hệ thống. Nhà phân tích tài liệu hóa các yêu cầu chức năng theo một cấu trúc phân cấp, phù hợp với SRS.

Với một số use cases phức tạp, nhà phân tích có thể phải vẽ một số mô hình phân tích như DFD, ERD, state-transition diagrams, object class, interaction diagrams, dialog maps of possible user interface architecture (bản đồ đối thoại về kiến trúc giao diện người dùng có thể có). Các mô hình đó tạo ra các góc nhìn khác nhau về yêu cầu, chúng có thể để lộ các thiếu sót, các nhập nhằng, sự thiếu nhất quán mà ta không dễ phát hiện khi các yêu cầu ở dưới dạng chữ viết. Chương 10 mô tả một số mô hình phân tích của Chemical Tracking System.

Trong một ngày sau phiên suy luận, nhà phân tích chuyển các mô tả use cases và các yêu cầu chức năng (functional requirements) liên quan cho những người tham gia phiên suy luận để chuẩn bị cho phiên suy luận tiếp theo. Việc soát xét thường xuyên sẽ để lộ ra các lỗi, các tiến trình thay thế chưa được phát hiện, các loại trừ (exception), các yêu cầu chức năng (functional requirements) không đúng, các bước bị thiếu trong đối thoại actor-system. Cách tiếp cận use case rất có lợi để cải tiến chất lượng yêu cầu qua việc soát xét dần dần, nhưng đừng tổ chức các phiên soát xét quá gần nhau vì bạn cần để trí não mình ngấm dần các hiểu biết mà bạn thu được trong phiên trước đã.

Ngay từ sớm trong quy trình phát triển yêu cầu, các ý tưởng về test cases (tình huống kiểm thử) đã phải được đặt ra. Các test cases giúp nhóm dự án có một hiểu biết chung, sáng sủa về việc hành vi của hệ thống cần phải được mô tả ra sao thành các kịch bản sử dụng cụ thể. Các test cases giúp các nhà phân tích kiểm tra xem liệu họ đã nắm bắt (capture) tất cả các yêu cầu chức năng (functional requirements) cần thiết đủ để sinh ra các hành vi hệ thống đã được tài liệu hóa trong các test cases. Họ cũng sử dụng các test cases để xác định liệu các mô hình phân tích đã đầy đủ, chính xác và nhất quán với các yêu cầu chức năng (functional requirements).

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

requirements) hay chưa. Chương 14 thảo luận chi tiết về việc sinh ra các test cases từ các yêu cầu.

3. USE CASES VÀ YÊU CẦU CHỨC NĂNG (USE CASES AND FUNCTIONAL REQUIREMENTS)

Các mô tả use cases tự nó không cung cấp cho các nhà phát triển đủ đến mức chi tiết các tính năng để họ xây dựng hệ thống. Nếu bạn dùng phát triển yêu cầu tại mức yêu cầu người dùng (user requirements), bạn sẽ thấy khi xây dựng hệ thống, nhà phát triển sẽ phải hỏi nhiều câu hỏi để lấp vào các chỗ thông tin còn thiếu. Để giảm bớt sự không chắc chắn này, bạn cần phải biến các use cases thành các yêu cầu chức năng (functional requirements) chi tiết (Arlow 1998).

Mỗi use case sẽ dẫn tới một số yêu cầu chức năng (functional requirements) tạo điều kiện cho actor thực thi tác vụ (task) tương ứng, và một số use cases có thể lại sử dụng chung một yêu cầu chức năng. Ví dụ, nếu 5 use cases đòi hỏi định danh của người dùng được xác thực thì bạn không cần phải viết 5 khối mã nguồn khác nhau để thực hiện việc đó. Bạn có thể tài liệu hóa các yêu cầu chức năng (functional requirements) liên quan đến một use case bằng nhiều cách. Cách tiếp cận mà bạn sử dụng phụ thuộc vào việc liệu bạn có mong đợi nhóm dự án thực hiện quá trình thiết kế, xây dựng, kiểm thử dựa trên các use – case documents, dựa trên SRS, hoặc dựa trên sự kết hợp của 2 tài liệu này. Không có phương pháp nào là hoàn hảo, hãy chọn cách tiếp cận phù hợp nhất với cách bạn muốn tài liệu hóa và quản lý các yêu cầu phần mềm. Tránh các thông tin lặp lại ở nhiều nơi của tài liệu, sự dư thừa sẽ khiến việc quản lý yêu cầu trở nên khó khăn.

Chỉ sử dụng use cases (Use cases Only)

Bạn đưa các yêu cầu chức năng (functional requirements) vào use-case description, mặc dù bạn vẫn có thể cần một SRS riêng để chứa các thông tin không liên quan đến các use cases cụ thể. Bạn cần tham chiếu chéo các yêu cầu chức năng (functional requirements) chung giữa nhiều use cases. Một cách để làm điều đó là dùng quan hệ “include” đã thảo luận trước đây.

Use cases và SRS

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Một lựa chọn khác là giới hạn use-case description trong phần yêu cầu người dùng (user requirements) và viết các yêu cầu chức năng (functional requirements) mà bạn dẫn từ mỗi use case trong SRS. Theo cách này, bạn cần thiết lập khả năng liên kết giữa các use cases và các yêu cầu chức năng (functional requirements) liên quan. Cách tốt nhất để thực hiện việc này là lưu trữ tất cả các use cases và yêu cầu chức năng (functional requirements) trong một CSDL hoặc một công cụ quản lý yêu cầu có thể cho phép bạn định nghĩa các liên kết như vậy (Xem Chương 19).

Chỉ sử dụng SRS (SRS only)

Cách tiếp cận thứ ba là tổ chức SRS theo use case, cùng với các use case description và mô tả yêu cầu chức năng (functional requirements description). Theo cách này bạn sẽ không phải viết riêng các use-case documents chi tiết; tuy nhiên, bạn cần chỉ rõ các yêu cầu chức năng lặp lại (duplicated functional requirements) hoặc xác định mỗi yêu cầu chức năng (functional requirements) duy nhất ngay từ trước và tham chiếu tới các báo cáo gốc của nó ngay khi yêu cầu xuất hiện lại trong một use case khác.

4. LỢI ÍCH CỦA USE CASES (BENEFITS OF USE CASES)

Sức mạnh của cách tiếp cận use-case là suy luận yêu cầu theo quan điểm hướng người dùng (user-centric) và hướng tác vụ (task – centric). Người dùng sẽ có cái nhìn sáng sủa hơn về những gì mà hệ thống sẽ cung cấp cho họ so với cách tiếp cận hướng chức năng (function – centric). Đại diện của khách hàng trong một số dự án phát triển Web thấy rằng cách tiếp cận use-case thực sự giúp họ phân loại các ý tưởng cần phải làm đối với mỗi khách hàng. Use cases giúp các nhà phân tích và các nhà phát triển hiểu cả nghiệp vụ của người dùng (user's business) và miền ứng dụng (problem domain) của bài toán. Suy nghĩ kỹ càng về chuỗi đối thoại actor-system có thể揭露 ra các nhập nhằng và mơ hồ ngay từ sớm khi phát triển dự án, và cũng sinh ra được các test cases từ các use cases.

Những gì còn nhập nhằng sẽ không bao giờ được đưa vào mã nguồn. Với use cases, các yêu cầu chức năng (functional requirements) được dẫn ra để người dùng thực hiện tác vụ (tasks) của họ. Kỹ thuật use-case ngăn ngừa tính năng “mồ côi” – các chức năng có vẻ như là cần thiết khi suy luận nhưng không có ai sử dụng vì chúng chẳng liên quan gì đến các tác vụ (tasks) của họ.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Kỹ thuật use-case còn nhiều lợi ích khác nữa. Cách tiếp cận use-case để lộ ra các domain objects và công việc nó đảm đương. Nhà phát triển sử dụng cách thiết kế hướng đối tượng có thể đưa các use cases và mô hình đối tượng. Hơn nữa, khi các quy trình nghiệp vụ (business processes) thay đổi theo thời gian, các tác vụ (tasks) được mã hoá trong các use cases cụ thể sẽ thay đổi. Nếu bạn đã thiết lập vết giữa các yêu cầu chức năng (functional requirements), thiết kế, mã nguồn, và các test cases thì bạn có thể quay trở lại use cases gốc - tiếng nói của khách hàng - để đánh giá tác động của thay đổi trên toàn bộ dự án.

5. TRÁNH CÁC BẪY KHI SỬ DỤNG USE CASES (USE CASE TRAPS TO AVOID)

Đừng sa vào những cái bẫy dưới đây khi ứng dụng cách tiếp cận use-case:

- *Quá nhiều use cases (Too many use cases)*. Nếu bạn tự thấy mình bị vướng vào một use-case explosion (sự bùng nổ use-case) thì bạn không nên viết các use cases ở mức trừu tượng thích hợp. Đừng tạo ra một use case riêng cho mỗi kịch bản. Thay vì vậy, hãy gom tiến trình chuẩn (normal course), các tiến trình thay thế (alternative courses), các loại trừ (exceptions) thành các kịch bản (scenarios) trong một use case duy nhất. Đừng xử lý mỗi bước trong chuỗi tương tác giữa actor và system như một use case riêng. Mỗi use case cần mô tả một tác vụ (tasks) độc lập. Thông thường, bạn sẽ có nhiều, rất nhiều use cases so với số yêu cầu kinh doanh (business requirements), nhưng sẽ có nhiều, rất nhiều yêu cầu chức năng (functional requirements) so với số use cases. Mỗi use case cần phải kể một câu chuyện về một cách mà người dùng phải tương tác với hệ thống để đạt được một mục đích cụ thể.
- *Trùng lắp trong các use cases (Duplication across use cases)*. Nếu chức năng đã định xuất hiện trong nhiều use cases thì sẽ gây sinh nguy cơ thực thi một chức năng nhiều lần. Để tránh sự lặp lại đó, hãy sử dụng quan hệ “include” để nhiều use cases sử dụng chung một chức năng.
- *Thiết kế giao diện người dùng kèm theo use cases (User interface design included in the use cases)*. Các use cases cần phải tập trung vào cái mà người dùng cần phải làm với hệ thống, chứ không phải là màn hình sẽ trông như thế nào. Use cases tập trung tìm hiểu các ý tưởng về đối thoại giữa actors và system, các chi tiết về thiết kế giao diện người dùng sẽ đến giai

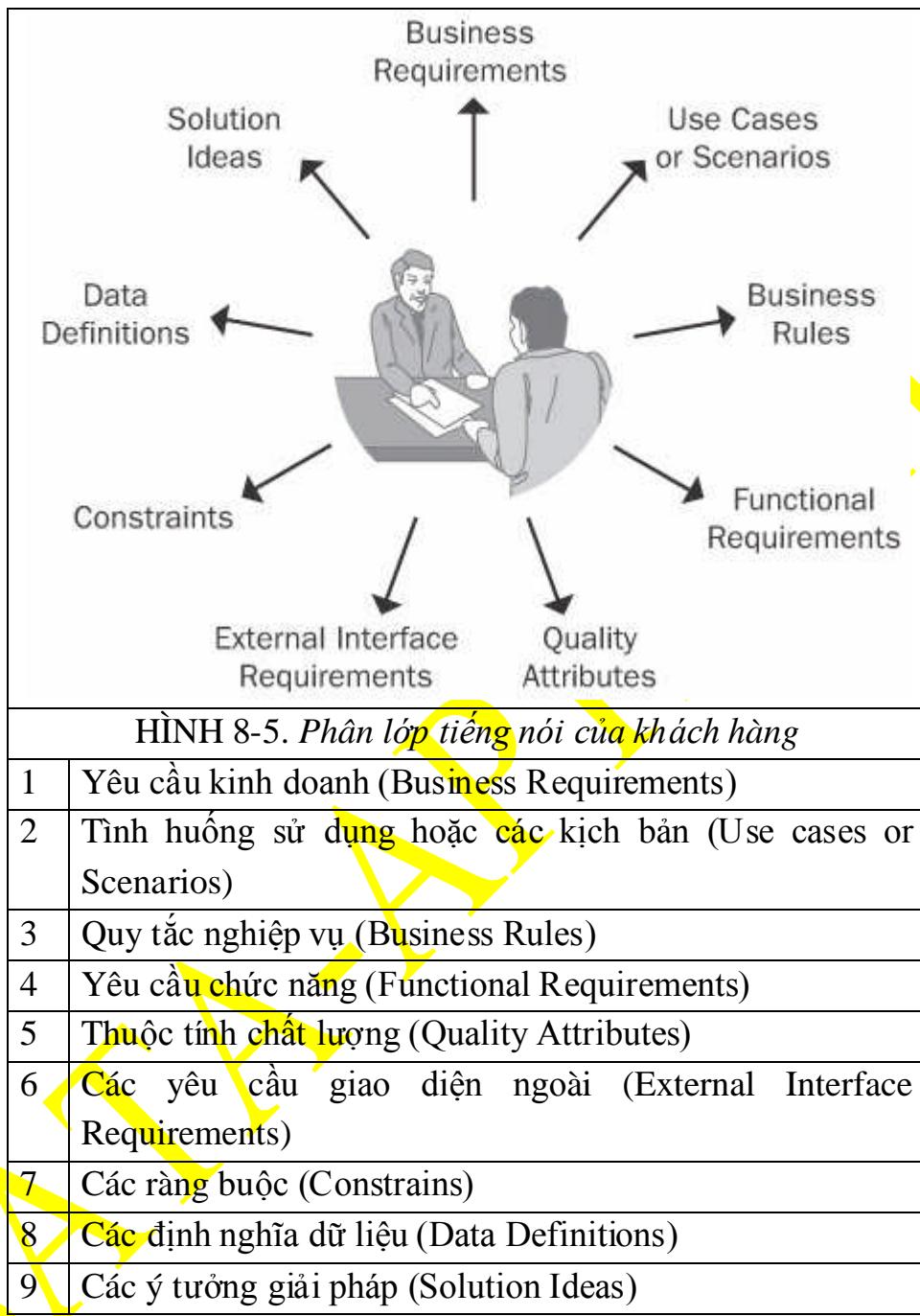
Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

đoạn thiết kế. Mọi phác thảo màn hình hoặc các bản đồ kiến trúc giao diện đều chỉ được sử dụng để làm trực quan hóa các tương tác actor-system, chứ không phải là các đặc tả thiết kế vững chắc.

- *Đưa các định nghĩa dữ liệu vào use cases (Including data definitions in use cases).* Tôi đã thấy các use cases chứa các định nghĩa về các data items và data structures, các data type, độ dài, khuôn dạng và các giá trị hợp lệ. Cách tiếp cận này gây khó khăn cho những người tham gia dự án khi tìm kiếm các định nghĩa mà họ cần, cũng như sự không rõ ràng nếu use-case description bao chứa luôn cả định nghĩa dữ liệu. Cách làm này có thể sinh ra các định nghĩa dư thừa, chúng rất dễ mất đi sự nhất quán khi một thể hiện (instance) của dữ liệu bị thay đổi còn các thể hiện khác thì không. Hãy tập hợp các định nghĩa dữ liệu của toàn bộ dự án vào data dictionary (sẽ thảo luận trong Chương 9).
- *Có gắn kết mỗi yêu cầu với một use case (Attempting to associate every requirement with a use case).* Trong khi các use cases nắm bắt (capture) các hành vi cơ bản của hệ thống thì bạn sẽ thấy có một số yêu cầu không liên quan gì đến các tác vụ (tasks) của người dùng hoặc các tương tác giữa actors và hệ thống. Bạn vẫn cần một đặc tả yêu cầu riêng để mô tả các yêu cầu phi chức năng, các yêu cầu giao diện ngoài và các yêu cầu chức năng (functional requirements) không dẫn xuất từ một use case.

III. PHÂN LỚP ĐẦU VÀO CỦA KHÁCH HÀNG (CLASSIFYING CUSTOMER INPUT)

Đừng hy vọng các khách hàng của bạn sẽ trình bày với nhà phân tích yêu cầu một danh sách đầy đủ, được tổ chức tốt, thể hiện nhu cầu của họ. Nhà phân tích cần phân lớp các đầu vào của khách hàng sao cho có thể sử dụng thông tin thu được một cách có ích nhất. Hình 8-5 minh họa một số cách phân lớp như vậy.



Yêu cầu kinh doanh (Business Requirements)

Bất cứ mô tả gì về lợi ích thương mại, thị trường, hoặc lợi ích kinh doanh mà khách hàng hoặc tổ chức phát triển sản phẩm có thể có được từ sản phẩm thì có khả năng đó chính là yêu cầu kinh doanh (business requirements). Lắng nghe các báo cáo về các con số mà người dùng trực tiếp hoặc gián tiếp của sản phẩm phản

mềm nêu lên như “Tăng thị phần lên X%”, “Tiết kiệm \$ Y mỗi năm”, hoặc “Thay thế hệ thống có chi phí bảo trì cao Z”.

Tình huống sử dụng hoặc các kịch bản (Use cases or Scenarios)

Báo cáo chung về mục đích của người dùng hoặc các tác vụ (tasks) mà người dùng cần thực hiện với hệ thống có thể là các use cases, trong khi đó các mô tả tác vụ cụ thể thể hiện các kịch bản thường dùng. Làm việc với khách hàng để tổng quát hóa các tác vụ (tasks) cụ thể thành các use cases. Bạn cũng có thể lượm được các use cases bằng cách đề nghị khách hàng mô tả các hoạt động theo luồng nghiệp vụ của họ (business workflow activities).

Quy tắc nghiệp vụ (Business Rules)

Khi một khách hàng nói rằng một hoạt động nào đó có thể được thực hiện chỉ bởi các cá nhân nào đó hoặc vai trò nào đó, dưới những điều kiện nào đó thì có nghĩa anh ta đang mô tả một quy tắc nghiệp vụ (business rule), ví dụ, “Một nhà hoá học có thể đặt hàng một hoá chất thuộc loại nguy hiểm cấp 1 nếu anh ta đã được đào tạo về cách thức làm việc với hoá chất nguy hiểm”. Quy tắc nghiệp vụ (business rule) là các nguyên tắc vận hành một quy trình nghiệp vụ (business processes). Bạn có thể dẫn xuất ra một số yêu cầu chức năng (functional requirements) để làm quy tắc có hiệu lực, ví như tạo ra một CSDL lưu trữ thông tin đào tạo có thể truy nhập bởi Chemical Tracking System. Như vậy, các quy tắc nghiệp vụ không phải là các yêu cầu chức năng (functional requirements).

Yêu cầu chức năng (Functional Requirements)

Một báo cáo của khách hàng bắt đầu với “Người dùng phải <thực hiện một chức năng nào đó>”, hoặc “Hệ thống phải <biểu hiện một số hành vi>” thì có khả năng đó chính là một yêu cầu chức năng (functional requirements). Các yêu cầu chức năng (functional requirements) mô tả các hành vi có thể quan sát của hệ thống, thường là trong bối cảnh của hoạt động tương tác actor – system. Các yêu cầu chức năng (functional requirements) định nghĩa cái hệ thống phải làm, chúng tạo nên phần chủ yếu của SRS. Nhà phân tích cần phải giải thích cho một số người hiểu tại sao hệ thống “phải” thực hiện các chức năng đó. Các yêu cầu chức năng (functional requirements) được đề xuất đôi khi phản ánh các quy trình nghiệp vụ

(business processes) không hiệu quả hoặc đã lỗi thời và không nên đưa vào hệ thống mới.

Thuộc tính chất lượng (Quality Attributes)

Các báo cáo xác định hệ thống thực hiện một số hành vi tốt như thế nào hoặc cho phép người dùng thực hiện một hành động nào đó chính là các thuộc tính chất lượng, một kiểu yêu cầu phi chức năng. Hãy lắng nghe các từ mô tả các đặc tính mà người dùng kỳ vọng ở hệ thống như: nhanh, dễ dàng, trực quan, thân thiện, vững chắc, đáng tin cậy, an ninh, hiệu quả (fast, easy, intuitive, user-friendly, robust, reliable, secure, efficient). Bạn sẽ phải làm việc với người dùng để định nghĩa chính xác cái mà họ ngũ ý do các thuộc tính này rất dễ được sử dụng theo ý chủ quan và nội dung của nó dễ nhập nhằng, xem thêm Chương 11.

Các yêu cầu giao diện ngoài (External Interface Requirements)

Lớp yêu cầu này mô tả kết nối giữa hệ thống của bạn và phần còn lại của thế giới bên ngoài hệ thống (connections between your system and the rest of the universe). SRS cần phải bao gồm mục nói về các yêu cầu đó, chúng mô tả các giao diện và cách thức truyền thông với người dùng, với phần cứng, và với các hệ thống khác. Các câu sau đây mà khách hàng nói thì là ngũ ý về giao diện ngoài:

- “Phải đọc các tín hiệu từ <thiết bị nào đó>”
- “Phải gửi các messages tới <một hệ thống khác nào đó>”
- “Phải đọc các tệp theo <khuôn dạng nào đó>”
- “Phải kiểm soát <một thiết bị phần cứng nào đó>”

Các ràng buộc (Constraints)

Các ràng buộc là các điều kiện nhằm giới hạn một cách hợp lệ các lựa chọn có thể đổi với người thiết kế và người lập trình. Chúng là thể hiện của một kiểu yêu cầu phi chức năng khác mà bạn cần viết trong SRS. Hãy cố gắng thuyết phục khách hàng đưa ra các ràng buộc không cần thiết vì chúng có thể ngăn cản việc tạo ra một giải pháp tốt nhất. Các ràng buộc không cần thiết sẽ làm giảm khả năng sử dụng các software components có sẵn để tạo nên giải pháp. Một số ràng buộc lại có thể khiến đáp ứng tốt hơn các thuộc tính chất lượng. Một ví dụ, để cải tiến khả năng uyển chuyển (portability – khả năng hệ thống chạy được trên nhiều nền tảng khác nhau) người ta chỉ sử dụng duy nhất các câu lệnh tiêu chuẩn (standard commands). Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

của một ngôn ngữ lập trình, mà không cho phép mở rộng để sử dụng lệnh từ các nhà cung cấp công cụ khác. Các câu sau thể hiện ý khách hàng trong việc đưa ra các ràng buộc:

- “Phải sử dụng <một hệ QTCSDL hoặc một ngôn ngữ lập trình nào đó>”
- “Không thể đòi hỏi hơn <một lượng bộ nhớ nào đó>”
- “Cần phải nhất quán với <một ứng dụng nào đó>”

Định nghĩa dữ liệu (Data definitions)

Bất cứ khi nào khách hàng mô tả khuôn dạng, các giá trị được phép, các giá trị ngầm định cho một data item hoặc cấu tạo của một cấu trúc dữ liệu nghiệp vụ phức hợp thì nghĩa là họ đang đưa ra một định nghĩa dữ liệu. Ví dụ, “ZIP code chứa 5 chữ số, tiếp theo là một dấu nối và một nhóm tùy chọn 4 chữ số, ngầm định là 0000” là một định nghĩa dữ liệu. Hãy tập hợp chúng lại trong một từ điển dữ liệu để cung cấp một nguồn tham chiếu chính mà những người tham gia dự án có thể sử dụng trong suốt tiến trình phát triển và bảo trì sản phẩm. (Xem Chương 9).

Các ý tưởng giải pháp (Solution Ideas)

Nếu khách hàng mô tả cụ thể một cách tương tác với hệ thống nhằm thực hiện một hành động nào đó (ví dụ, “Người dùng lựa chọn mục mà anh ta muốn từ một dropdown list”), thì có nghĩa bạn đang nghe về đề xuất của một giải pháp, đó không phải là một yêu cầu. Các giải pháp đề xuất có thể làm rối trí nhóm suy luận. Khi suy luận yêu cầu, hãy tập trung vào cái hệ thống cần làm thay vì hệ thống làm cái đó như thế nào. Hãy suy nghĩ xem tại sao khách hàng đề xuất một cách thức cài đặt cụ thể, hãy giúp họ hiểu cả nhu cầu thực và các kỳ vọng ngầm của khách hàng về hệ thống sẽ làm việc như thế nào.

IV. MỘT SỐ LUU Ý THẬN TRỌNG KHI SUY LUẬN (SOME CAUTIONS ABOUT ELICITATION)

Trong quá trình suy luận yêu cầu, có thể bạn sẽ thấy phạm vi sản phẩm được định nghĩa chưa đúng, hoặc quá lớn hoặc quá nhỏ (Christel and Kang 1992). Nếu phạm vi quá lớn thì bạn sẽ thu thập được nhiều yêu cầu hơn là nhu cầu sử dụng thật sự, quy trình suy luận sẽ kéo dài một cách chán ngắt. Nếu phạm vi được định nghĩa quá nhỏ thì sản phẩm làm ra sẽ không đáp ứng được nhu cầu của khách hàng. Suy luận yêu cầu thường dẫn tới việc chỉnh sửa tầm nhìn và phạm vi (vision and scope) Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

của dự án, nhưng chúng cũng có thể gây ra những thay đổi ảnh hưởng rất nhiều đến dự án.

Thường các yêu cầu được định nghĩa là *cái* hệ thống phải làm, trong khi đó *làm thế nào* lại được dành cho công việc thiết kế. Về mặt lý thuyết là như vậy nhưng trên thực tế, đó là một sự đơn giản hóa quá mức. Suy luận yêu cầu đúng thực là tập trung vào *cái gì*, nhưng có một vùng xám giữa phân tích và thiết kế. Bạn có thể sử dụng các giả thiết *làm thế nào* để làm sáng tỏ và làm minh sự hiểu biết của bạn về cái mà người dùng cần. Các mô hình phân tích, các phác thảo màn hình, các nguyên mẫu sẽ giúp các ý tưởng được bàn thảo trong suy luận yêu cầu trở nên hữu hình và đó cũng là cách để bạn dễ dàng nhìn thấy các lỗi và thiếu sót của các ý tưởng đó. Trình diễn (view) các mô hình và các màn hình chương trình dự kiến mà bạn sinh ra trong quá trình phát triển yêu cầu sẽ là các gợi ý làm quá trình truyền thông trở nên sáng sủa hơn, đó không phải là các ràng buộc về các lựa chọn của người thiết kế.

Các hội thảo suy luận yêu cầu mà có quá nhiều người tham gia thì sẽ làm chậm quá trình phát triển. Bạn đồng nghiệp Debbie của tôi trước đây đã bị thất bại khi tổ chức các hội thảo suy luận use cases do nguyên nhân đó. Mười hai người tham gia đã mở rộng các thảo luận về các chi tiết không cần thiết và đã dành một lượng thời gian đáng kể để bàn về việc các use cases phải hoạt động như thế nào. Sau đó mọi việc đã tiến triển tốt hơn khi cô rút số người tham gia xuống còn 6 người, đại diện cho các vai trò chính như khách hàng, kiến trúc hệ thống, người phát triển, người thiết kế.

Ngược lại, thu thập các yêu cầu từ quá ít những người đại diện, hoặc nghe tiếng nói chỉ từ những khách hàng ít ảnh hưởng, cũng có thể dẫn tới vấn đề khó khăn. Vì vậy, cân bằng được các tiếng nói là mục đích cần đạt tới của người trưởng dự án.

V. BẠN BIẾT LÀM THẾ NÀO NẾU BẠN ĐÃ LÀM RỒI? (HOW DO YOU KNOW WHEN YOU ARE DONE?)

Không có cách nào đơn giản để bạn thu thập yêu cầu, cũng không có cách nào đơn giản để bạn biết khi nào thì nên ngừng quá trình suy luận yêu cầu lại, các gợi ý sau có thể giúp bạn biết khi nào thì nên ngừng:

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- Nếu người dùng không thể nghĩ được thêm bất cứ use cases nào, thì có lẽ bạn nên kết thúc suy luận yêu cầu. Người dùng có khuynh hướng định danh các use cases theo thứ tự giảm dần của tầm quan trọng.
- Nếu người dùng đề xuất các use cases mới, nhưng bạn đã dẫn xuất các yêu cầu chức năng (functional requirements) liên quan từ các use cases khác, thì có lẽ bạn nên kết thúc suy luận yêu cầu. Các use cases mới vừa xuất có thể là các tiến trình thay thế cho các use cases mà bạn đã có từ trước.
- Nếu người dùng bắt đầu nói lại những vấn đề mà bạn và họ đã thảo luận từ các phiên họp trước thì có lẽ bạn nên kết thúc suy luận yêu cầu.
- Nếu các use cases mới được gợi ý hoặc các yêu cầu chức năng (functional requirements) dường như nằm ngoài phạm vi đã định thì có lẽ bạn nên kết thúc suy luận yêu cầu.
- Nếu các yêu cầu mới được đề xuất nằm trong số có độ ưu tiên thấp liên quan đến những yêu cầu mà bạn đã đặc tả thì có lẽ bạn nên kết thúc suy luận yêu cầu.

Trong các dự án trước đây, hãy tạo một danh sách các nhóm chức năng mà hệ thống thường phải có. Bạn có thể tạo ra một checklist về các chức năng phổ biến để cân nhắc mỗi khi kết thúc quy trình suy luận yêu cầu của một dự án mới, ví dụ như viết error logs, sao lưu và khôi phục, báo cáo, in ấn, các tính năng xem trước, ... Sau khi phát triển yêu cầu, hãy so sánh bản checklist với các chức năng mà bạn đã đặc tả. Nếu không thấy sự “cập kẽm” giữa chúng thì có lẽ bạn nên kết thúc suy luận yêu cầu.

Các bước tiếp theo

- Chọn một đoạn nào đó trong một tài liệu ghi nhận tiếng nói của khách hàng trong dự án của bạn hoặc trong SRS. Hãy phân loại mỗi mục trong đoạn đó vào các nhóm (category) trong Hình 8-5. Nếu bạn phát hiện thấy các mục không được viết thành tài liệu đúng đắn như các yêu cầu chức năng (functional requirements) thì hãy quay lại và sửa chữa vị trí tương ứng trên SRS hoặc tài liệu gốc.
- Viết một use case cho dự án hiện tại của bạn, sử dụng use-case template trong Hình 8-3, gồm các các tiến trình thay thế (alternative course) thích

đáng và các loại trừ (exception). Định nghĩa yêu cầu chức năng (functional requirements) cho phép người dùng thực hiện thành công use case này. Kiểm tra xem liệu SRS trong dự án hiện tại của bạn đã chứa tất cả các yêu cầu chức năng (functional requirements) đó.

- Liệt kê các phương pháp suy luận yêu cầu được sử dụng trong dự án của bạn. Những phương pháp nào là tốt? Tại sao? Những phương pháp nào chưa tốt? Tại sao? Hãy xác định các kỹ thuật suy luận yêu cầu mà bạn nghĩ chúng tốt hơn và quyết định ứng dụng chúng cho dự án tiếp theo. Hãy xác định bất cứ rào chắn nào mà bạn phải đương đầu để thực hiện kỹ thuật đó, hãy tập kích nã để vượt qua các rào chắn đó.

SATA-APTECH

CHƯƠNG 9

TÀI LIỆU HÓA CÁC YÊU CẦU

Phát triển yêu cầu tạo ra một sản phẩm cuối là một thỏa thuận được tài liệu hóa giữa khách hàng và nhóm phát triển về sản phẩm cần xây dựng. Thỏa thuận này bao trùm lên cả 3 nhóm yêu cầu: yêu cầu kinh doanh (business requirements), yêu cầu người dùng (user requirements), yêu cầu chức năng (functional requirements). Như chúng ta đã thấy trước đây, **tài liệu tầm nhìn và phạm vi (vision and scope)** **chứa yêu cầu kinh doanh, yêu cầu người dùng thì được thể hiện trong tài liệu use-case**. Bạn cũng phải tài liệu hóa cả yêu cầu chức năng dẫn xuất từ các use cases và các yêu cầu phi chức năng như các thuộc tính chất lượng, các yêu cầu về giao diện ngoài. Trừ khi bạn viết các yêu cầu đó theo một khuôn dạng có cấu trúc và dễ đọc để các stakeholders soát xét và chấp thuận chúng, còn không thì người ta sẽ không dám chắc về cái gì được thỏa thuận.

Bạn có thể tài liệu hóa các yêu cầu phần mềm theo 3 cách:

- Tài liệu dưới dạng chữ viết được trình bày theo một khuôn dạng có cấu trúc và viết bằng ngôn ngữ tự nhiên.
- Các mô hình đồ họa (graphical models) minh họa các quy trình biến đổi, các trạng thái hệ thống và thay đổi giữa các trạng thái đó, các quan hệ dữ liệu, các luồng logic, các lớp đối tượng và quan hệ giữa các lớp đó.
- Các đặc tả hình thức định nghĩa các yêu cầu bằng cách sử dụng ngôn ngữ logic hình thức chính xác của toán học.

Trong khi các đặc tả hình thức mang lại sự chính xác lớn nhất thì lại ít nhà phát triển phần mềm nào, và hầu như không vị khách hàng nào quen thuộc với việc sử dụng các ký pháp hình thức đó. Mặc dù có nhiều hạn chế, nhưng ngôn ngữ tự nhiên vẫn được sử dụng nhiều nhất để viết các tài liệu yêu cầu trong hầu hết các dự án. Một đặc tả yêu cầu phần mềm dựa trên ngôn ngữ tự nhiên chứa các yêu cầu chức năng và yêu cầu phi chức năng cũng đáp ứng hầu hết nhu cầu của các dự án. Các mô hình phân tích đồ họa thêm vào SRS là các góc nhìn khác làm tăng độ chính xác của yêu cầu.

Chương này mô tả mục đích và cấu trúc của SRS, bao gồm cả một đề xuất template cho tài liệu SRS. Các hướng dẫn viết yêu cầu chức năng cũng được đề cập, cùng với một số ví dụ về các báo cáo yêu cầu không hoàn hảo (imperfect) và các đề xuất để cải tiến chúng. Các kỹ thuật mô hình hóa để biểu diễn yêu cầu là chủ đề của Chương 10. Các phương pháp đặc tả hình thức không được mô tả kỹ trong cuốn sách này.

I. ĐẶC TẢ YÊU CẦU PHẦN MỀM (SRS - THE SOFTWARE REQUIREMENTS SPECIFICATION)

Đặc tả yêu cầu phần mềm được hiểu là đặc tả chức năng (functional specification), thỏa thuận yêu cầu (requirements agreement) và đặc tả hệ thống (system specification). SRS là cơ sở cho tất cả các hoạt động tiếp theo như lập kế hoạch dự án, thiết kế, mã hoá, kiểm thử và viết tài liệu trợ giúp người dùng. SRS phải mô tả càng đầy đủ càng tốt các hành vi thể hiện ra bên ngoài mà người dùng có thể quan sát được của hệ thống. Nó không bao gồm các chi tiết về thiết kế, xây dựng, kích thước hoặc quản lý dự án, nó có thể bao hàm các ràng buộc về thiết kế và thi công hệ thống. SRS cần đáp ứng các mục đích sau:

- Khách hàng và bộ phận marketing dựa vào SRS để biết họ có thể kỳ vọng những gì sẽ được chuyển giao.
- Các nhà quản lý dự án bố trí các kế hoạch và ước lượng lịch biểu, nguồn lực (effort), thời gian dành cho sản phẩm dựa trên các mô tả sản phẩm trong SRS.
- Nhóm phát triển dựa vào SRS để hiểu họ cần xây dựng cái gì.
- Nhóm kiểm thử sử dụng các mô tả hành vi trong SRS để dàn ra các kế hoạch kiểm thử (test plans), các tình huống kiểm thử, các thủ tục kiểm thử.
- Các nhân viên hỗ trợ và bảo trì phần mềm tham chiếu SRS để hiểu công việc của họ.
- Nhóm xuất bản (publication group) viết các tài liệu người dùng như sách hướng dẫn sử dụng, các màn hình hỗ trợ dựa trên SRS và thiết kế giao diện người dùng.
- Người đào tạo (về sản phẩm) có thể sử dụng cả SRS và tài liệu người dùng để giúp họ phát triển các tài liệu đào tạo.

Là tài liệu cuối cùng chưa đựng tất cả các yêu cầu về sản phẩm, trong SRS không còn giả định nào nữa cả, mọi thứ đều đã rõ. Nếu bất cứ yêu cầu chức năng (functional requirements) và phi chức năng nào mà không chứa trong SRS thì có nghĩa nó không nằm trong phạm vi của dự án và không ai mong đợi nó sẽ xuất hiện trong sản phẩm.

Không ai nói rằng bạn cần phải viết toàn bộ SRS trước khi bắt đầu thiết kế và xây dựng hệ thống. Bạn có thể tiếp cận đặc tả yêu cầu theo cách lặp, tăng dần. Đó là do một số yếu tố, ví dụ không ai có thể biết hết tất cả các yêu cầu mà mình cần ở sản phẩm, không ai biết các yêu cầu sẽ cố định mà không thay đổi... Tuy nhiên, mỗi dự án cần phải có một thỏa thuận được vạch ranh giới (baselined agreement) cho mỗi một tập hợp yêu cầu sẽ được đưa vào thi công. *Vạch ranh giới* là tiến trình chuyển đổi một SRS đang được phát triển thành một phiên bản SRS được soát xét và chấp thuận (*nghĩa là một phiên bản của hệ thống sẽ được thi công theo SRS đã được vạch ranh giới này - ND*). Thay đổi trong một SRS đã được vạch ranh giới (baselined SRS) phải được thực hiện thông qua quy trình kiểm soát thay đổi đã định nghĩa sẵn của dự án. Sau khi ~~đã~~ vạch ranh giới thì tất cả những người tham gia dự án phải làm việc trên tập hợp yêu cầu đó (*được mô tả trong baselined SRS - ND*) để tránh rối loạn.

Chương 1 đã giới thiệu một số đặc tả của các tài liệu yêu cầu chất lượng cao: đầy đủ, nhất quán, có thể sửa, có thể lần vết. Công việc của bạn là cấu trúc hóa và viết các SRS sao cho người dùng và những người có liên quan khác có thể đọc hiểu SRS, dưới đây là các lời khuyên hữu ích cho việc đó:

- Đánh số các đề mục (section), đề mục con (subsection) và từng yêu cầu riêng phải nhất quán.
- ~~Sắp xếp văn bản có hàng, có lối chữ không để tất cả các dòng đều thẳng tắp một hàng.~~
- Để các khống gian trống (white space) trên tài liệu.
- Sử dụng các dấu hiệu nhấn mạnh (như **chữ đậm**, chữ gạch dưới, *chữ nghiêng*, và các phông chữ kiểu khác).
- Tạo một bảng mục lục và một chỉ dẫn (index) để giúp độc giả tìm nhanh thông tin họ cần.

- Đánh số và gán nhãn tất cả các hình vẽ (figures), các bảng (tables) và có tham chiếu tới chúng theo số đó.
- Sử dụng các tham chiếu chéo để tham chiếu trong tài liệu.

1. GÁN NHÃN YÊU CẦU (LABELING REQUIREMENTS)

Để đáp ứng các tiêu chuẩn chất lượng của SRS về khả năng lằn vết và khả năng chỉnh sửa, mỗi yêu cầu phần mềm phải được định danh duy nhất. Sự định danh này cho phép bạn tham chiếu ngay tới mỗi yêu cầu khi cần thay đổi, cần tìm hiểu quá trình thay đổi của mỗi yêu cầu, cần tham chiếu chéo, hoặc cần lập ma trận lằn vết yêu cầu (requirement traceability matrix). Do các danh sách được lập theo kiểu tuyến tính không đáp ứng nhu cầu này nên tôi mô tả một số phương pháp gán nhãn yêu cầu cùng với hạn chế và thuận lợi của mỗi phương pháp. Bạn có thể lựa chọn một phương pháp phù hợp nhất cho mình.

Đánh số tuần tự (Sequence Number)

Cách đơn giản nhất là gán cho mỗi yêu cầu một số tuần tự, ví dụ UR-2, SRS13. Các công cụ quản lý yêu cầu thương mại hoá dùng cách này khi một yêu cầu mới được thêm vào CSDL (phần lớn các công cụ như vậy cũng hỗ trợ cách đánh số phân cấp). Tiềm tố xác định loại yêu cầu, ví dụ UR là yêu cầu người dùng (user requirements). Các số không được dùng lại, khi có một yêu cầu nào bị loại bỏ thì đơn giản là đánh dấu vào số đó, yêu cầu mới sẽ nhận số tiếp theo. Cách đánh số đơn giản này không tạo ra một nhóm các yêu cầu có liên quan với nhau về mặt logic hoặc nhóm yêu cầu phân cấp, các nhãn không cho bạn chỉ dẫn nào về mỗi yêu cầu nói cái gì.

Đánh số phân cấp (Hierarchical Numbering)

Có lẽ đó là cách đánh số thông dụng nhất. Nếu yêu cầu chức năng (functional requirements) xuất hiện trong mục 3.2 của SRS thì một trong số yêu cầu chức năng sẽ có các nhãn như 3.2.4.3. Các số thêm vào chỉ các yêu cầu ở mức chi tiết hơn, ưu tiên thấp hơn. Phương pháp này đơn giản và thuận tiện, có thể đánh số tự động được. Tuy nhiên, chúng không nói cho bạn biết về mục đích của mỗi yêu cầu. Nếu bạn thêm vào một yêu cầu mới thì các số của tất cả các yêu cầu tiếp theo trong mục này (section) sẽ tăng lên. Xoá hoặc loại bỏ một yêu cầu thì các số tiếp sau sẽ giảm

đi. Các thay đổi này có thể làm hỏng các tham chiếu lẫn nhau giữa các yêu cầu trên toàn hệ thống.

Một cải tiến đối với cách đánh số này là đánh số các mục chính (major sections) của yêu cầu theo cách phân cấp, sau đó định danh mỗi yêu cầu chức năng riêng trong mỗi mục bằng một đoạn mã ngắn (short text code) ngay phía sau số tuần tự. Ví dụ, SRS có thể chứa “Section 3.2.5 – Editor Functions”, và các yêu cầu trong mục này có thể đánh mã là ED-1, ED-2... Cách tiếp cận này khiến nhãn của yêu cầu ngắn, mã có ý nghĩa nhất định và định vị yêu cầu độc lập. Thêm một yêu cầu mới ED-9 vào giữa ED -1 và ED -2 không khiến bạn phải đánh số lại phần còn lại của mục (section) đó.

Nhãn yêu cầu bằng chữ có phân cấp (Hierarchical textual tags)

Nhà tư vấn Tom Gilb đề xuất một sơ đồ gán nhãn phân cấp bằng chữ để gán nhãn các yêu cầu riêng rẽ (Gilb 1988). Bạn hãy cân nhắc yêu cầu này: “Hệ thống sẽ hỏi người dùng để xác nhận bất cứ điều hỏi nào in trên 10 bản”. Yêu cầu này có thể gán nhãn PRINT.COPIES.CONFIRM, chúng chỉ ra đó là một phần của chức năng in ấn và liên quan đến vấn đề thiết lập số lượng các bản sao in. Cách gán nhãn này được cấu trúc, nhiều ngữ nghĩa, không ảnh hưởng gì khi thêm, xoá, loại bỏ mỗi yêu cầu. Nhược điểm của nó là “cồng kềnh” so với nhãn bằng số phân cấp.

2. XỬ LÝ SỰ KHÔNG ĐẦY ĐỦ (DEALING WITH INCOMPLETENESS)

Đôi khi bạn biết mình thiếu một số thông tin về một yêu cầu cụ thể nào đó. Bạn có thể cần đến sự tư vấn của khách hàng, cần kiểm tra một interface description của một hệ thống khác, hoặc cần định nghĩa một yêu cầu khác trước khi giải quyết yêu cầu chưa đầy đủ trên. Hãy sử dụng ký pháp TBD (cần xác định, to be determine) như một chỉ dẫn tiêu chuẩn (standard indicator) để làm rõ những chỗ nào chưa hiểu rõ trong SRS. Theo cách này, bạn có thể tìm các TBDs trong SRS để xác định các “vết tối” nơi mà bạn cần phải làm sáng tỏ vấn đề. Hãy ghi rõ vào tài liệu ai sẽ phân giải các vấn đề đó, phân giải thế nào và phân giải khi nào. Hãy đánh số mỗi TBD và tạo lập một danh sách TBDs để bạn có thể giám sát việc làm sáng tỏ chúng.

Phân giải tất cả các TBDs trước khi bạn bắt đầu bước tiếp theo của quy trình phát triển hệ thống, mỗi sự không chắc chắn nào đều làm tăng nguy cơ cho dự án sau này. Khi nhà phát triển phải đối mặt với một TBD hoặc một điểm nhập nhằng nào đó, anh ta không thể quay trở lại nơi phát sinh yêu cầu để làm sáng tỏ hoặc phân giải nó. Thay vì thế, anh ta sẽ phỏng đoán cách thức làm sáng tỏ chỗ chưa rõ đó và trong đa số các trường hợp điều đó không phải bao giờ cũng đúng. Nếu bạn vẫn phải tiếp tục quy trình phát triển mà vẫn còn các TBDs thì hãy trì hoãn thực hiện các yêu cầu liên quan nếu bạn có thể, hoặc hãy thiết kế hệ thống sao cho có thể dễ dàng chỉnh sửa thiết kế khi các TBDs được làm sáng tỏ sau.

3. GIAO DIỆN NGƯỜI DÙNG VÀ SRS

Đưa các thiết kế giao diện người dùng vào SRS vừa mang lại thuận lợi lại vừa có những nhược điểm. Dưới khía cạnh nhược điểm thì các hình ảnh màn hình và kiến trúc giao diện người dùng là các mô tả về giải pháp (tức là các thiết kế), chứ đó không phải là yêu cầu. Nếu bạn không thể vạch ranh giới SRS cho đến lúc thiết kế giao diện người dùng được hoàn thành thì quy trình phát triển yêu cầu sẽ kéo dài hơn là thời gian dự định dành cho nó. Việc này sẽ thách thức các nhà quản lý, khách hàng, hoặc các nhà phát triển - những người đã tiêu tốn thời gian để phát triển yêu cầu. Các mô tả giao diện người dùng bằng hình ảnh không phải là phương án thay thế cho định nghĩa rõ ràng về các yêu cầu chức năng (functional requirements). Đừng kỳ vọng nhà phát triển suy luận được các chức năng và quan hệ dữ liệu từ các phác thảo màn hình. Đưa các thiết kế giao diện người dùng vào SRS cũng ngũ ý nhà phát triển phải tuân theo quy trình thay đổi yêu cầu mỗi khi họ muốn lựa chọn một cách thể hiện giao diện người dùng.

Trong một khía cạnh khác, thể hiện các giao diện người dùng có thể giúp bạn làm minh các yêu cầu và làm rõ hơn các tương tác người dùng – hệ thống cho cả người dùng và nhà phát triển. Các thể hiện giao diện người dùng cũng có thể khiến cho việc lập kế hoạch dự án và ước lượng chi phí dễ dàng hơn. Bạn có thể đếm các phần tử GUI hoặc tính toán số lượng các điểm chức năng liên quan tới mỗi màn hình và ước lượng nguồn lực cần thiết để mã hóa các màn hình đó, việc này cũng dựa trên kinh nghiệm mà bạn đã có từ các dự án trước.

Một sự lựa chọn hợp lý là chỉ đưa vào SRS các hình ảnh dưới dạng ý tưởng (conceptual images) – các phác thảo - của các yếu tố giao diện người dùng chọn lọc mà không cần thiết phải làm giao diện người dùng đúng như mô hình đó. Cách làm này sẽ nâng cao khả năng truyền thông bằng cách biểu diễn các yêu cầu dưới một hình thức khác nhưng không ràng buộc các nhà phát triển và không bỏ qua quy trình quản lý thay đổi.

II. MẪU ĐẶC TẢ YÊU CẦU PHẦN MỀM (A SOFTWARE REQUIREMENTS SPECIFICATION TEMPLATE – SRS TEMPLATE)

Mỗi tổ chức phát triển phần mềm cần phải thống nhất một SRS template tiêu chuẩn cho tất cả các dự án. Một số SRS template được khuyến nghị trong (Davis 1993; Robertson and Robertson 1999). Dortman và Thayer (1990) đã tập hợp được 20 tiêu chuẩn yêu cầu (requirements standards) và một số ví dụ từ National Bureau of Standards, Bộ Quốc phòng Mỹ, NASA và một số nguồn khác từ Canada và Anh quốc. Nhiều người sử dụng các template của chuẩn IEEE Standard 830-1998, “IEEE Recommended Practice for Software Requirements Specifications” (IEEE 1998). Đây là một template được cấu trúc tốt, uyển chuyển và phù hợp với nhiều loại dự án phần mềm.

1. GIỚI THIỆU (INTRODUCTION)

- 1.1.Mục đích (Purpose)
- 1.2.Các quy ước của tài liệu (Document Conventions)
- 1.3.Hướng dẫn đọc tài liệu (Intended Audience and Reading Suggestions)
- 1.4.Phạm vi của sản phẩm phần mềm (Product Scope)
- 1.5.Tham chiếu (Reference)

2. MÔ TẢ TỔNG QUÁT (OVERALL DESCRIPTION)

- 2.1. Bối cảnh của sản phẩm phần mềm (Product Perspective)
- 2.2. Các chức năng của sản phẩm phần mềm (Product Functions)
- 2.3. Các lớp người dùng và đặc tính của mỗi lớp người dùng (User Classes and Characteristics)
- 2.4. Môi trường vận hành (Operating Environment)

2.5. Các ràng buộc thiết kế và thi công (Design and Implementation Constraints)

2.6. Các giả định và phụ thuộc (Assumptions and Dependencies)

3. CÁC YÊU CẦU GIAO DIỆN NGOÀI (EXTERNAL INTERFACE REQUIREMENTS)

3.1.Giao diện người dùng (User Interfaces)

3.2.Giao diện phần cứng (Hardware Interfaces)

3.3.Giao diện phần mềm (Software Interfaces)

3.4.Giao diện truyền thông (Communications Interfaces)

4. CÁC TÍNH NĂNG CỦA SẢN PHẨM PHẦN MỀM (SYSTEM FEATURES)

4.x. Tính năng X

4.x.1. Mô tả và mức ưu tiên (Description and Priority)

4.x.2. Chuỗi Kích thích/Đáp ứng (Stimulus/Response Sequences)

4.x.3. Các yêu cầu chức năng (Functional Requirements)

5. CÁC YÊU CẦU PHI CHỨC NĂNG KHÁC (OTHER NONFUNCTIONAL REQUIREMENTS)

5.1. Yêu cầu hiệu năng (Performance Requirements)

5.2. Yêu cầu an toàn (Safety Requirements)

5.3. Yêu cầu an ninh (Security Requirements)

5.4. Các thuộc tính chất lượng phần mềm (Software Quality Attributes)

5.5. Các quy tắc nghiệp vụ (Business Rules)

5.6. Tài liệu người dùng (User Documentation)

6. CÁC YÊU CẦU KHÁC (OTHER REQUIREMENTS)

PHỤ LỤC A: BẢNG THUẬT NGỮ (APPENDIX A: GLOSSARY)

PHỤ LỤC B: MÔ HÌNH PHÂN TÍCH (APPENDIX B: ANALYSIS MODELS)

PHỤ LỤC C: DANH SÁCH TBD (APPENDIX C: TBD LIST)

HÌNH 9-1. SRS template

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Hình 9-1 minh họa một SRS template đã được thông qua và được mở rộng theo chuẩn IEEE 830. Hãy chỉnh sửa template này để thích ứng với nhu cầu và bản chất dự án của bạn. Nếu một mục nào đó của template này không ứng dụng được cho dự án của bạn thì hãy chỉ để đâu đê thôi nhưng ghi chú là không ứng dụng. Bạn làm như thế để tránh đọc giả đọc tài liệu và băn khoăn tự hỏi liệu có cái gì đó quan trọng bị quên hay không. Hãy sử dụng template này để định hướng tư duy của bạn, hãy thêm bất cứ mục nào mà bạn thấy cần thiết đối với dự án của bạn. Như mọi tài liệu phần mềm, hãy thêm một bảng mục lục và một lịch sử soát xét (revision history) để biết các thay đổi đã thực hiện với SRS, gồm ngày thay đổi, ai thực hiện thay đổi, lý do thay đổi.

Phần còn lại của mục II Chương này sẽ diễn giải mỗi đề mục trong template trên để bạn rõ. Bạn có thể nối kết với các tài liệu khác bằng các tham chiếu (như tài liệu tầm nhìn và phạm vi, thiết kế đặc tả giao diện).

1. GIỚI THIỆU (INTRODUCTION)

Phần này trình bày một cái nhìn khái quát về SRS để giúp đọc giả hiểu tài liệu được tổ chức như thế nào, đọc và diễn giải tài liệu như thế nào cho đúng.

1.1. Mục đích (Purpose)

Chỉ ra yêu cầu phần mềm được trình bày ở đây thuộc sản phẩm nào, xác định cả số soát xét (revision number) hoặc số phiên bản (release number). Nếu SRS này chỉ mô tả yêu cầu của một phần hệ thống thì hãy chỉ rõ hệ thống con đó.

1.2. Các quy ước của tài liệu (Document Conventions)

Mô tả bất cứ chuẩn nào hoặc quy ước nào mà bạn tuân theo khi viết tài liệu SRS này, bao gồm kiểu chữ (text styles), nhấn mạnh (highlighting) và các ký pháp khác. Ví dụ, liệu mức ưu tiên được gán cho một yêu cầu mức cao có được thừa kế bởi tất cả các yêu cầu chi tiết dẫn từ yêu cầu mức cao đó, hoặc liệu mỗi mô tả yêu cầu đều có một mức ưu tiên.

1.3. Hướng dẫn đọc tài liệu (Intended Audience and Reading Suggestions)

Liệt kê các độc giả trực tiếp của SRS như người phát triển, các nhà quản trị dự án, người làm marketing, người dùng, người kiểm thử, người viết tài liệu. Mô tả phần Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

còn lại của SRS được tổ chức như thế nào. Hướng dẫn cách đọc tài liệu thích hợp nhất cho mỗi loại độc giả.

1.4. Phạm vi của sản phẩm phần mềm (Product Scope)

Viết một mô tả ngắn về phần mềm đang được đặc tả và mục đích của nó, lợi ích mà nó mang lại cho các bên liên quan. Nên viết mô tả này trong định hướng của chiến lược kinh doanh của doanh nghiệp. Nếu bạn đã có sẵn một tài liệu tầm nhìn và phạm vi (vision and scope) thì hãy tham chiếu đến tài liệu đó từ đây.

1.5. Tham chiếu (Reference)

Liệt kê mọi tài liệu và các nguồn khác mà SRS tham chiếu, đó có thể là tài liệu hướng dẫn thiết kế giao diện người dùng, các hợp đồng, các tiêu chuẩn, các đặc tả yêu cầu hệ thống, tài liệu use-case, hoặc SRS của một sản phẩm liên quan. Cung cấp thông tin đủ để người đọc có thể truy nhập đến mỗi tham chiếu, bao gồm tên tài liệu, tác giả, số phiên bản, ngày tháng, nguồn hoặc vị trí của tài liệu.

2. MÔ TẢ TỔNG QUÁT (OVERALL DESCRIPTION)

Mục này giới thiệu một cái nhìn khái quát về sản phẩm đang được đặc tả, về môi trường tại đó sản phẩm được sử dụng, về những người dùng tương lai của sản phẩm, về các ràng buộc đã biết, các giả định và các phụ thuộc khác.

2.1. Bối cảnh của sản phẩm phần mềm (Product Perspective)

Mô tả bối cảnh và sự phát sinh sản phẩm đang được đặc tả trong SRS này. Hãy xác định sản phẩm này có phải là thành viên tiếp theo của một dòng sản phẩm hay không, hay là phiên bản tiếp theo của một sản phẩm khác, là sự thay thế của một sản phẩm đang chạy, là một sản phẩm mới. Nếu SRS này định nghĩa một thành phần (component) của một hệ thống lớn hơn thì hãy xác định phần mềm đó liên quan như thế nào đến sản phẩm lớn hơn đó, hãy chỉ ra giao diện giữa chúng.

2.2. Các chức năng của sản phẩm phần mềm (Product Functions)

Tóm tắt các chức năng chính mà sản phẩm phải thực hiện. Các chi tiết sẽ được cung cấp trong Mục 4, vì vậy bạn chỉ cần một tóm tắt tổng quát ở đây thôi. Hãy tổ chức các chức năng sao cho chúng dễ nắm bắt với bất cứ độc giả nào. Một bức

tranh nhóm lại các yêu cầu và sự liên quan giữa chúng, như DFD khái quát hoặc sơ đồ lớp chặng hạn, có thể giúp đạt mục đích này.

2.3. Các lớp người dùng và đặc tính của mỗi lớp người dùng (User Classes and Characteristics)

Xác định các lớp người dùng khác nhau mà bạn đoán sẽ sử dụng sản phẩm này và mô tả các đặc tính tương ứng của mỗi lớp người dùng (Xem Chương 7). Một số yêu cầu có thể gắn liền với duy nhất một lớp người dùng nào đó. Phân biệt giữa lớp người dùng quan trọng nhất và các lớp người dùng khác ít quan trọng hơn.

2.4. Môi trường vận hành (Operating Environment)

Mô tả môi trường trong đó phần mềm vận hành, gồm nền tảng phần cứng, hệ điều hành và số hiệu phiên bản, các components và các ứng dụng cần thiết khác.

2.5. Các ràng buộc thiết kế và thi công (Design and Implementation Constraints)

Xác định bất cứ vấn đề nào làm giới hạn khả năng lựa chọn của nhà phát triển và mô tả tại sao lại có các giới hạn đó. Các giới hạn đó có thể là:

- Các công nghệ cụ thể, công cụ, ngôn ngữ lập trình, CSDL nên sử dụng hoặc cần tránh.
- Các quy ước và tiêu chuẩn cần thiết (ví dụ, nếu tổ chức của khách hàng sẽ bảo trì phần mềm thì cần mô tả các ký pháp được dùng để thiết kế, các chuẩn mã hóa và trao đổi cho khách hàng).
- Các chính sách, quy định của Chính phủ, các chuẩn công nghiệp.
- Các giới hạn của phần cứng như tốc độ phản ứng, giới hạn bộ nhớ.
- Các khuôn dạng trao đổi dữ liệu chuẩn.

2.6. Các giả định và phụ thuộc (Assumptions and Dependencies)

Liệt kê mọi yếu tố giả định (đối lập lại với các yếu tố đã được biết rõ) có thể ảnh hưởng tới các yêu cầu đã được xác định trong SRS. Các yếu tố giả định có thể gồm các components có sẵn mà bạn lập kế hoạch sử dụng, các vấn đề xung quanh việc phát triển hoặc môi trường điều hành. Bạn có thể giả định rằng sản phẩm sẽ tuân thủ một quy ước thiết kế giao diện người dùng riêng nào đó, một người khác có thể

giả định khác. Dự án có thể bị ảnh hưởng nếu các giả định đó không đúng đắn, không được chia sẻ, hoặc thay đổi.

Cũng vậy, hãy chỉ ra bất cứ sự phụ thuộc nào của dự án vào các yếu tố bên ngoài. Ví dụ, nếu bạn mong muốn tích hợp vào hệ thống một số components đang được phát triển bởi một dự án khác thì có nghĩa bạn đã bị phụ thuộc vào lịch biểu của dự án kia. Nếu các phụ thuộc đó đã được tài liệu hóa ở đâu đó, ví dụ kế hoạch dự án, thì hãy tham chiếu tới tài liệu đó.

3. CÁC YÊU CẦU GIAO DIỆN NGOÀI (EXTERNAL INTERFACE REQUIREMENTS)

Sử dụng mục này để đặc tả các yêu cầu mô tả việc sản phẩm mới sẽ kết nối hợp lệ với các components ngoài (external components). Sơ đồ bối cảnh (context diagram) thể hiện các giao diện ngoài ở một mức trừu tượng cao. Hãy mô tả chi tiết components dữ liệu và components kiểm soát (data and control components) của giao diện trong từ điển dữ liệu.

3.1. Giao diện người dùng (User Interfaces)

Hãy mô tả các tiêu chuẩn cần thiết của giao diện người dùng như:

- Các tiêu chuẩn GUI hoặc các hướng dẫn về phong cách của dòng sản phẩm (product family style) cần phải được tuân thủ.
- Sự sắp xếp màn hình hoặc các ràng buộc về độ phân giải.
- Các buttons chuẩn, chức năng chuẩn, các phím tắt.
- Các chuẩn hiển thị error message.

Tài liệu hóa chi tiết thiết kế giao diện người dùng, như sự sắp xếp các dialog boxes cụ thể, được mô tả trong tài liệu đặc tả giao diện người dùng chứ không phải trong SRS.

3.2. Giao diện phần cứng (Hardware Interfaces)

Hãy mô tả đặc tính của mỗi giao diện giữa các hardware components và các software components của hệ thống. Mô tả này có thể bao gồm các loại thiết bị trợ giúp, bản chất của các tương tác dữ liệu và tương tác kiểm soát (control

interaction) giữa phần mềm và phần cứng, các giao thức thông tin liên lạc được sử dụng.

3.3. Giao diện phần mềm (Software Interfaces)

Mô tả sự kết nối giữa sản phẩm và các software components ngoài khác (xác định tên và số phiên bản) như các CSDL, HĐH, tools, thư viện, các components thương mại tích hợp khác. Xác định và mô tả mục đích của các data items hoặc các messages trao đổi giữa sản phẩm và các software components đó. Hãy mô tả các dịch vụ cần thiết và bản chất của sự liên lạc giữa các components. Xác định dữ liệu được dùng chung giữa các software components. Nếu cơ chế chia sẻ dữ liệu cần phải được thực hiện theo một cách riêng biệt, một vùng dữ liệu lớn trong một HĐH đa nhiệm chẳng hạn, thì hãy mô tả điều đó như một ràng buộc thực thi.

3.4. Giao diện truyền thông (Communications Interfaces)

Mô tả các yêu cầu liên quan đến các chức năng truyền thông mà sản phẩm sẽ phải sử dụng, gồm email, Web browser, chuẩn giao thức truyền thông của mạng,... Đặc tả yêu cầu an ninh truyền thông, tốc độ truyền dữ liệu, cơ chế đồng bộ hóa.

4. CÁC TÍNH NĂNG CỦA SẢN PHẨM PHẦN MỀM (SYSTEM FEATURES)

Template trong Hình 9-1 thể hiện các yêu cầu chức năng (functional requirements) được thể hiện thành các tính năng hệ thống (system features) – các dịch vụ chính được hệ thống cung cấp. Bạn có thể thích thể hiện các nội dung này dưới dạng các use cases, lớp người dùng, lớp đối tượng hoặc phân cấp chức năng hơn (IEEE 1998). Bạn cũng có thể sử dụng kết hợp những cách biểu diễn này, hãy chọn cách biểu diễn nào mà bạn và người dùng thấy thuận tiện nhất.

4.x. Tính năng X

Đặt một tên ngắn gọn cho tính năng, ví dụ “4.1. Spell Check and Spelling Dictionary Management”. Bạn sẽ lặp lại những gì tương tự cho tính năng 4.x.2, 4.x.3...

4.x.1. Mô tả và mức ưu tiên (Description and Priority)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Mô tả ngắn gọn về tính năng và gán cho nó một mức ưu tiên cao, trung bình hay thấp. Xem chi tiết cách xếp mức ưu tiên trong Chương 13.

4.x.2. Chuỗi Kích thích/Đáp ứng (Stimulus/Response Sequences)

Liệt kê danh sách các kích thích (hành động của người dùng, tín hiệu từ các thiết bị ngoài, hoặc các triggers khác), các đáp ứng của hệ thống định nghĩa hành vi của tính năng này. Các chuỗi kích thích/đáp ứng đó phải phù hợp với các thảo luận về đối thoại trong use cases như đã mô tả ở Chương 8.

4.x.3. Các yêu cầu chức năng (Functional Requirements)

Ghi lại từng yêu cầu chức năng chi tiết có liên quan đến tính năng này. Đó chính là các công năng phần mềm (software capabilities) cần phải có để người dùng thực hiện công việc của họ - các tác vụ (tasks) đã được mô tả trong một use case. Hãy mô tả sản phẩm đáp ứng như thế nào các điều kiện gây ra lỗi đã được dự đoán hoặc các đầu vào không đúng khác. Hãy xác định mỗi yêu cầu một cách duy nhất như đã mô tả từ đầu trong chương này.

5. CÁC YÊU CẦU PHI CHỨC NĂNG KHÁC (OTHER NONFUNCTIONAL REQUIREMENTS)

Trong mục này, bạn liệt kê bất cứ yêu cầu phi chức năng nào ngoài các yêu cầu về giao diện ngoài và các ràng buộc.

5.1. Yêu cầu hiệu năng (Performance Requirements)

Xác định bất cứ yêu cầu hiệu năng nào của sản phẩm cho các kịch bản sử dụng khác nhau (various usage scenarios), diễn giải về mục đích của các yêu cầu hiệu năng này để giúp các nhà phát triển đưa ra các lựa chọn thiết kế phù hợp. Chỉ ra (specify) số lượng những người dùng hệ thống đồng thời (concurrent users) hoặc các phép xử lý (operations) được thực hiện, thời gian đáp ứng, và các quan hệ thời gian đối với hệ thống thời gian thực (realtime system). Bạn cũng có thể chỉ ra các yêu cầu dung lượng (capacity requirements) ở đây, như yêu cầu dung lượng bộ nhớ và không gian đĩa, hoặc số lượng tối đa các bản ghi được lưu trữ trong các bảng của cơ sở dữ liệu. Nếu có thể định lượng được các yêu cầu hiệu năng thì hãy cố gắng định lượng. Bạn có thể phải xác định các yêu cầu hiệu năng cho từng yêu cầu chức năng hoặc tính năng một, hơn là tập hợp tất chúng vào trong một nhóm. Ví

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

dụ, “95% các truy vấn catalog database cần phải được hoàn thành trong 2s trên một PC sử dụng con chip Pentium IV tốc độ 1200 MHz, cài đặt HĐH Windows 2000, với ít nhất 50% tài nguyên hệ thống là còn trống.”

5.2. Yêu cầu an toàn (Safety Requirements)

Xác định các yêu cầu liên quan đến các mất mát, hư hại xảy ra do sử dụng sản phẩm. Chỉ ra và thực hiện bất cứ cách thức hoặc hành động nào khiến cho việc sử dụng sản phẩm an toàn, cũng như các hành động nguy hiểm tiềm tàng cần phải tránh. Chỉ ra bất cứ chứng chỉ, chính sách và quy định an toàn nào mà sản phẩm cần tuân thủ. Một ví dụ về yêu cầu an toàn là: “Một hoạt động vận hành cần phải chấm dứt ngay trong vòng 1s nếu áp lực đo được vượt quá 95% áp lực tối đa được phép.”

5.3. Yêu cầu an ninh (Security Requirements)

Xác định bất cứ yêu cầu nào liên quan đến vấn đề an ninh, đảm bảo tính toàn vẹn của dữ liệu, hoặc các vấn đề liên quan đến tính riêng tư ảnh hưởng đến việc sử dụng sản phẩm, dữ liệu được sử dụng hoặc được tạo ra bởi sản phẩm. Định nghĩa bất cứ yêu cầu xác thực (authentication) hoặc yêu cầu cấp quyền (authorization) nào cần thiết. Xác định bất cứ chính sách an ninh hoặc đảm bảo tính riêng tư hoặc chứng chỉ mà sản phẩm cần thỏa mãn. Bạn cần phải ưu tiên xác định các yêu cầu thông qua thuộc tính chất lượng được gọi là đảm bảo tính toàn vẹn (integrity), thuộc tính này sẽ được mô tả trong Chương 11. Một ví dụ về yêu cầu an ninh là: “Mỗi người dùng cần phải thay đổi ngay lập tức mật khẩu được gán cho họ trong lần truy nhập đầu tiên vào hệ thống. Mật khẩu đầu tiên này không thể được sử dụng lại.”

5.4. Các thuộc tính chất lượng phần mềm (Software Quality Attributes)

Xác định bất cứ đặc tính chất lượng nào khác là quan trọng đối với khách hàng hoặc nhà phát triển sản phẩm. (Xem Chương 11). Các đặc tính đó cần phải cụ thể, có thể định lượng, và có thể kiểm tra nếu cần thiết. Ít nhất cũng phải xác định một số đặc tính như sản phẩm phải dễ học, dễ sử dụng hoặc linh hoạt trên nhiều môi trường cài đặt khác nhau.

5.5. Các quy tắc nghiệp vụ (Business Rules)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Liệt kê bất cứ nguyên tắc vận hành nào (operating principles) đối với sản phẩm, ví dụ các cá nhân hoặc vai trò nào chỉ có thể thực hiện các chức năng nào đó trong một hoàn cảnh cụ thể. Đó không thật sự là yêu cầu chức năng nhưng chúng ngũ ý một số yêu cầu chức năng nhằm tăng cường cho các quy tắc nghiệp vụ. Ví dụ về một quy tắc nghiệp vụ là: “Chỉ những người dùng có mã truy nhập điều hành (supervisor) mới có thể ký phiếu xuất tiền mặt 100.000 USD hoặc hơn.”

5.6. Tài liệu người dùng (User Documentation)

Liệt kê các thành phần của tài liệu người dùng (user documentation components) sẽ được chuyển giao cùng với phần mềm, ví dụ như sổ tay người dùng (user manuals), tài liệu hỗ trợ trực tuyến, hoặc hướng dẫn tức thì (tutorials). Định ra các khuôn dạng và tiêu chuẩn của tài liệu chuyển giao.

6. CÁC YÊU CẦU KHÁC (OTHER REQUIREMENTS)

Định nghĩa bất cứ yêu cầu nào không được mô tả trong SRS, ví dụ như yêu cầu quốc tế hóa hoặc địa phương hóa, hoặc yêu cầu pháp luật (legal requirement). Bạn cũng có thể thêm vào các mục về vận hành, quản trị và bảo trì để kiểm soát việc cài đặt sản phẩm, cấu hình sản phẩm, khởi động và kết thúc phiên làm việc, khôi phục và chịu lỗi, ghi nhật ký vận hành (log) và giám sát vận hành. Thêm bất cứ mục nào vào template cho phù hợp với dự án của bạn. Nếu bạn không phải thêm vào yêu cầu nào khác, hãy bỏ qua mục này.

PHỤ LỤC A: BẢNG THUẬT NGỮ (APPENDIX A: GLOSSARY)

Định nghĩa tất cả các khái niệm cần thiết để đọc giả hiểu một cách đúng đắn SRS, các khái niệm như vậy gồm các từ cấu tạo từ các chữ cái đầu của chuỗi từ (như NASA), các từ viết tắt.

PHỤ LỤC B: MÔ HÌNH PHÂN TÍCH (APPENDIX B: ANALYSIS MODELS)

Mục tùy chọn này bao hàm, hoặc tham chiếu đến những vị trí đã có, các mô hình phân tích phù hợp như DFDs, class diagrams, state-transition diagrams, ERD. (Xem Chương 11).

PHỤ LỤC C: DANH SÁCH TBD (APPENDIX C: TBD LIST)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Biên soạn một danh sách được đánh số thứ tự các tham chiếu TBD được duy trì trong SRS sao cho chúng có thể được giám sát để hoàn thành.

III. HƯỚNG DẪN VIẾT YÊU CẦU (GUIDELINES FOR WRITING REQUIREMENTS)

Không có cách nào đã được lập thành công thức để viết các yêu cầu tuyệt hảo, người thầy tốt nhất chính là kinh nghiệm của bạn. Học từ các bài toán mà bạn đã đối mặt trong quá khứ sẽ dạy bạn nhiều hơn. Nhiều tài liệu yêu cầu có thể được cải tiến chất lượng thông qua các hướng dẫn kỹ thuật viết hiệu quả và sử dụng các thuật ngữ của người dùng hơn là biệt ngữ máy tính (Kovitz 1999). Ghi nhớ các khuyến nghị sau khi bạn tài liệu hóa các yêu cầu phần mềm:

- Sử dụng các câu và các đoạn ngắn.
- Sử dụng cách nói tích cực (active voice).
- Viết các câu đầy đủ đúng ngữ pháp, chính tả và chấm câu đầy đủ.
- Dùng thuật ngữ nhất quán như đã được định nghĩa trong bảng thuật ngữ.
- Định vị các yêu cầu trong một cách thức nhất quán như “Hệ thống phải” hoặc “Người dùng phải”, theo sau là một động từ, sau đó là một kết quả quan sát được. Ví dụ: “Hệ thống con quản lý kho phải hiển thị một danh sách tất cả các công-ten-nơ chứa hóa chất được yêu cầu mà hóa chất này hiện có trong kho.”
- Giảm bớt sự *nhập nhằng*, tránh sự *mơ hồ*, các khái niệm chủ quan như *thân thiện người dùng, dễ, đơn giản, nhanh, hiệu quả, hỗ trợ, một số, thuần thục, mạnh hơn, có thể chấp nhận, mạnh* (*user-friendly, easy, simple, rapid, efficient, support, several, state-of-the-art, superior, acceptable, robust*). Hãy tìm kiếm những cách gì đó để diễn đạt những khái niệm này sao cho khách hàng có thể đo lường được.
- Tránh các từ so sánh như *cải tiến, tối đa hóa, tối thiểu hóa, tối ưu hóa* (*improve, maximize, minimize, optimize*). Định lượng hóa các mức độ cải tiến là cần thiết, hoặc xác định các giá trị có thể chấp nhận của sự tối đa hóa, tối thiểu hóa thông qua một số tham số. Hãy chắc chắn rằng bạn biết cái mà khách hàng ngũ ý khi họ nói hệ thống mới phải “xử lý”, “hỗ trợ”, hoặc “quản lý” một cái gì đó. Ngôn ngữ nhập nhằng sẽ dẫn tới các yêu cầu không thể kiểm tra.

Do các yêu cầu được viết dưới dạng phân cấp (hierarchy), hãy phân rã một yêu cầu mức cao nhập nhằng thành các yêu cầu ở mức độ thấp hơn nhằm làm sáng tỏ yêu cầu mức cao và xoá bỏ sự nhập nhằng. Viết các yêu cầu đủ chi tiết sao cho nếu yêu cầu được hoàn thành thì nhu cầu của khách hàng sẽ được đáp ứng, nhưng không quá chi tiết để có thể xuất hiện những ràng buộc không cần thiết đối với thiết kế. Nếu bạn có thể hoàn thành một yêu cầu theo một số cách và tất cả đều có thể được chấp nhận thì mức độ chi tiết thế là đủ. Tuy nhiên, nếu một người thiết kế xem xét SRS mà không rõ về ngữ ý của khách hàng thì bạn cần phải đưa thêm vào các chi tiết để giảm bớt khả năng làm lại sản phẩm nếu một sự không hiểu ý lại xuất hiện.

Các tác giả của yêu cầu thường phải vật lộn (struggle) để tìm ra mức phân rã yêu cầu hợp lý. Một hướng dẫn hữu hiệu là viết các yêu cầu sao cho có thể kiểm thử từng yêu cầu một. Nếu bạn có thể nghĩ được một số ít các test cases có liên quan để xác nhận rằng một yêu cầu đã được thực thi đúng thì mức phân rã chi tiết như vậy là hợp lý. Nếu các phép kiểm thử mà bạn mường tượng là nhiều và đa dạng thì có lẽ một số yêu cầu đã được gom lại cùng nhau và cần phải tách chúng riêng rẽ ra. Các yêu cầu có thể kiểm thử được gọi ý như là một số đo (metric) cho kích thước của sản phẩm phần mềm (Wilson 1995).

Viết các yêu cầu ở một mức chi tiết nhất quán. Tôi đã xem các lời thề hiện yêu cầu (requirement statements) trong cùng một SRS và thấy chúng biến đổi một cách rộng rãi về phạm vi. Ví dụ, “Sự kết hợp các phím Control – S được diễn giải như là File Save” và “Sự kết hợp các phím Control – P được diễn giải như là File Print” được phân tách thành các yêu cầu riêng rẽ. Tuy nhiên, “Sản phẩm phải đáp ứng được việc soạn thảo văn bản trực tiếp bằng lời nói” được mô tả như một hệ thống con toàn vẹn chứ không phải là một yêu cầu chức năng đơn nhất.

Tránh các đoạn tường thuật dài chứa nhiều yêu cầu. Các liên từ như “và”, “hoặc” trong một yêu cầu gợi ý đó là nhiều yêu cầu được kết hợp lại. Không bao giờ sử dụng “và”, “hoặc”, “vân, vân...” trong một câu thể hiện yêu cầu.

Tránh định vị (stating) các yêu cầu một cách dư thừa trong SRS. Mặc dù ghép cùng một yêu cầu ở nhiều vị trí khác nhau có thể khiến tài liệu dễ đọc hơn nhưng Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

cũng khó bảo trì hơn. Nhiều thể hiện (instances) của một yêu cầu cùng phải được cập nhật tại mỗi thời điểm, vì vậy sẽ có thể dẫn tới sự thiếu nhất quán. Các tham chiếu chéo liên quan giữa các yếu tố (items) của SRS giúp giữ được sự đồng bộ khi có các thay đổi. Lưu trữ chỉ một lần mỗi yêu cầu trong một công cụ quản lý yêu cầu hoặc một cơ sở dữ liệu sẽ tránh các vấn đề dư thừa.

Hãy suy nghĩ về cách thức hiệu quả nhất để biểu diễn mỗi yêu cầu. Cần nhắc một tập các yêu cầu được biểu diễn theo mẫu (pattern) sau: “Bộ Text Editor phải phân tích cú pháp của tài liệu theo *<format>* định nghĩa các quy tắc *<jurisdiction>*”. Có 3 giá trị có thể cho *format* và 4 giá trị có thể cho *jurisdiction*, như vậy có tất cả 12 yêu cầu tương tự nhau. Khi bạn xét duyệt (review) một danh sách các yêu cầu tương tự nhau, sẽ rất khó để nhận ra một yêu cầu bị khuyết, ví như “Bộ Text Editor phải phân tích cú pháp của tài liệu *untagged* định nghĩa các các quy tắc *international*”. Biểu diễn các yêu cầu theo mẫu như trong Bảng 9-1 thì bạn sẽ không bị mất yêu cầu nào. Yêu cầu mức cao có thể được định vị thế này “ED-13. Bộ Text Editor cần phải duyệt các tài liệu trong một số *formats* định nghĩa các quy tắc theo nhiều *jusridictions*, như thể hiện trong Table <XX>.”

BẢNG 9-1. KHUÔN DẠNG BẢNG MẪU ĐỂ LIỆT KÊ CÁC SỐ YÊU CẦU THEO MỘT PATTERN			
Jusridiction	Tagged Format	Untagged Format	ASCII Format
Federal	ED-13.1	ED-13.2	ED-13.3
State	ED-13.4	ED-13.5	ED-13.6
Teritorial	ED-13.7	ED-13.8	ED-13.9
International	ED-13.10	ED-13.11	ED-13.12

CÁC YÊU CẦU MẪU, TRƯỚC VÀ SAU

Chương 1 đã xác định một số đặc tính của lời thể hiện yêu cầu chất lượng cao là: *đầy đủ, đúng đắn, khả thi, cần thiết, được ưu tiên hóa, không nhập nhằng, và có thể kiểm tra*. Vì vậy, các yêu cầu mà không thể hiện các đặc tính đó thì sẽ gây sự rối loạn và nhiều công việc cứ phải làm đi, làm lại, bạn cần phải tìm và sửa chữa bất cứ vấn đề nào của các yêu cầu càng sớm càng tốt. Dưới đây là một số yêu cầu có vấn đề được trích từ các dự án thật. Hãy kiểm tra mỗi yêu cầu trong bối cảnh các đặc tính chất lượng trên để xem liệu bạn có thể phát hiện các vấn đề hay không. Tôi đã giới thiệu suy nghĩ của tôi về những gì có thể sai trong mỗi yêu cầu. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com.

và các đề nghị cải tiến. Tôi không nghi ngờ gì ai đó sẽ làm tốt hơn tôi, nhưng mục đích của bạn không phải là viết các yêu cầu hoàn hảo (perfect). Bạn chỉ cần viết các yêu cầu đủ tốt để nhóm của bạn thực hiện các thiết kế và thi công hệ thống dựa trên các yêu cầu đó với mức độ rủi ro có thể chấp nhận.

Ví dụ 1

“Sản phẩm cần phải cung cấp các messages trạng thái trong một khe thời gian theo quy định không nhỏ hơn mỗi 60 s.”

Yêu cầu này có vẻ như không đầy đủ: *Messages trạng thái* là gì và dưới điều kiện nào thì chúng được thể hiện ra cho người dùng thấy? Chúng sẽ duy trì trạng thái thể hiện trong bao lâu? Phân nào của “sản phẩm” mà chúng ta đang nói về nó? Khe thời gian cũng rất khó hiểu. Có phải khe thời gian thể hiện các messages trạng thái là ít nhất 60 s, vậy thể hiện một message mới trong mỗi năm cũng được sao? Nếu ngũ ý là không lớn hơn 60 s giữa các messages, vậy 1 ms có quá ngắn không? Sự nhất quán của khe thời gian khi thể hiện các messages là như thế nào? Từ “mỗi” chỉ gây thêm sự lộn xộn trong câu mà thôi. Do các vấn đề đó mà yêu cầu này là không thể kiểm tra.

Dưới đây là một cách mà chúng ta có thể viết lại yêu cầu để chỉ mặt đặt tên các hạn chế đó (thiết lập một số phỏng đoán về những gì được ngũ ý là cái mà chúng ta phải xác nhận lại với khách hàng):

1. Background Task Manager (BTM) phải thể hiện các messages trạng thái trong một vùng đã định của giao diện người dùng.
 - 1.1.Các messages phải được cập nhật mỗi 60 s (hơn hoặc kém 10) sau khi việc xử lý tác vụ nền bắt đầu và phải duy trì liên tục sự thể hiện.
 - 1.2.Nếu việc xử lý tác vụ nền đang được thực hiện bình thường, thì BTM phải thể hiện lượng phần trăm tác vụ nền đã được hoàn thành.
 - 1.3.BTM phải thể hiện một “Done” message khi tác vụ nền hoàn thành.
 - 1.4.BTM phải thể hiện một error message nếu tác vụ nền chết.

Tôi ché nhỏ thành nhiều yêu cầu vì mỗi cái trong số chúng đòi hỏi mỗi test case riêng và phải lần vết riêng. Nếu một số yêu cầu xoắn bện lại nhau trong một đoạn (paragraph) thì sẽ dễ bị bỏ sót khi thi công hoặc kiểm thử. Chú ý rằng yêu cầu đã xét duyệt (revised requirement) không đặc tả chính xác các messages trạng thái sẽ

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

được thể hiện như thế nào. Đó là công việc của thiết kế, và nếu chúng ta đặc tả chính xác ở đây thì nó sẽ trở thành một ràng buộc thiết kế đối với nhà phát triển, nếu làm như vậy thì có thể sẽ có các thiết kế không tối ưu.

Ví dụ 2

“Sản phẩm phải chuyển đổi ngay lập tức giữa việc thể hiện và che dấu các ký tự không được in.”

Máy tính chẳng thể làm cái gì ngay lập tức được, vậy yêu cầu này không khả thi. Nó cũng không đầy đủ do không định vị được nguyên nhân dẫn tới việc chuyển đổi. Có phải phần mềm tự tạo ra sự thay đổi dưới một số điều kiện nào đó hoặc người dùng thực hiện hành động khởi đầu cho sự thay đổi? Cũng vậy, phạm vi của việc thay đổi thể hiện trong tài liệu là gì: các ký tự đã được chọn, toàn bộ tài liệu? Vấn đề quá nhập nhằng. Các ký tự “không được in” là các đoạn văn bị che hay là các ký tự điều khiển? Vậy, yêu cầu này là không thể kiểm tra.

Đây có thể là cách tốt hơn để viết yêu cầu “Người dùng cần phải kiểm soát giữa việc thể hiện hay che dấu tất cả các HTML markup tags trong tài liệu đang được soạn thảo với sự kích hoạt của một cơ chế trigger cụ thể.” Như vậy các ký tự “không được in” là HTML markup tags. Yêu cầu đã được sửa chữa đã chỉ ra rằng người dùng kích hoạt sự thay đổi, nhưng không ràng buộc thiết kế do không định nghĩa chính xác cơ chế đó là gì. Khi người thiết kế chọn một cơ chế kích hoạt thích hợp (như một phím nóng, một menu command, một voice input), bạn có thể viết các tests cụ thể để kiểm tra xem liệu việc kích hoạt hoạt động tốt hay không.

Ví dụ 3

“Bộ phân tích cú pháp phải sinh ra một HTML markup error report cho phép phân giải nhanh các errors khi được sử dụng bởi những người mới học HTML.”

Từ “nhanh” là một sự nhập nhằng. Thiếu định nghĩa về những gì dẫn tới error report chỉ ra sự không đầy đủ. Tôi không chắc về việc bạn sẽ kiểm tra yêu cầu này như thế nào. Hãy tìm một ai đó là người mới học HTML và xem liệu có ta có thể phân giải các errors đủ nhanh bằng cách sử dụng report? Việc người mới học sử

dụng tham chiếu tới bộ phân tích cú pháp hay error report cũng không sáng sủa. Và khi nào thì report được sinh ra?

Có thể sửa chữa như sau:

1. Sau khi bộ phân tích cú pháp HTML đã phân tích xong 1 tệp, thì nó phải sinh ra một error report chứa số dòng và đoạn mô tả về bất cứ lỗi HTML nào trong tệp đã phân tích và một mô tả ngắn về mỗi error được tìm thấy.
2. Nếu không có error nào được tìm thấy thì error report sẽ không được sinh ra.

Bây giờ chúng ta đã biết khi nào thì error report được sinh ra và nội dung của nó là gì, nhưng chúng ta hãy để người thiết kế quyết định report trông cụ thể như thế nào. Chúng ta cũng đã đặc tả một điều kiện loại trừ (exception condition): nếu không có error nào thì không sinh ra report.

Ví dụ 4

“Các charge numbers phải (should) được xác nhận trực tuyến dựa trên master corporate charge number list, nếu có thể.”

Tôi thấy ngay: “Nếu có thể” là gì? Nó có khả thi về mặt kỹ thuật hay không? Nếu *master corporate charge number list* có thể được truy nhập trực tuyến? Nếu bạn không chắc chắn liệu một công năng (capability) được đề nghị có thể được chuyển giao, hãy sử dụng TBD để chỉ ra chi tiết này cần được phân giải. Yêu cầu là không đầy đủ do nó không xác định được điều gì xảy ra nếu sự xác nhận bị bỏ qua hoặc bị lỗi. Tránh các từ không chính xác như “phải”. Khách hàng hoặc cần chúc năng này hoặc không. Một số đặc tả yêu cầu sử dụng các phân biệt tinh tế trong các từ như *shall*, *should* hoặc *may* như là một cách để chỉ ra tầm quan trọng. Tôi thì thích sử dụng *shall* hoặc *will* như một lời thề hiện sang sửa về ngữ ý của yêu cầu và để đặc tả tầm ưu tiên một cách rõ ràng.

Dưới đây là một cách cải tiến lời thề hiện yêu cầu này:

“Hệ thống sẽ (shall) xác nhận charge number được nhập vào dựa trên master corporate charge number list trực tuyến. Nếu charge number không được tìm thấy trong list, hệ thống sẽ hiển thị một error message và lệnh đặt hàng (order) không được chấp nhận.”

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Một yêu cầu liên quan thứ hai có thể tài liệu hóa điều kiện loại trừ khi master corporate charge number list không sẵn sàng tại thời điểm cần xác nhận.

Ví dụ 5

“Sản phẩm không được đưa ra các options tìm kiếm và thay thế có thể dẫn tới các kết quả tai hại.”

Sự lưu ý “kết quả tai hại” là một diễn giải chủ quan. Thực hiện một thay đổi không chủ định khi soạn thảo một tài liệu có thể là tai hại nếu người dùng không phát hiện được lỗi hoặc không có cách sửa chữa lỗi. Bạn cũng phải sáng suốt khi sử dụng các yêu cầu theo hướng ngược lại, nghĩa là các yêu cầu mô tả những gì mà hệ thống sẽ không thực hiện.

IV. TÙ ĐIỂN DỮ LIỆU (THE DATA DICTIONARY)

Trước đây khá lâu, tôi làm việc trong một dự án mà vì một số lý do, 3 lập trình viên đã sử dụng 3 tên biến khác nhau, độ dài khác nhau, và các giá trị hợp lệ khác nhau cho cùng một mục dữ liệu (data item). Từ đó dẫn tới sự rắc rối về cái mà dữ liệu định nghĩa thực sự là gì không được biết, độ dài của thực sự của dữ liệu khi được lưu trữ trong một biến là bao nhiêu cũng không rõ. Chúng ta đã tự làm rối mình vì không có từ điển dữ liệu - một kho dữ liệu dùng chung xác định ý nghĩa, kiểu, kích thước và khuôn dạng của dữ liệu, đơn vị đo lường, độ chính xác, sự giới hạn được phép hoặc danh sách các giá trị của tất cả các phần tử dữ liệu và các cấu trúc được sử dụng trong ứng dụng.

Từ điển dữ liệu là chất keo gắn kết các tài liệu yêu cầu và các mô hình phân tích lại với nhau. Các vấn đề tích hợp (integration) được rút gọn nếu tất cả các nhà phát triển tuân thủ đúng đắn nội dung của từ điển dữ liệu. Để tránh sự dư thừa và sự không nhất quán, hãy thiết lập một từ điển dữ liệu riêng cho dự án của bạn, thay vì định nghĩa mỗi mục dữ liệu (data item) trong mỗi yêu cầu ngay khi mục dữ liệu xuất hiện. Duy trì từ điển dữ liệu riêng rẽ với SRS, tại một vị trí mà bất cứ stakeholder nào cũng có thể truy nhập tại bất cứ thời điểm nào trong quy trình phát triển và bảo trì hệ thống.

Các mục (item) trong từ điển dữ liệu được biểu diễn bằng cách sử dụng một ký pháp đơn giản (Robertson and Robertson 1994). Mục dữ liệu được đặt bên trái của một phuong trình, định nghĩa của mục dữ liệu được đặt bên phải. Ký pháp này định nghĩa các phần tử dữ liệu nguyên thủy, sự kết hợp của nhiều phần tử dữ liệu thành các cấu trúc và tất cả những gì liên quan đến dữ liệu. Các ví dụ dưới đây lấy từ nghiên cứu tình huống CTS.

Phần tử dữ liệu nguyên thủy. Một phần tử dữ liệu nguyên thủy là một phần tử dữ liệu không kết hợp từ các phần tử dữ liệu khác, không thể chia nhỏ được nữa. Nó có thể được gán một giá trị vô hướng. Định nghĩa phần tử nguyên thủy cần phải xác định kiểu dữ liệu, kích thước, dãy các giá trị được phép, ... Các phần tử nguyên thủy được định nghĩa bằng một chú thích (comment), với các đoạn văn ngắn cách nhau bởi dấu phẩy như sau:

*Request ID = *hệ thống 6 chữ số, được sinh từ các số tự nhiên tuần tự, bắt đầu với số 1, xác định duy nhất mỗi request**

Kết hợp (Composition). Một cấu trúc dữ liệu hoặc một bản ghi chứa nhiều mục dữ liệu. Nếu một phần tử trong cấu trúc dữ liệu là tùy chọn thì bao nó trong dấu ngoặc đơn:

*Requested Chemical = Chemical ID
+ Quantity
+ Quantity Unit
+ (Vendor Name)*

Cấu trúc này chứa tất cả thông tin liên quan đến một đề nghị về một hóa chất cụ thể. Vendor Name là tùy chọn vì người đưa ra đề nghị mua hóa chất có thể không quan tâm tới thông tin về người bán. Mỗi mục dữ liệu trong cấu trúc đều cần phải được định nghĩa trong từ điển dữ liệu. Các cấu trúc có thể tích hợp lại với nhau.

Lặp (Iteration)

Nếu nhiều thẻ hiện của một mục dữ liệu xuất hiện nhiều lần trong một cấu trúc thì hãy bao mục dữ liệu đó bằng một dấu ngoặc {}. Nếu đã biết trước số lượng được phép của lần lặp thì thẻ hiện trước dấu {} ký hiệu *minimum:maximum*:

*Request = Request ID
+ Charge Number
+ 1:10 {Requested Chemical}*

Ví dụ này thẻ hiện một chemical request phải chứa ít nhất 1 hóa chất nhưng không nhiều hơn 10. Mỗi đề nghị mua phải có duy nhất một request ID và một charge number mà khuôn dạng của nó đã được định nghĩa trong từ điển dữ liệu.

Phép lựa chọn (Selection)

Nếu một phần tử dữ liệu nguyên thủy có thẻ nhận một số lượng giới hạn các giá trị riêng rẽ thì có thể định nghĩa:

*Quantity Units =[“grams”/”kilograms”/”each”]
*chuỗi 9 ký tự chỉ đơn vị đo hóa chất**

Ví dụ trên cho thấy có 3 giá trị được lựa chọn để thẻ hiện đơn vị đo cho *Quantity Units*. Câu chú thích đưa ra định nghĩa không chính thức về phần tử dữ liệu.

Thời gian mà chúng ta bỏ ra để làm từ điển dữ liệu và bảng thuật ngữ sẽ được bù đắp nhanh chóng bằng thời gian tiết kiệm được do không phải các sửa các lỗi mắc phải của các thành viên dự án vì không chia sẻ chung một hiểu biết về các thông tin quan trọng. Nếu bạn giữ lại các bảng thuật ngữ và từ điển dữ liệu thì bạn có công cụ giá trị để bảo trì hệ thống và các hệ thống liên quan khác.

Các bước tiếp theo

- Trích ra một trang các yêu cầu chức năng từ SRS của dự án bạn đang làm. Với mỗi lời thẻ hiện yêu cầu, bạn hãy kiểm tra xem nó có tuân theo các đặc tính của yêu cầu tuyệt vời hay không. Hãy viết lại yêu cầu nào không tuân theo đặc tính đó.

- Nếu tổ chức của bạn không có một khuôn dạng tiêu chuẩn để tài liệu hóa các yêu cầu, hãy triệu tập một nhóm nhỏ để thiết lập một SRS template tiêu chuẩn. Bắt đầu với template trong Hình 9-1 và thích ứng nó với nhu cầu của tổ chức của bạn. Hãy thoả thuận một quy ước về cách đánh số các yêu cầu.
- Hãy tập hợp một nhóm từ 3 đến 7 stakeholders để soát xét chính thức SRS của dự án của bạn. Hãy chắc chắn rằng mỗi lời thẻ hiện yêu cầu là sáng sủa, khả thi, có thể kiểm tra, không nhập nhằng và cù thế. Hãy tìm kiếm các xung đột giữa các yêu cầu khác nhau trong SRS, tìm kiếm các yêu cầu bị khuyết, các mục bị khuyết của SRS. Hãy theo dõi để chắc chắn tất cả các khuyết khuyết mà bạn tìm thấy đều được sửa chữa và l่าน vết tới các bản thành phẩm từ các yêu cầu bị sửa chữa đó để thực hiện các chỉnh sửa tương ứng.

SATA-APTECH

CHƯƠNG 10

MỘT BỨC TRANH ĐÁNG GIÁ 1024 LỜI NÓI

Nhóm dự án CTS đã thực hiện phiên soát xét đầu tiên về SRS. Những người tham gia là Dave (trưởng dự án), Lori (phân tích yêu cầu), Helen (trưởng nhóm lập trình), Ramesh (trưởng nhóm kiểm thử), Tim (người trợ giúp sản phẩm của các nhà hóa học), Roxane (người trợ giúp sản phẩm của kho lưu trữ hóa chất). Tim bắt đầu nói, “Tôi đã đọc toàn bộ SRS. Phần lớn các yêu cầu có vẻ hợp lý trong cái nhìn của tôi, nhưng tôi có một số khó khăn khi tìm hiểu. Tôi không chắc lắm về việc liệu chúng ta đã xác định tất cả các bước cần thiết trong quy trình đề xuất mua hóa chất (chemical request process) hay chưa.”

Ramesh thêm vào, “Rất khó khăn cho tôi khi nghĩ về tất cả các test cases mà tôi sẽ cần để bao phủ được tất cả các thay đổi trạng thái (status changes) của một đề xuất như nó diễn ra trên thực tế. Tôi đã thấy một số yêu cầu có thể gây rắc rối cho SRS về các thay đổi trạng thái, nhưng tôi không thể nói liệu có yêu cầu nào bị khuyết hoặc không nhất quán hay không.”

Roxane cũng có một vấn đề tương tự, “Tôi đã rất bối rối khi đọc về cách mà tôi muốn đề xuất một hóa chất trên thực tế,” cô nói. “Từng yêu cầu một thì rất có ý nghĩa, nhưng tôi vẫn nghi ngờ việc xây dựng dây các bước mà tôi phải tuân theo.”

Sau khi những người khác đưa ra các ý kiến của họ, Lori kết luận, “Có vẻ như SRS không nói cho chúng ta biết về những thứ chúng ta cần thiết để từ đó hiểu hệ thống và đảm bảo chắc chắn chúng ta không bị mất một yêu cầu nào đó và gây ra những thiếu sót nào đó. Tôi sẽ vẽ một số sơ đồ để giúp chúng ta trực quan hóa các yêu cầu và xem liệu có làm sáng tỏ các vấn đề đó hay không. Cám ơn tất cả mọi người.”

Theo tác giả viết về yêu cầu Alan Davis, không một góc nhìn (view) duy nhất nào về yêu cầu có thể cho ta một sự hiểu biết đầy đủ về chúng (Davis 1995). Bạn cần kết hợp cách biểu diễn yêu cầu bằng lời nói và hình ảnh để vẽ nên một bức tranh

toàn cảnh về hệ thống đang chuẩn bị thực thi và giúp bạn phát hiện những gì thiếu nhất quán, nhập nhằng, các lỗi, và các thiếu sót. Các biểu diễn bằng hình ảnh, hoặc các mô hình phân tích, sẽ cải thiện sự hiểu biết của bạn về yêu cầu. Các sơ đồ (diagrams) trình bày được một số thông tin cho những người tham gia dự án hiệu quả hơn rất nhiều so với chữ viết, chúng có thể bắc được cây cầu vượt qua các rào chắn về ngôn ngữ và từ vựng giữa các thành viên khác nhau của nhóm (*nhất là các thành viên làm IT và các thành viên làm nghiệp vụ - ND*). Chương này cung cấp một cái nhìn khái quát về một số kỹ thuật mô hình hóa yêu cầu, tất cả đã giúp tôi rất nhiều trong việc hiểu được các vấn đề của người dùng và các nhu cầu phần mềm.

I. MÔ HÌNH HÓA YÊU CẦU (MODELING THE REQUIREMENTS)

Khi tôi bắt đầu vẽ các mô hình phân tích (analysis models) nhiều năm trước đây, tôi hy vọng sẽ tìm thấy một kỹ thuật giúp tôi gom tất cả mọi thứ vào một nơi có thể mô tả trọn vẹn về các yêu cầu. Cuối cùng tôi thấy rằng không có cái gì có thể làm được điều đó. Mục đích ngay từ đầu của các phương pháp phân tích hệ thống có cấu trúc là thay thế toàn bộ đặc tả chức năng theo kiểu cổ điển bằng các sơ đồ kiểu hình ảnh và các biểu diễn (representations) hình thức hơn so với chữ viết tự nhiên (DeMarco 1979). Tuy nhiên, kinh nghiệm đã chỉ ra rằng các mô hình phân tích chỉ bổ sung, chứ không thay thế, một đặc tả yêu cầu bằng ngôn ngữ tự nhiên (Davis 1995).

Các mô hình phô bày các góc nhìn bằng hình ảnh về yêu cầu, gồm DFD, ERD, STD (state – transition diagram), dialog maps, và class diagrams. Các cách tiếp cận mô hình hóa ít quy ước (conventional) hơn cũng có giá trị. Một nhóm dự án đã sử dụng một cách thành công một tool lập lịch biểu dự án để lập sơ đồ các yêu cầu thời gian cho một sản phẩm phần mềm nhúng, phần mềm này làm việc trong thang đo thời gian mili giây. Các mô hình này thường được dùng để phác thảo và duyệt (exploring) yêu cầu, cũng như để thiết kế các giải pháp phần mềm. Được sử dụng với tư cách là công cụ phân tích yêu cầu, các sơ đồ đó giúp bạn mô hình hóa miền bài toán (problem domain) hoặc tạo ra các biểu diễn ở mức ý tưởng của hệ thống mới. Các sơ đồ giúp các nhà phân tích và khách hàng đi đến một sự hiểu biết chung và sâu về yêu cầu, đồng thời cũng cho họ thấy các lỗi yêu cầu (requirement errors).

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Khi sử dụng các mô hình để phân tích yêu cầu hoặc để thiết kế bạn sẽ phụ thuộc vào 2 yếu tố là thời gian và hàm ý của việc mô hình hóa. Trong khi phát triển yêu cầu, bạn tạo ra các mô hình để chắc chắn rằng bạn hiểu yêu cầu. Các mô hình thể hiện các khía cạnh logic của các data components thuộc miền bài toán, các transactions và các transformations, các real-world objects, và các states được phép. Bạn có thể hoặc vẽ các mô hình từ các yêu cầu được thể hiện bằng chữ (textual requirements), hoặc bạn có thể dẫn xuất các yêu cầu chức năng từ các mô hình mà bạn vẽ dựa trên các đầu vào của người dùng. Trong khi thiết kế, bạn vẽ các mô hình vật lý, chứ không phải mô hình logic để trình bày các đặc tả mà bạn dự định sử dụng để thi công hệ thống: các cơ sở dữ liệu mà bạn đang tạo ra, các object classes mà bạn sẽ khởi tạo, và các code modules mà bạn sẽ phát triển.

Các kỹ thuật mô hình hóa phân tích mô tả trong chương này được hỗ trợ bởi nhiều công cụ thiết kế đã được thương mại hóa, các CASE tools. CASE tools cung cấp một số lợi ích thiết thực mà bạn nên tận dụng.

Các mô hình phân tích làm thuận tiện việc truyền thông giữa các thành viên dự án về một số khía cạnh của hệ thống. Bạn cũng có thể không cần thiết phải hoàn thành đầy đủ tập hợp các mô hình cho toàn bộ hệ thống. Hãy tập trung nỗ lực mô hình hóa của bạn vào các phần then chốt nhất và phức tạp nhất của hệ thống, các phần làm giảm sự nhập nhằng nhất và không chắc chắn nhất. Các ký pháp được giới thiệu ở đây cung cấp một ngôn ngữ chung để các thành viên dự án sử dụng, nhưng bạn cũng có thể sử dụng các sử dụng ít hình thức hơn để gia tăng việc truyền thông của dự án.

II. **TỪ TIẾNG NÓI CỦA KHÁCH HÀNG TỚI CÁC MÔ HÌNH PHÂN TÍCH (FROM VOICE OF CUSTOMER TO ANALYSIS MODELS)**

Bằng cách lắng nghe cẩn thận khách hàng trình bày các yêu cầu của họ, nhà phân tích có thể rút tóm ra các từ khóa để biến các yêu cầu thành các yếu tố của mô hình phân tích. Bảng 10-1 đề xuất các ánh xạ có thể có của các danh từ và động từ có ý nghĩa từ những gì khách hàng trình bày thành các model components cụ thể, chương sau sẽ mô tả kỹ hơn. Khi bạn đã phác ra đầu vào của khách hàng thành các yêu cầu thành văn và các mô hình, thì bạn cần phải lằn vết được giữa mỗi model component tới yêu cầu tương ứng.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

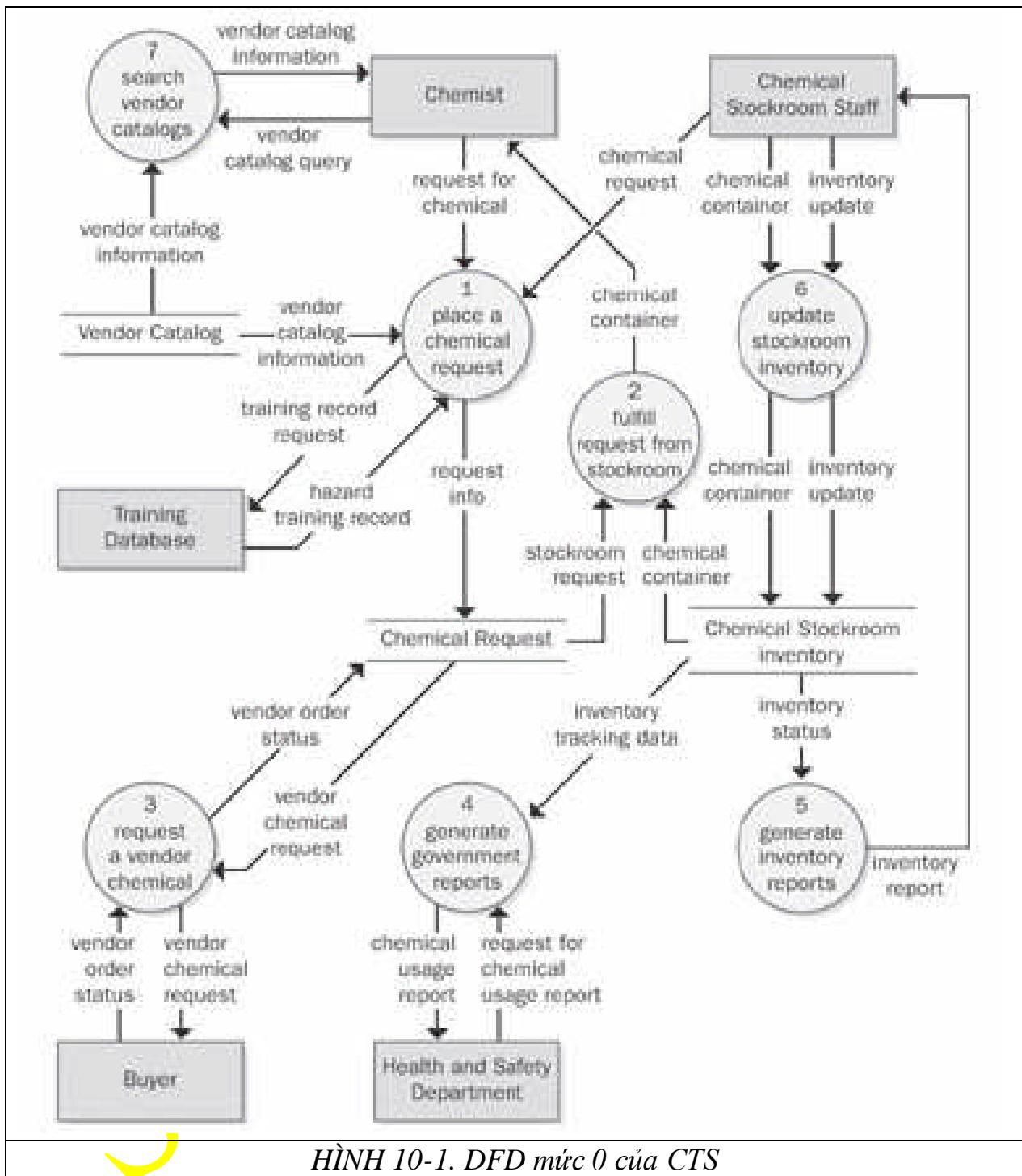
BẢNG 10-1: MỐI LIÊN QUAN GIỮA TIẾNG NÓI CỦA KHÁCH HÀNG VỚI
CÁC ANALYSIS MODEL COMPONENTS

KIỂU TỪ	VÍ DỤ	ANALYSIS MODEL COMPONENTS
Danh từ	Con người, tổ chức, các hệ thống phần mềm, mục dữ liệu, các objects	<ul style="list-style-type: none"> Các đầu cuối (terminators) hoặc data stores (DFD). Các entities hoặc các attributes của chúng (ERD). Các classes hoặc attributes của chúng (class diagram).
Động từ	Các hành động, những gì mà người dùng có thể làm, hoặc các events có thể xảy ra.	<ul style="list-style-type: none"> Các quy trình (DFD). Các relationships (ERD). Các transitions (STD). Các class operations (class diagram)

Trong toàn bộ cuốn sách này, tôi đã sử dụng CTS như một case study. Dựa trên ví dụ này, ta sẽ nghiên cứu các đoạn mô tả nhu cầu người dùng được cung cấp bởi đại diện sản phẩm của lớp các nhà hóa học. Các danh từ có ý nghĩa được để **chữ đậm**, các động từ **chữ nghiêng**.

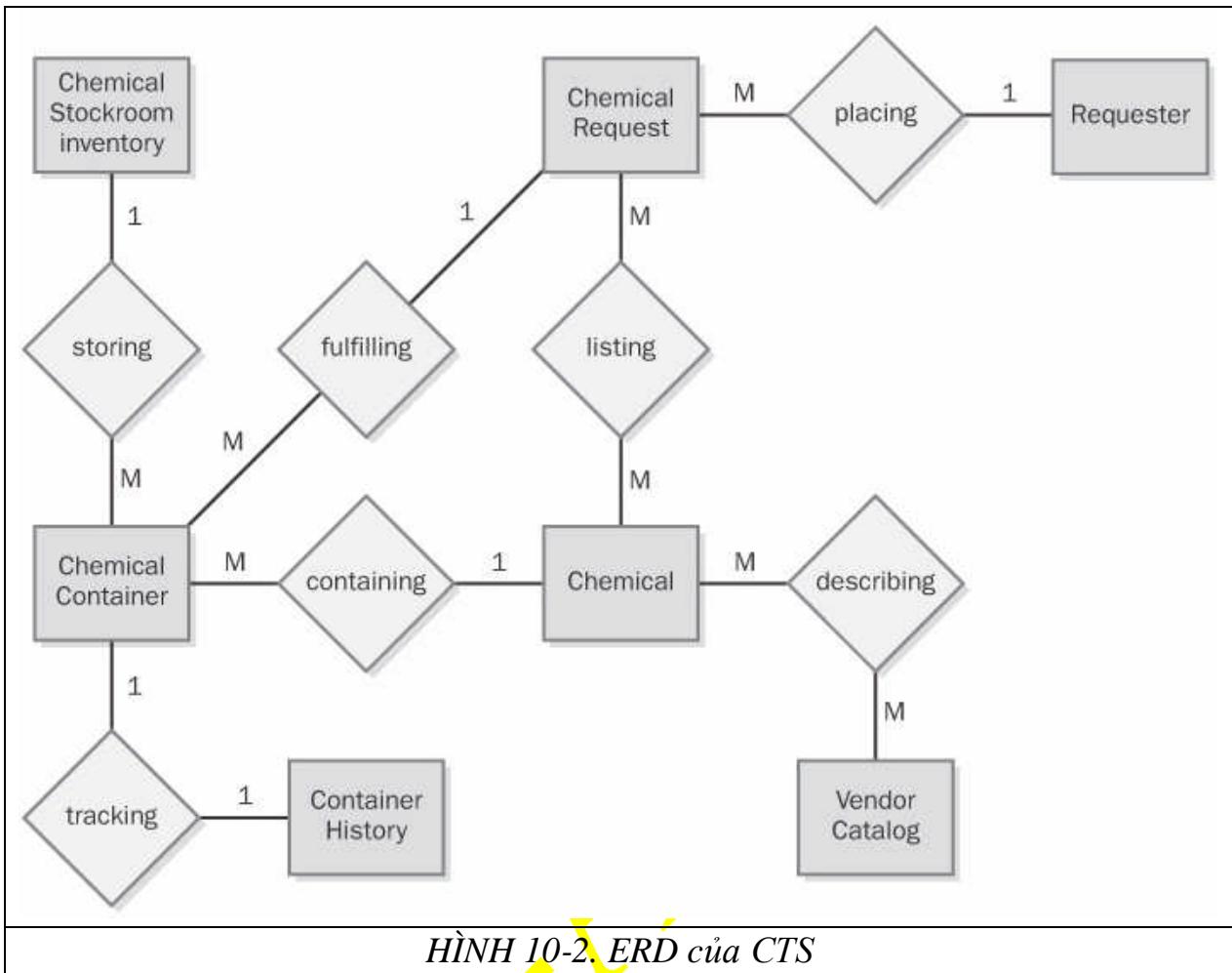
“Một **chemist** là một thành viên của **chemical stockroom staff** có thể **đặt** một **request** về một hoặc nhiều **chemicals**. Request có thể được **thực hiện** hoặc bằng cách **chuyển giao** một **container** hóa chất đã có trong **chemical stockroom's inventory** hoặc bằng cách **đặt** một **order** về một **container** hóa chất cần thiết từ một **vendor** bên ngoài. **Person** đặt request có thể **tìm kiếm** **vendor catalogs** trực tuyến **các** hóa chất cụ thể trong khi đang **chuẩn bị** request. Hệ thống cần phải **giám sát** **status** của mỗi request từ khi đang chuẩn bị cho đến khi hoặc được **thực hiện** hoặc bị **hủy bỏ**. Cũng cần phải **giám sát** **history** của mỗi container từ lúc **đến** **công ty** cho đến khi hết hóa chất hoặc vứt bỏ.

III. DFD



IV. ERD

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com



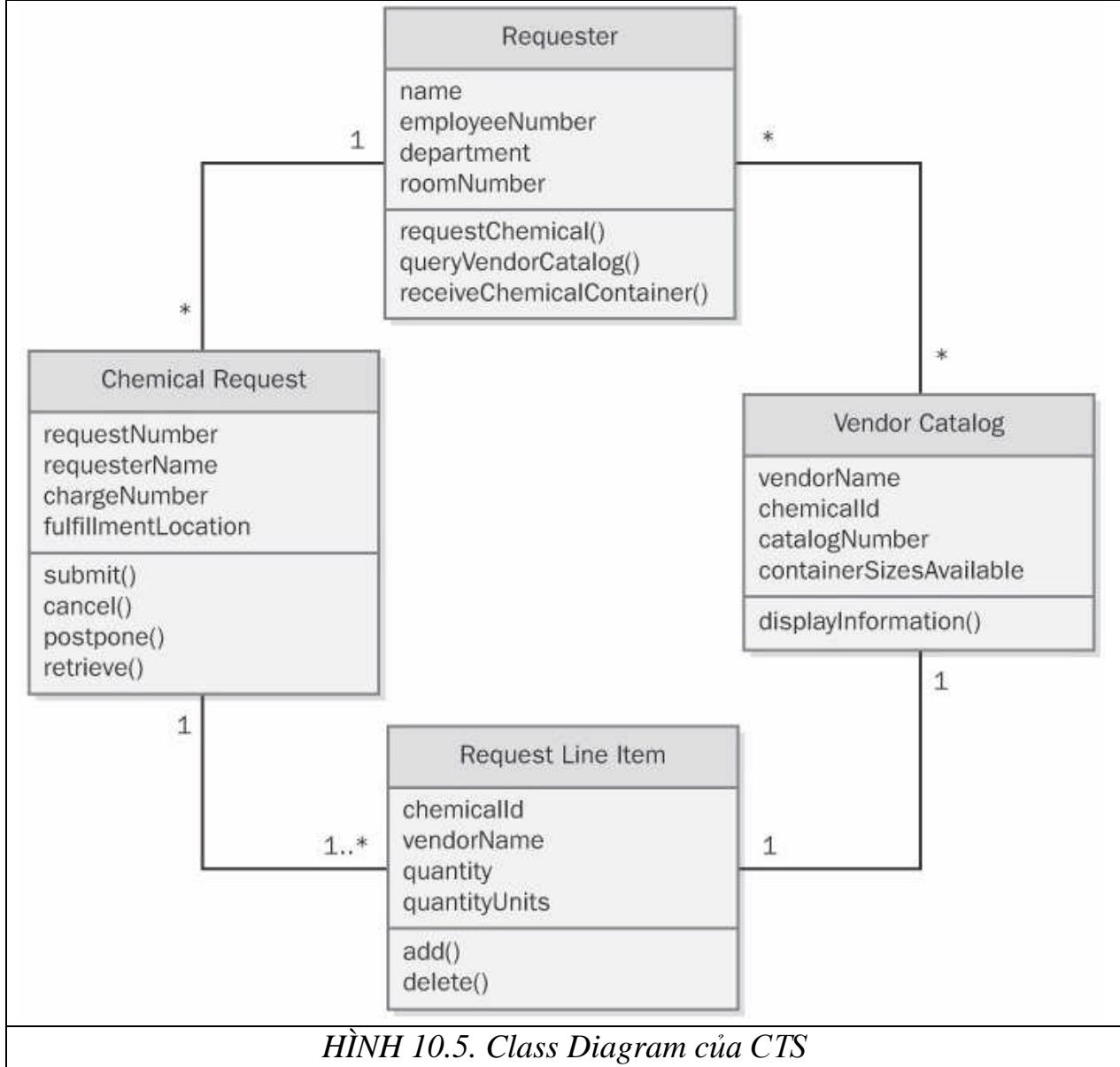
IV. CLASS DIAGRAMS

Phát triển phần mềm hướng đối tượng đã thế chỗ phân tích và thiết kế có cấu trúc trong nhiều dự án. Các *objects* thường tương ứng với các đối tượng trong thế giới thực được mô tả trong miền bài toán. Chúng biểu diễn cho các thể hiện cụ thể được dẫn xuất từ một template chung gọi là *class*. Các mô tả class bao gồm cả attributes (data) và các phép xử lý (method) có thể được thực hiện trên các attributes. Một *class diagram* là một cách sử dụng hình ảnh để mô tả các classes được định danh trong phân tích hướng đối tượng và quan hệ (relationships) giữa chúng.

Các sản phẩm được phát triển bằng cách sử dụng phương pháp hướng đối tượng không đòi hỏi các tiếp cận phát triển yêu cầu duy nhất. Đó là vì phát triển yêu cầu tập trung vào cái mà người dùng cần phải làm với hệ thống và các chức năng mà hệ thống phải có, chứ không quan tâm đến việc hệ thống sẽ được xây dựng như thế

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

nào. Người dùng không quan tâm đến các objects và các classes. Tuy nhiên, nếu bạn biết bạn đang xây dựng hệ thống bằng kỹ thuật hướng đối tượng, thì bạn cần người dùng giúp xác định rõ ràng các domain classes và các attributes cùng các behaviors tương ứng. Việc này sẽ thúc đẩy sự chuyển đổi thuận lợi từ phân tích sang thiết kế khi người thiết kế ánh xạ các objects của miền bài toán thành các objects của hệ thống và chi tiết hóa hơn các attributes và các operations của mỗi class. Dưới đây là class diagram của một phần CTS, Hình 10-5.



Chú ý có sự tương tự giữa các mô hình phân tích đã giới thiệu trong chương này.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

V. LỜI CUỐI

Mỗi kỹ thuật mô hình hóa được mô tả trong chương này đều có thuận lợi và giới hạn riêng. Hãy nhớ rằng bạn tạo ra các mô hình phân tích để cung cấp một mức độ hiểu biết và truyền thông khác về yêu cầu, bất cứ một mô hình phân tích riêng nào cũng đều không phản ánh hết tất cả những gì mà một hệ thống cần phải có.

Các bước tiếp theo

- Sử dụng các kỹ thuật mô hình hóa để mô tả bằng hình ảnh một bài toán nào đó bạn đang làm.
- Xác định một phần SRS bạn đang thực hiện và phần này khó cho người đọc nếu họ chỉ đọc yêu cầu về hệ thống bằng chữ viết. Hãy chọn một mô hình phân tích bằng hình ảnh nào đó để thể hiện phần này.

CHƯƠNG 11

CÁC THUỘC TÍNH CHẤT LƯỢNG PHẦN MỀM

Nhiều năm trước đây, tôi đã tham gia một dự án thay thế một ứng dụng chạy trên mainframe bằng một ứng dụng mới. Dựa trên đề xuất của người dùng, nhóm phát triển đã thiết kế một giao diện cửa sổ, các tệp dữ liệu mới được định nghĩa có dung lượng gấp đôi so với hệ thống cũ. Mặc dù hệ thống mới đáp ứng các đặc tả kỹ thuật, nhưng nó vẫn không được khách hàng chấp nhận. Người dùng phần mềm về hiệu năng (performance) kém cỏi của giao diện người dùng và không gian đĩa mà các tệp dữ liệu ngắn quá nhiều.

Người dùng có một số mong muốn không nói ra về các đặc tính của sản phẩm mới ngoài các yêu cầu chức năng đã mô tả. Rất không may là các nhà phát triển lại không thảo luận một cách rõ ràng về hàm ý hiệu năng có thể có của các cách tiếp cận kỹ thuật mới, do vậy mong muốn ngầm của khách hàng đã không được đáp ứng.

I. CÁC YÊU CẦU PHI CHỨC NĂNG (NONFUNCTIONAL REQUIREMENTS)

Người dùng tập trung đặc tả các yêu cầu chức năng, hay các yêu cầu về hành vi - những gì mà phần mềm sẽ cho phép họ thực hiện. Thêm nữa, thông qua đó, người dùng thường có những kỳ vọng không nói ra về việc sản phẩm sẽ làm việc *tốt như thế nào*. Các tính năng của sản phẩm dễ sử dụng như thế nào, thực hiện nhanh ra làm sao, đáng tin cậy đến đâu, hành vi ra sao khi các tình huống không mong đợi phát sinh liên tục. Các đặc tính như vậy được gọi là các thuộc tính chất lượng, hoặc các yếu tố chất lượng, chúng là một phần của các yêu cầu phi chức năng (hay phi hành vi).

Các thuộc tính chất lượng thường khó định nghĩa, thường có sự khác nhau giữa những gì mà một sản phẩm thực hiện đúng như định nghĩa về chức năng với một sản phẩm làm khách hàng vui lòng. Như Robert Charette (1990) đã chỉ ra, “Trong các hệ thống thực tế, việc đáp ứng các yêu cầu phi chức năng thường quan trọng hơn việc đáp ứng các yêu cầu chức năng khi xác định một sản phẩm là thành công”.

hay thất bại.” Các sản phẩm phần mềm tuyệt vời phản ánh một sự cân bằng tối ưu giữa các thuộc tính chất lượng xung đột nhau. Nếu bạn không tìm kiếm các kỳ vọng về chất lượng của khách hàng trong khi suy luận yêu cầu (requirement elicitation), thì bạn sẽ rất may mắn nếu sản phẩm làm hài lòng họ.

Khách hàng nói chung thường không tự nói ra các kỳ vọng về yêu cầu phi chức năng của họ, mặc dù thông tin mà họ cung cấp khi suy luận yêu cầu có thể giúp nhà phát triển lần ra một số đầu mối quan trọng về các đặc tính chất lượng. Sự nghiêm ngặt thường bị giảm đi khi khách hàng chỉ nói rằng họ muốn phần mềm phải “bền vững” (robust), “đáng tin cậy” (reliable), hoặc “hiệu quả” (efficient). Chất lượng, trong nhiều chiều kích của nó, cần được định nghĩa theo cả mong muốn của khách hàng và những gì mà nhà phát triển sẽ xây dựng, kiểm thử và bảo trì. Các câu hỏi khơi dậy được các kỳ vọng ngầm ý của người dùng có thể dẫn tới các phát biểu mục tiêu chất lượng và tiêu chuẩn thiết kế giúp nhà phát triển tạo ra một sản phẩm thỏa mãn đầy đủ khách hàng.

II. CÁC THUỘC TÍNH CHẤT LƯỢNG (QUALITY ATTRIBUTES)

Rất nhiều đặc tính có thể gọi là các thuộc tính chất lượng, mặc dù phần lớn các dự án chỉ quan tâm đến những gì hữu ích đối với khách hàng. Nếu nhà phát triển biết đặc tính nào trong số các đặc tính đó là then chốt nhất đối với sự thành công của dự án thì họ có thể chọn những cách tiếp cận theo quan điểm công nghệ phần mềm để hoàn thành các mục tiêu chất lượng đã xác định đó (Glass 1992; DeGrace and Stahl 1993). Các thuộc tính chất lượng đã được phân loại theo những hệ thống (scheme) khác nhau (Boehm 1976; DeGrace and Stahl 1993).

Bảng 11-1 liệt kê một số thuộc tính chất lượng thành 2 loại mà hầu hết các dự án đều quan tâm mặc dù cũng có những cách phân loại khác (Charette 1990). Một số thuộc tính là then chốt đối với các hệ nhúng (hiệu năng, độ tin cậy – efficiency, reliability), trong khi một số lại quan trọng đối với các ứng dụng trên mainframe (độ sẵn sàng, khả năng bảo trì – availability, maintainability), hoặc các ứng dụng desktop (khả năng liên vận hành, tính khả dụng – interoperability, usability). Trong một thế giới lý tưởng, mỗi hệ thống cần thể hiện tối đa các giá trị có thể có của mỗi thuộc tính chất lượng. Hệ thống cần phải sẵn sàng tại mọi thời điểm, không bao giờ bị lỗi, cung cấp kết quả ngay tức thì và luôn dễ sử dụng. Nhưng đòi hỏi như thế là Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

không thực tế, với mỗi dự án bạn chỉ cần lọc ra một tập con từ các thuộc tính chất lượng từ Bảng 11-1 được đánh giá là quan trọng nhất đối với dự án là đủ. Sau đó hãy định nghĩa các mục tiêu của người dùng và của nhà phát triển trong một quan điểm sao cho các nhà thiết kế sản phẩm có thể lựa chọn được các phương án đáp ứng thích hợp.

BẢNG 11-1. CÁC THUỘC TÍNH CHẤT LUỢNG PHẦN MỀM (Định nghĩa các thuộc tính sẽ trình bày dưới đây)	
CHỦ YÊU QUAN TRỌNG ĐỐI VỚI NGƯỜI DÙNG	CHỦ YÊU QUAN TRỌNG ĐỐI VỚI NHÀ PHÁT TRIỂN
Khả năng sẵn sàng (availability)	Khả năng bảo trì (Maintainability)
Hiệu năng (Efficiency)	Khả năng di chuyển (Portability)
Khả năng linh hoạt (Flexibility)	Khả năng sử dụng lại (Reusability)
Khả năng toàn vẹn (Integrity)	Khả năng kiểm thử (Testability)
Khả năng liên vận (Interoperability)	
Khả năng tin cậy (Reliability)	
Khả năng bền vững (Robustness)	
Khả năng sử dụng (Usability)	

Các thành phần khác nhau của sản phẩm có thể có sự kết hợp theo những cách khác nhau của các thuộc tính chất lượng mong muốn. Hiệu năng có thể là đòi hỏi then chốt cho một component nào đó, trong khi khả năng sử dụng lại là quan trọng đối với các components khác. Phân biệt các thuộc tính chất lượng áp dụng cho toàn bộ sản phẩm với các thuộc tính chỉ áp dụng cho một component, một số user classes hoặc một tình huống sử dụng cụ thể. Tài liệu hoá bất cứ yêu cầu thuộc tính toàn thể nào trong mục 5.4, kết hợp các thuộc tính cụ thể với các mô tả tính năng, use cases, hoặc các yêu cầu chức năng trong mục 4 của SRS template.

III. ĐỊNH NGHĨA CÁC THUỘC TÍNH CHẤT LUỢNG (DEFINING QUALITY ATTRIBUTES)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Bạn phải đặc tả các thuộc tính chất lượng trong quan điểm người dùng kỳ vọng như thế nào vào hành vi của hệ thống. Đặc tả một cách định lượng các thuộc tính chất lượng quan trọng để làm sáng tỏ hơn kỳ vọng của người dùng, chúng sẽ giúp người thiết kế lựa chọn được những giải pháp thích hợp nhất (Gilb 1998). Tuy vậy, phần lớn người dùng sẽ không biết làm thế nào để trả lời câu hỏi kiểu, “Khả năng liên vận hành quan trọng như thế nào đối với anh?” hoặc, “Khả năng tin cậy của phần mềm phải như thế nào?” Trong mỗi dự án, nhà phân tích đã xây dựng một số câu hỏi dẫn đường (prompting questions) căn cứ trên mỗi thuộc tính mà họ nghĩ sẽ quan trọng đối với một số lớp người dùng. Họ đề nghị một số đại diện của mỗi lớp người dùng xếp hạng mỗi thuộc tính trên thang từ 1 đến 5 (quan trọng nhất). Câu trả lời giúp nhà phân tích xác định những thuộc tính chất lượng nào là quan trọng nhất để sử dụng như một tiêu chuẩn thiết kế.

Khi đó, nhà phân tích làm việc với người dùng để phác nêu các yêu cầu cụ thể có thể đo lường, có thể kiểm thử cho mỗi đặc tính (Robertson and Robertson 1997). Nếu các mục tiêu chất lượng là không thể kiểm thử thì bạn sẽ không thể nói khi nào thì bạn đã hoàn thành chúng. Để có thể hoàn thành chúng, hãy chỉ ra thang bậc (scale) hoặc đơn vị (unit) đo lường cho mỗi thuộc tính và mục tiêu cần đạt của nó, các giá trị tối thiểu, tối đa. Nếu bạn không thể lượng hóa một số thuộc tính chất lượng quan trọng đối với dự án của bạn, ít nhất hãy định nghĩa mức ưu tiên cho nó. *IEEE standard for a Software Quality Metrics Methodology* trình bày một cách tiếp cận để định nghĩa các yêu cầu chất lượng phần mềm trong bối cảnh một quality metrics framework (hệ số đo chất lượng) tổng thể.

Một cách khác để định nghĩa một thuộc tính là đặc tả bất cứ hành vi hệ thống nào vi phạm các kỳ vọng chất lượng của bạn (Voas 1999). Bằng cách định nghĩa các hành vi không mong muốn - một kiểu yêu cầu ngược - bạn có thể nghĩ ra các tests với nỗ lực thúc đẩy hệ thống thực hiện các hành vi không mong muốn đó. Nếu bạn không thể thúc đẩy hệ thống thực hiện chúng, thì coi như bạn đã đạt được các mục tiêu chất lượng. Cách tiếp cận này đặc biệt có giá trị đối với các ứng dụng đòi hỏi mức độ an ninh cực cao, nơi mà nếu hệ thống vi phạm các ngưỡng về khả năng tin cậy hoặc hiệu suất (performance) thì sẽ gây ra một rủi ro nghiêm trọng đối với sự tồn tại của hệ thống.

Phần còn lại của chương này mô tả ngắn gọn mỗi thuộc tính trong Bảng 11-1 và gợi ý một số câu hỏi có thể giúp người dùng định vị các kỳ vọng của họ đối với thuộc tính đó. Một số phát biểu mẫu cũng được cung cấp, dẫn từ CTS hoặc từ các dự án khác, mặc dù các ví dụ đó khá đơn giản. Bạn cần phải lựa chọn cách tốt nhất để định vị mỗi yêu cầu thuộc tính chất lượng cho dự án của bạn để hướng dẫn nhà phát triển ra các quyết định thiết kế.

CÁC THUỘC TÍNH QUAN TRỌNG ĐỐI VỚI NGƯỜI DÙNG

1. **Khả năng sẵn sàng (availability)**

Khả năng sẵn sàng được quy về lượng phần trăm thời gian mà hệ thống sẵn sàng để sử dụng và vận hành đầy đủ. Một cách định lượng, khả năng sẵn sàng tương đương với thời gian trung bình bị sự cố (MTTF, mean time to failure) của hệ thống chia cho tổng của MTTF và thời gian trung bình để sửa chữa hệ thống sau khi một sự cố xảy ra. Một số tác vụ đòi hỏi mức độ then chốt về mặt thời gian (time – critical) hơn tất cả các yêu cầu khác và người dùng sẽ bị thiệt hại nếu họ muốn thực hiện tác vụ mà tác vụ không sẵn sàng tại thời điểm đó. Một yêu cầu về khả năng sẵn sàng có thể là: “Hệ thống cần phải sẵn sàng ít nhất 99,5% trong các ngày trong tuần từ 6 giờ sáng đến 12 giờ đêm theo giờ địa phương, và ít nhất 99,95% sẵn sàng từ 4 giờ chiều đến 6 giờ chiều theo thời gian địa phương.”

Tình huống thực tế: Chi phí cho chất lượng

Đặt mục tiêu 100% cho các thuộc tính chất lượng như khả năng tin cậy (reliability) hoặc khả năng sẵn sàng là không thể do chi phí quá đắt để đạt được điều đó. Một công ty có yêu cầu về khả năng sẵn sàng hệ thống sản xuất của họ là 100%, 24g mỗi ngày, 365 ngày mỗi năm. Để có thể đạt được mức độ nghiêm ngặt như vậy, công ty đã lắp đặt 2 hệ thống máy tính độc lập nhau sao cho các nâng cấp phần mềm có thể được cài đặt trên hệ thống máy tính này thì hệ thống máy tính kia sẽ điều khiển quá trình sản xuất. Đây là một giải pháp đắt đỏ để đạt được yêu cầu về khả năng sẵn sàng, nhưng chi phí cho giải pháp này thấp hơn là chi phí công ty phải bỏ ra để giải quyết vấn đề nếu không sản xuất được sản phẩm chất lượng tốt.

2. **Hiệu năng (Efficiency)**

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Hiệu năng là một số đo về cách làm thế nào hệ thống tận dụng được khả năng của bộ vi xử lý, không gian đĩa, bộ nhớ hoặc băng thông (Davis 1993). Nếu một hệ thống tiêu tốn tất cả các nguồn lực có thể thì người dùng sẽ phải đổi mặt với sự giảm hiệu suất, một chỉ số rõ ràng về sự không có hiệu năng. Hiệu suất kém có thể chỉ đơn giản là sự khó chịu (irritant) của người dùng khi họ đợi cơ sở dữ liệu truy vấn một kết quả, hoặc khi đổi mặt với rủi ro nghiêm trọng về an toàn, cũng như khi một hệ thống kiểm soát quy trình làm việc thời gian thực bị “tràn”. Để cho phép các hành lang an toàn (safety buffers) trong các điều kiện không tiên liệu được, bạn có thể đặc tả như sau: “10% công suất của bộ VXL sẵn sàng và 15% bộ nhớ của hệ thống sẵn sàng không được sử dụng tại các điểm tải cao nhất như đã lập kế hoạch.” Điều quan trọng là cần quan tâm đến các cấu hình phân cứng tối thiểu khi định nghĩa các mục tiêu hiệu suất (performance), công suất (capacity), hiệu năng.

Tình huống thực tế: Tôi không có cả một ngày để chờ đợi

Một công ty bán hàng trực tuyến đã thiết kế một biểu tượng bằng hình ảnh để khách hàng lưu trữ sản phẩm họ đã chọn mua. Khách hàng có thể vào trang mua hàng và lựa chọn một số sản phẩm, mỗi khi lựa chọn biểu tượng sẽ nhấp nháy và một âm thanh vui phát ra, sau đó sản phẩm lựa chọn của họ được đánh dấu lại. Giao diện này làm việc tốt trong thời gian phát triển do kết nối internet tốc độ cao tới các máy chủ của nhóm phát triển. Không may là khách hàng lại không sử dụng kết nối internet tốc độ cao nên khi lựa chọn hàng thì biểu tượng cứ chạy ì ạch và không có âm thanh nào phát ra cả, sản phẩm cũng không được đánh dấu lại. Công ty đã phải cho thiết kế lại trang web bán hàng sao cho với đường truyền dung lượng thấp, các tính năng cần thiết vẫn hoạt động.

3. Khả năng linh hoạt (Flexibility)

Khả năng này cũng được gọi là khả năng mở rộng (extensibility), khả năng tăng trưởng (augmentability), hoặc extendability, expandability. Tất cả các khái niệm này xác định sẽ mất bao nhiêu nỗ lực (effort) để thêm vào hệ thống một tính năng mới, một năng lực mới. Nếu nhà phát triển tiên đoán được các nâng cấp của hệ thống, họ có thể lựa chọn các cách tiếp cận thiết kế tối đa hóa được khả năng linh hoạt của phần mềm. Thuộc tính này là đòi hỏi cơ bản đối với các sản phẩm được phát triển theo cách thức lặp, tăng dần thông qua một loạt các phiên bản kế tiếp nhau. Trong một dự án graphics mà tôi đã từng làm việc trước đây, mục tiêu linh Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

hoạt được phát biểu: “Một lập trình viên phục vụ công tác bảo trì có ít nhất 6 tháng kinh nghiệm hỗ trợ sản phẩm thì có thể thêm một thiết bị in ấn mới vào hệ thống, bao gồm cả việc chỉnh sửa và kiểm thử mà không mất quá 1 giờ làm việc”. Nếu để thêm một máy tin vào hệ thống mà mất đến 75 phút thì khả năng linh hoạt của hệ thống phần mềm là không đạt yêu cầu.

4. Khả năng toàn vẹn (Integrity)

Khả năng toàn vẹn (hoặc an ninh) ngăn ngừa các truy nhập không được phép vào các chức năng hệ thống, tránh mất mát thông tin, đảm bảo phần mềm được bảo vệ trước nguy cơ nhiễm virus, bảo vệ tính riêng tư của dữ liệu được nhập vào hệ thống. Khả năng toàn vẹn đã trở thành mối quan tâm chính của ngành phần mềm từ khi có World Wide Web. Người dùng các hệ thống thương mại điện tử quan tâm đến việc bảo vệ thông tin credit card, những người lướt web thì không muốn thông tin cá nhân hoặc các hồ sơ về thông tin các sites mà họ ghé thăm được sử dụng một cách không thích hợp. Yêu cầu về khả năng toàn vẹn không chứa bất kỳ độ chịu lỗi nào (tolerance for error): dữ liệu và các truy nhập cần được bảo vệ đầy đủ theo mọi cách. Định vị các yêu cầu về khả năng toàn vẹn bằng các khái niệm không nhập nhằng: kiểm tra user ID, các mức ưu tiên của người dùng, các giới hạn truy nhập, hoặc dữ liệu đặc biệt cần phải được bảo vệ. Một yêu cầu về khả năng toàn vẹn mẫu: “Chỉ duy nhất những người dùng có quyền truy nhập Auditor mới được xem lịch sử giao dịch của khách hàng.”

5. Khả năng liên vận hành (Interoperability)

Khả năng này xác định mức độ dễ dàng của việc trao đổi dữ liệu hoặc dịch vụ giữa hệ thống của chúng ta với các hệ thống khác. Để đánh giá mức độ của khả năng này, bạn cần phải biết các hệ thống khác và dữ liệu nào được kỳ vọng sẽ trao đổi. Người dùng của CTS phải vẽ các cấu trúc hóa học bằng các công cụ thương mại khác, vì vậy mà yêu cầu về khả năng liên vận hành được phát biểu: “CTS phải nhập một cấu trúc hóa học bất kỳ từ ChemiDraw (*version 2.3 hoặc sớm hơn*) và Chem-Struct tools (*version 5 hoặc sớm hơn*).”

6. Khả năng tin cậy (Reliability)

Khả năng có thể thực thi mà không bị sự cố trong một khoảng thời gian cụ thể của phần mềm thì được gọi là khả năng tin cậy (Musa, Iannino, and Okumoto 1987). Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Khả năng bền vững (Robustness) và khả năng sẵn sàng đổi khi cũng được coi là một khía cạnh của khả năng tin cậy. Những cách để đo lường khả năng tin cậy là lượng phần trăm các phép xử lý được thực thi một cách đúng đắn, độ dài của thời gian mà hệ thống vận hành trước khi để lộ (revealing) một khiếm khuyết (defect) mới, và các mật độ của khiếm khuyết. Mô tả yêu cầu về khả năng tin cậy một cách định lượng dựa trên mức độ ảnh hưởng đối với công việc của khách hàng nếu một sự cố (failure) xảy ra và cân nhắc sao cho chi phí để tối đa hóa khả năng tin cậy là có thể chấp nhận. Nếu phần mềm thỏa mãn các yêu cầu về khả năng tin cậy thì có thể đặt vấn đề chuyên giao cho khách hàng thậm chí trong trường hợp phần mềm vẫn còn khiếm khuyết. Hệ thống đòi hỏi khả năng tin cậy cao cũng phải được thiết kế với những yêu cầu kiểm thử cao.

Nhóm của chúng tôi trước đây đã viết một số phần mềm để kiểm soát các thiết bị thí nghiệm được thực hiện dài ngày trong các điều kiện nghiêm ngặt về môi trường và các hóa chất đắt tiền. Người dùng đòi hỏi software component phải có khả năng tin cậy cao, trong khi các yêu cầu chức năng khác như ghi nhận dữ liệu nhiệt độ định kỳ thì ít nghiêm ngặt hơn. Một yêu cầu về khả năng tin cậy cho hệ thống cần phải được định vị kiểu: “Không lớn hơn 5 thí nghiệm trên 1000 thí nghiệm có thể bị mất dữ liệu do các sự cố phần mềm.”

7. Khả năng bền vững (Robustness)

Khả năng bền vững là mức độ mà hệ thống hoặc component vẫn tiếp tục thực hiện chức năng của nó một cách đúng đắn trong các trường hợp dữ liệu đầu vào sai, gặp các khiếm khuyết của phần mềm hoặc phần cứng kết nối vào, hoặc các điều kiện vận hành không mong đợi. Phần mềm bền vững che dấu một cách nhẹ nhàng các tình huống của bài toán và không bắt lỗi người dùng một cách quá chặt chẽ. Khi suy luận các mục tiêu của yêu cầu bền vững từ người dùng, hãy hỏi họ về các điều kiện gây lỗi đã biết mà hệ thống có thể phải đối mặt và người dùng mong muốn hệ thống đáp ứng lại như thế nào.

Trước đây tôi đã lãnh đạo một dự án phát triển một software component có thể sử dụng lại gọi là Graphics Engine, hệ thống xử lý các tệp dữ liệu mô tả các graphical plots và hoàn trả các plots trên một thiết bị ra đã định. Nhiều ứng dụng cần sinh ra các plots là đầu vào của Graphics Engine. Vì vậy chúng tôi muốn không có kiểm *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

soát nào trên dữ liệu đó, khả năng bền vững là cần thiết trong trường hợp này. 1 trong 4 yêu cầu về khả năng bền vững là: “Tất cả các tham số mô tả plot cần phải có khuôn dạng ngầm định từ trước, các khuôn dạng này được sử dụng nếu dữ liệu đầu vào bị khiếm khuyết hoặc không hợp lệ.” Ví dụ này phản ánh một cách tiếp cận thiết kế nhằm đạt được khả năng bền vững cho hệ thống, nơi mà “người dùng” của hệ thống là các ứng dụng khác.

8. Khả năng sử dụng (Usability)

Cũng được gọi là “dễ sử dụng” và “thân thiện người dùng”, khả năng sử dụng xác định rất nhiều yếu tố hình thành nên cái gọi là “thân thiện” đó. Khả năng này đo lường các nỗ lực cần thiết nhằm chuẩn bị đầu vào để xử lý và diễn dịch đầu ra của sản phẩm. Bạn có thể hiểu dễ sử dụng là dễ học cách vận hành hệ thống. Những nhà phân tích CTS hỏi người dùng các câu hỏi như: “Việc đặt các đề xuất (request) hoá chất một cách nhanh chóng và đơn giản, và xem thông tin quan trọng như thế nào đối với anh?” và “Tính trung bình, bao lâu thì anh nghĩ đến việc phải đề xuất sử dụng một hóa chất?” Các điểm bắt đầu đơn giản như vậy hướng về việc định nghĩa nhiều đặc tính khiến cho phần mềm dễ sử dụng. Thảo luận về khả năng sử dụng có thể dẫn tới các mục tiêu có thể đo lường như “Một người dùng được đào tạo có thể đề xuất một hóa chất được chọn từ một vendor catalog trong trung bình 3 phút, tối đa là 5 phút.”

Cũng vậy, hãy điều tra xem liệu hệ thống mới phải tuân theo chuẩn giao diện người dùng nào hoặc quy ước nào, hoặc liệu giao diện người dùng có nhất quán với giao diện của các hệ thống thường sử dụng khác hay không. Đây là một yêu cầu mẫu về khả năng sử dụng: “Tất cả các chức năng trên File menu cần phải có shortcut keys được định nghĩa bằng cách bấm kết hợp Control key đồng thời với một key khác. Menu commands cũng xuất hiện trên **Microsoft Word 2000 File menu** thì sử dụng cùng shortcut keys được sử dụng trong Word.”

Khả năng sử dụng cũng bao gồm mức độ dễ dàng học sử dụng hệ thống đối với những người mới hoặc những người không thường xuyên tiếp cận hệ thống. Mục tiêu dễ học cũng có thể được định lượng và đo lường, ví dụ: “Một người dùng mới cần phải đặt một request cho một hóa chất sẽ mất không quá 30 phút tự tìm hiểu” hoặc “Những người vận hành mới (operators) phải thực hiện 95% các tác vụ được Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

đòi hỏi của họ một cách đúng đắn sau một khóa đào tạo dài 1 ngày.” Khi bạn định nghĩa các yêu cầu về khả năng sử dụng hoặc khả năng học hỏi, hãy cân nhắc đến chi phí dành cho việc kiểm thử liệu các yêu cầu đó có được đáp ứng hay không.

CÁC THUỘC TÍNH QUAN TRỌNG ĐỐI VỚI NHÀ PHÁT TRIỂN

1. Khả năng bảo trì (Maintainability)

Khả năng bảo trì xác định mức độ dễ dàng khi sửa chữa một lỗi (defect) hoặc khi thực hiện một thay đổi trong phần mềm. Khả năng bảo trì phụ thuộc mức độ dễ dàng để hiểu, thay đổi, và kiểm thử phần mềm, như vậy là khả năng bảo trì liên quan chặt chẽ đến khả năng linh hoạt. Khả năng bảo trì mức độ cao là yếu tố then chốt để sản phẩm chịu được các cuộc soát xét (revision) định kỳ, để sản phẩm được xây dựng nhanh chóng (và có lẽ để đạt được chất lượng trong thời gian ngắn hơn). Bạn có thể đo lường khả năng bảo trì thông qua thời gian trung bình cần thiết để sửa chữa một vấn đề và lượng phần trăm các sửa chữa đó thành công.

CTS có các yêu cầu về khả năng bảo trì sau: “Các sửa chữa đối với các báo cáo đã có được thực hiện trong vòng 1 tuần lễ sau khi nhận được các quy định của chính phủ liên bang.”

2. Khả năng di chuyển (Portability)

Nỗ lực cần thiết để di chuyển một phần của phần mềm từ một môi trường vận hành này tới một môi trường vận hành khác là một số đo về khả năng di chuyển (*khả năng mà phần mềm có thể chạy trên nhiều môi trường nền khác nhau như Windows, Linux, desktop, laptop...*). Các tiếp cận thiết kế khiến phần mềm có khả năng di chuyển cũng tương tự với cách tiếp cận khiến phần mềm có khả năng sử dụng lại (Glass 1992). Khả năng di chuyển là điển hình đối với sự thành công của dự án hoặc một cách ngầm định hoặc về doanh thu của dự án. Các mục tiêu về khả năng di chuyển phải định vị các phần của sản phẩm có thể phải di chuyển tới môi trường khác và định danh các môi trường hoạt động mục tiêu đó. Các nhà phát triển có thể lựa chọn các cách tiếp cận thiết kế và mã hóa sẽ nâng cấp khả năng di chuyển của sản phẩm một cách thích hợp.

3. Khả năng sử dụng lại (Reusability)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Một mục tiêu dài hạn của phát triển phần mềm là khả năng sử dụng lại, đó là khả năng chỉ ra mức độ một software component có thể được sử dụng lại từ các ứng dụng khác thay vì phát triển mới hoàn toàn. Phát triển một component có khả năng sử dụng lại thì đắt hơn là tạo ra một component chỉ sử dụng duy nhất cho ứng dụng hiện thời. Phần mềm có khả năng sử dụng lại cần phải được mô-đun hóa, được tài liệu hóa tốt, độc lập với một ứng dụng cụ thể và môi trường vận hành, và một số yếu tố khác nữa (DeGrace and Stahl 1993). Đặc tả các yếu tố của hệ thống mới cần được xây dựng theo một cách thúc đẩy khả năng sử dụng lại, hoặc quy định hình thành nên thư viện về các components có thể sử dụng lại, thư viện này cần phải được tạo ra như một sản phẩm phụ (spin-off) của dự án.

4. Khả năng kiểm thử (Testability)

Khả năng kiểm thử mô tả mức độ dễ dàng của các components hoặc sản phẩm tích hợp có thể được kiểm thử để tìm kiếm các khiếm khuyết. Thiết kế để thuận tiện cho việc kiểm thử là yếu tố then chốt nếu sản phẩm có các thuật toán phức tạp, hoặc nếu nó chứa các mối quan hệ chức năng tinh tế. Khả năng kiểm thử cũng quan trọng nếu sản phẩm sẽ được chỉnh sửa thường xuyên, vì vậy sản phẩm phải chịu được sự kiểm thử lùi dần (regression testing) thường xuyên để xác định liệu các thay đổi có gây ra khiếm khuyết ở chức năng đang có nào hay không.

Do chúng tôi biết phải kiểm thử Graphics Engine nhiều lần vì được nâng cấp thường xuyên nên chúng tôi đề ra: “Độ phức tạp cyclomatic tối đa của một mô-đun không được vượt quá 20.” Độ phức tạp cyclomatic là một số đo về số lượng các nhánh logic (logic branches) trong một mô-đun mã nguồn (McCabe 1982). Thêm nhiều nhánh và vòng lặp vào một mô-đun sẽ khiến mô-đun khó kiểm thử hơn, khó hiểu và khó bảo trì hơn. Dự án sẽ không gặp sự cố nếu một mô-đun nào đó có độ phức tạp cyclomatic lớn hơn 20, nhưng các tài liệu, ví dụ tiêu chuẩn thiết kế, giúp các nhà phát triển đạt được một mục tiêu chất lượng đã được định nghĩa.

IV. CÂN BẰNG CÁC THUỘC TÍNH (ATTRIBUTE TRADE-OFFS)

Một số thuộc tính trái ngược nhau về bản chất vì vậy, trên thực tế, nếu thực hiện chúng thì sẽ xảy ra xung đột, do đó phải cân bằng chúng. Người dùng và nhà phát triển cần phải quyết định trong số các thuộc tính, cái nào quan trọng hơn cái nào, và họ cần phải tôn trọng sự ưu tiên đó một cách nhất quán khi ra quyết định. Hình Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

11-1 mô tả một số quan hệ điển hình giữa các thuộc tính chất lượng trong Bảng 11-1.

	Availability	Efficiency	Flexibility	Integrity	Interoperability	Maintainability	Portability	Reliability	Reusability	Robustness	Testability	Usability
Availability							+	+				
Efficiency		-	-	-	-	-	-	-	-	-	-	
Flexibility	-	-	-	+	+	+			+			
Integrity	-		-				-	-	-	-		
Interoperability	-	+	-		+							
Maintainability	+	-	+				+		+			
Portability	-	+	+	-			+		+	-		
Reliability	+	-	+		+			+	+	+		
Reusability	-	+	-	+	+	+	-		+			
Robustness	+	-					+			+		
Testability	+	-	+		+	+					+	
Usability		-						+	-			

Dấu cộng chỉ rằng nếu tăng mức độ của thuộc tính trong hàng tương ứng thì sẽ tạo ra hiệu ứng dương trên thuộc tính của cột tương ứng. Ví dụ nếu tăng mức độ khả năng sử dụng lại của một software component thì sẽ tăng mức độ linh hoạt, dễ kết nối với các components khác, dễ bảo trì, dễ di chuyển và dễ kiểm thử.

Dấu trừ chỉ ra rằng nếu tăng mức độ của thuộc tính trong hàng tương ứng thì sẽ tạo ra hiệu ứng âm trên thuộc tính của cột tương ứng. Hiệu năng (efficiency) tạo ra ảnh hưởng âm trên rất nhiều các thuộc tính khác. Nếu bạn viết các mã nguồn chặt chẽ nhất, nhanh nhất có thể, sử dụng một compiler và một HĐH đã định thì điều đó có nghĩa bạn sẽ rất khó bảo trì và nâng cấp, khó di chuyển tới các môi trường khác.

Để đạt được sự cân bằng tối đa các thuộc tính chất lượng của sản phẩm, bạn phải xác định, đặc tả, xếp thứ tự ưu tiên các thuộc tính chất lượng thích đáng trong khi suy luận yêu cầu. Khi bạn định nghĩa các thuộc tính chất lượng quan trọng trong

dự án của bạn, hãy sử dụng Hình 11-1 để tránh cam kết đạt được các mục tiêu xung đột nhau. Ví dụ:

- Đừng kỳ vọng tối ưu hóa được khả năng sử dụng (usability) nếu phần mềm phải chạy trên nhiều platforms (thể hiện khả năng di chuyển – portability).
- Phần mềm có khả năng sử dụng lại (reusability) có thể phải phổ cập hóa để sử dụng trên nhiều môi trường khác nhau, vì vậy không thể đạt được mức chịu lỗi cụ thể (specific error tolerance, tức khả năng tin cậy) hoặc khả năng toàn vẹn.

Đạt được mức cân bằng các thuộc tính chất lượng hợp lý trong phần mềm không phải mục đích tự thân. Chúng là kết quả từ các thảo luận về kỳ vọng của người dùng vào thuộc tính chất lượng khi suy luận yêu cầu, sau đó ghi thành tài liệu vào SRS.

Các bước tiếp theo

Xác định một số thuộc tính chất lượng từ Bảng 11-1 có thể quan trọng đối với người dùng của dự án của bạn. Hãy xây dựng một số câu hỏi về mỗi thuộc tính để người dùng định hình các kỳ vọng của họ. Dựa trên các câu trả lời của người dùng, hãy viết một hoặc hai mục tiêu cụ thể cho mỗi thuộc tính quan trọng.

CHƯƠNG 12

GIẢM RỦI RO THÔNG QUA LÀM NGUYÊN MẪU

Gần đây tôi đã làm việc với một nhóm dự án đang bị khách hàng từ chối sản phẩm hoàn chỉnh mà họ chuyển giao cho khách hàng. Người dùng không thích giao diện của phần mềm, họ cũng thấy có vấn đề đối với cả các yêu cầu ngầm ẩn khác. Kỹ thuật làm nguyên mẫu sẽ giúp bạn giảm bớt rủi ro từ sự từ chối của khách hàng. Phản hồi sớm giúp nhóm phát triển hiểu đúng các yêu cầu và biết cách làm thế nào để thi công chúng một cách tốt nhất.

Thậm chí nếu bạn ứng dụng các thực hành suy luận, phân tích, đặc tả yêu cầu được mô tả trong các chương trước, thì vẫn có những phần của yêu cầu không rõ ràng hoặc đối với người dùng hoặc đối với nhà phát triển. Nếu bạn không sửa chữa các yêu cầu này, thì một khoảng cách kỳ vọng có thể xuất hiện giữa tầm nhìn về sản phẩm (vision of product) của người dùng và sự hiểu biết của nhà phát triển về cái cần phải xây dựng. Rất khó để trực quan hóa một cách chính xác một phần mềm sẽ có hành vi như thế nào trong một hoàn cảnh cụ thể chỉ bằng cách đọc các yêu cầu trên giấy hoặc nghiên cứu các mô hình phân tích. Làm nguyên mẫu khiến cho sản phẩm trở nên hữu hình, mang lại sức sống cho các use cases, thu hẹp khoảng cách trong hiểu biết của bạn về các yêu cầu. Người dùng nói chung thường cảm thấy hứng thú hơn với các nguyên mẫu so với đọc bản SRS (nói chung là rất chán ngán!).

Nguyên mẫu có nhiều ý nghĩa, các bên tham gia làm nguyên mẫu có các kỳ vọng rất khác nhau. Ví dụ, một nguyên mẫu máy bay bay được – là phiên bản đầu tiên của một máy bay thực. Ngược lại, một nguyên mẫu phần mềm nói chung chỉ là một phần hoặc một mô hình của một hệ thống thực, và nó có thể chỉ làm được bất cứ cái gì. Chương này khảo sát các kiểu nguyên mẫu phần mềm khác nhau, các ứng dụng của chúng đối với việc phát triển yêu cầu, các cách khiến cho việc làm nguyên mẫu như một phần hiệu quả của quy trình công nghệ phần mềm (Wood and Kang 1992).

SATA-APTECH

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

I. LÀM NGUYÊN MẪU: CÁI GÌ VÀ TẠI SAO (PROTOTYPING: WHAT AND WHY)

Một nguyên mẫu phần mềm là sự thực thi một bộ phận (partial implementation) của một sản phẩm mới đang được đề xuất. Các nguyên mẫu được sử dụng với 3 mục đích chính sau đây:

- *Làm sáng tỏ và hoàn chỉnh các yêu cầu.* Được sử dụng như một công cụ làm yêu cầu (requirements tool), nguyên mẫu là sự thực thi sơ bộ một phần của hệ thống mà phần này không được hiểu rõ. Sự đánh giá của người dùng về nguyên mẫu sẽ chỉ ra các vấn đề đối với yêu cầu, chúng giúp bạn sửa chữa yêu cầu với chi phí thấp trước khi bạn xây dựng sản phẩm thực.
- *Duyệt qua các lựa chọn thiết kế.* Được sử dụng như một công cụ làm thiết kế (design tool), một nguyên mẫu giúp bạn duyệt qua các kỹ thuật giao diện người dùng khác nhau, tối ưu hóa khả năng sử dụng, đánh giá các tiếp cận kỹ thuật có thể có.
- *Hội tụ vào sản phẩm chính.* Được sử dụng như một công cụ thi công (construction tool), một nguyên mẫu là sự thực thi đầy đủ về mặt chức năng của một bộ phận sản phẩm, nguyên mẫu giúp bạn phác ra sản phẩm hoàn chỉnh thông qua một dãy các chu trình phát triển lặp.

Lý do chính để tạo một nguyên mẫu là nhằm phân giải ngay từ sớm những gì không chắc chắn trong chu trình phát triển. Căn cứ trên những gì không chắc chắn để quyết định bạn sẽ làm nguyên mẫu về bộ phận nào và cái gì mà bạn hy vọng học được từ các đánh giá của người dùng về nguyên mẫu. Một nguyên mẫu cũng là một cách tuyệt vời giúp làm lộ ra và phân giải các nhập nhằng của yêu cầu.

II. NGUYÊN MẪU NẰM NGANG VÀ NGUYÊN MẪU THẮNG ĐÚNG (HORIZONTAL AND VERTICAL PROTOTYPES)

Khi ai đó nói “nguyên mẫu phần mềm” thì họ thường nghĩ về một kiểu nguyên mẫu mà giới chuyên môn gọi là *nguyên mẫu nằm ngang*, nguyên mẫu này thể hiện một giao diện người dùng có thể có (possible user interface). Một nguyên mẫu nằm ngang cũng được gọi là *nguyên mẫu hành vi* (*behavioral prototype*) hoặc một *bản sao hình ảnh* (*mock-up*). Nguyên mẫu này giúp bạn duyệt qua một số hành vi cụ thể của hệ thống mong muốn, với mục đích làm mịn các yêu cầu. Nguyên mẫu khiến người dùng cảm thấy một cách hữu hình chức năng được cung cấp bởi hệ. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

thống mong muốn xem liệu nó có giúp họ thực thi được nghiệp vụ (business tasks) của họ hay không. Chú ý là loại nguyên mẫu này thường chỉ thể hiện chức năng cần có mà không bàn gì về việc thi công chức năng đó.

Một nguyên mẫu nằm ngang thì giống như một bộ phim. Nó thể hiện vẻ bề ngoài của màn hình giao diện người dùng, có thể cho phép một số mối liên kết (navigation) giữa chúng, nhưng chưa rất ít hoặc không chứa một chức năng thực tế nào. Một bản sao hình ảnh (mock-up) thể hiện cho người dùng thấy các lựa chọn chức năng và mối liên kết (navigational options) sẵn sàng trên các màn hình được làm nguyên mẫu (prototyped screens). Một số mối liên kết sẽ làm việc nhưng người dùng sẽ chỉ thấy một message mô tả cái sẽ thực sự được hiển thị tại điểm đó. Thông tin xuất hiện khi đáp ứng một truy vấn cơ sở dữ liệu là giả hoặc không thay đổi, nội dung của report được mã hóa cung.

Nguyên mẫu không thực hiện bất kỳ một công việc thông thường nào của người dùng, mặc dù trong nó có vẻ làm được những việc đó. Sự mô phỏng thường chỉ đủ cho người dùng thấy được chức năng đó liệu có bị thiếu, bị sai hoặc không cần thiết hay không. Nguyên mẫu trình bày ý tưởng của nhà phát triển về việc một use case sẽ được thực thi như thế nào. Sự đánh giá của người dùng về nguyên mẫu có thể chỉ ra các tiến trình thay thế (alternative courses) cho mỗi use case, các bước quy trình bị thiếu, các điều kiện loại trừ trước đó không phát hiện ra.

Khi bạn xây dựng được các nguyên mẫu như vậy ở một mức trừu tượng hợp lý thì người dùng có thể tập trung vào các chi tiết về luồng công việc (workflow) và yêu cầu (requirement) ở mức khái quát mà không bị lôi kéo vào sự trình bày các chi tiết trên màn hình (Constantine 1998). Sau khi bạn đã làm sáng tỏ các yêu cầu và xác định được cấu trúc chung của giao diện, bạn có thể tạo ra các nguyên mẫu chi tiết hơn để dựa vào đó mà duyệt qua các đặc tả về thiết kế giao diện người dùng. Bạn có thể tạo ra các nguyên mẫu nằm ngang bằng cách sử dụng một tập hợp các công cụ thiết kế màn hình đa dạng, thậm chí với giấy và bút chì như sẽ thảo luận sau đây.

Nguyên mẫu thẳng đứng hay cũng gọi là *nguyên mẫu cấu trúc* (*structural prototype*) hoặc *sự minh chứng của ý tưởng* (*proof of concept*) thực hiện một phần Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

nhỏ (slice) của chức năng. Xây dựng một nguyên mẫu thảng đứng nếu bạn không chắc rằng liệu cách tiếp cận kiến trúc được đề xuất là tốt hoặc bạn muốn tối ưu hóa các thuật toán, đánh giá một lược đồ cơ sở dữ liệu được đề xuất (proposed database schema) hoặc kiểm thử các yêu cầu định thời gian then chót (critical timing requirements). Nguyên mẫu thảng đứng thường được xây dựng bằng các công cụ sản xuất trong môi trường thực tế để kết quả có ý nghĩa thực tế. Nguyên mẫu thảng đứng được dùng để giảm rủi ro trong khi thiết kế phần mềm hơn là được dùng để phát triển yêu cầu.

Trước đây tôi đã làm việc với một dự án nơi mà tôi muốn thi công một kiến trúc Client/Server hiếm gặp (unusual) như là một phần công việc của chiến lược chuyên đổi từ môi trường hướng mainframe sang môi trường ứng dụng dựa trên networked Unix Server and workstations (Thompson and Wiegers 1995). Một nguyên mẫu thảng đứng đã được thi công và chỉ thực hiện một phần nhỏ chức năng của user interface client và server tương ứng, từ đó người dùng có thể đánh giá các communication components, hiệu năng, khả năng tin cậy của kiến trúc được đề xuất. Kinh nghiệm đó là một thành công của tôi về cách phát triển dựa trên kiến trúc.

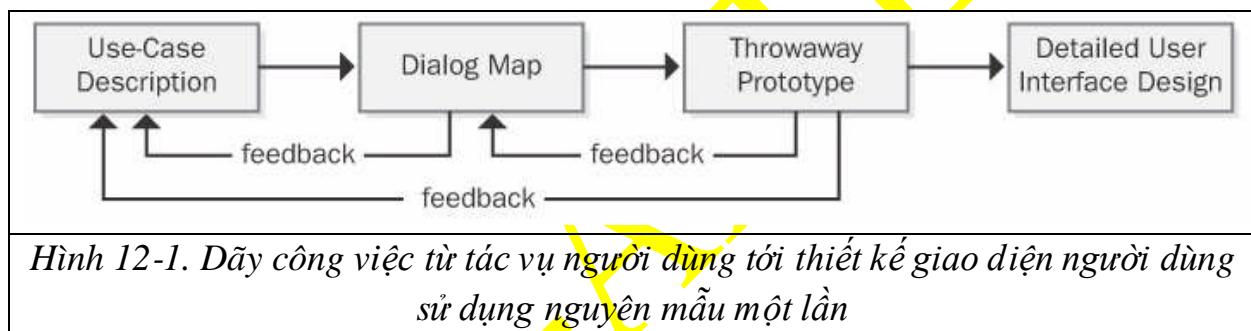
III. NGUYÊN MẪU MỘT LẦN VÀ NGUYÊN MẪU TIẾN HÓA (THROWAWAY AND EVOLUTIONARY PROTOTYPES)

Trước khi bạn xây dựng một nguyên mẫu, hãy quyết định rõ ràng và phổ biến rộng rãi liệu nguyên mẫu có bị vứt bỏ sau khi đánh giá hoặc sẽ tiến hoá thành một phần của sản phẩm (portion of product) hay không. Bạn xây dựng một *nguyên mẫu một lần* (hoặc *nguyên mẫu thăm dò – exploratory prototype*) để trả lời các câu hỏi, phân giải các bút định và cải tiến chất lượng yêu cầu (Davis 1993). Do bạn đã hàm ý vứt bỏ nguyên mẫu sau khi đánh giá, nên hãy xây dựng nguyên mẫu càng nhanh, càng rẻ thì càng tốt. Bạn càng đầu tư nhiều công sức vào nguyên mẫu thì những người tham gia dự án càng khó vứt bỏ nó.

Nếu bạn xây dựng một nguyên mẫu một lần, bạn sẽ không biết nhiều về các kỹ thuật xây dựng phần mềm bền vững (solid). Hãy nhấn mạnh việc thực thi nhanh và sửa đổi nhanh hơn là nhấn mạnh khả năng bền vững, khả năng tin cậy, hiệu năng, và khả năng bảo trì lâu dài. Vì lý do này, bạn không nên cho phép mã hóa từ một Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

nguyên mẫu một lần để nâng nó thành một sản phẩm trừ khi nó đáp ứng các tiêu chuẩn mã hóa theo quy định về chất lượng sản phẩm của bạn. Ngược lại, bạn và người dùng sẽ gánh chịu (suffer) các hậu quả trong suốt quãng đời của sản phẩm.

Nguyên mẫu một lần là thích hợp nhất nếu bạn phải đối mặt với sự nhập nhằng, bất định, không đầy đủ hoặc mờ ảo của yêu cầu. Bạn cần phải phân giải các chi tiết không hợp lý đó nhằm giảm bớt rủi ro khi bắt đầu thi công. Một nguyên mẫu giúp người dùng và nhà phát triển trực quan hóa việc các yêu cầu có thể được cài đặt như thế nào thì sẽ dễ phơi bày các điểm yếu của yêu cầu. Nguyên mẫu cũng tạo điều kiện để người dùng đánh giá liệu các yêu cầu đó có cần thiết để thực hiện các quy trình nghiệp vụ hay không.

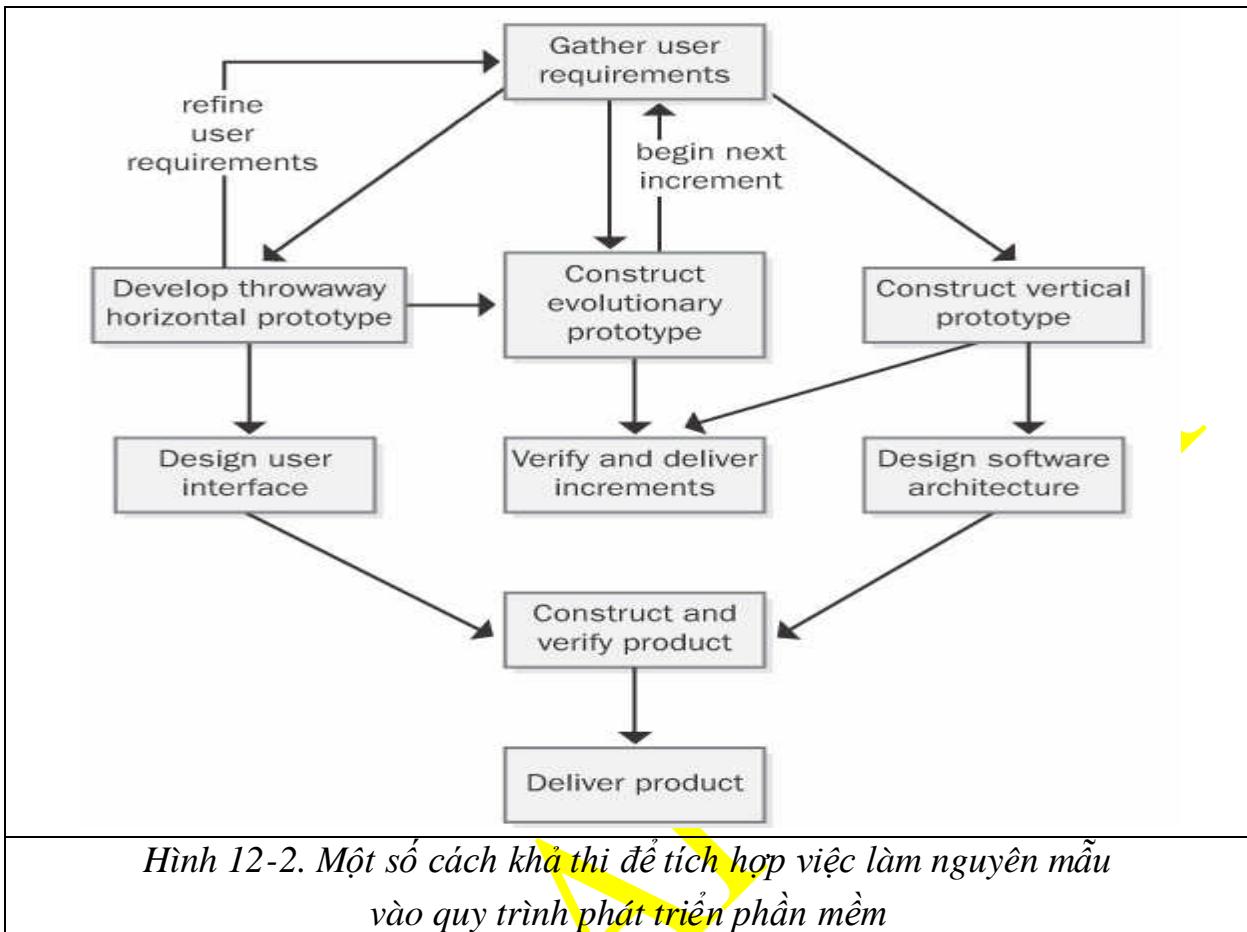


Hình 12-1 thể hiện một dãy các hoạt động phát triển bắt đầu từ các tác vụ của người dùng (use cases) tới thiết kế giao diện người dùng chi tiết bằng cách sử dụng nguyên mẫu một lần. Mỗi mô tả use case gồm một dãy các hoạt động của actor và đáp ứng của hệ thống, từ đó bạn có thể mô hình hóa bằng một dialog map để phác thảo một kiến trúc giao diện người dùng có thể có. Một nguyên mẫu một lần phác thảo (biến đổi) các dãy đối thoại (dialog elements) thành các màn hình cụ thể, các menus, các dialog boxes. Khi người dùng đánh giá nguyên mẫu, các phản hồi của họ có thể dẫn tới các thay đổi trong mô tả use case (ví dụ, nếu một tiến trình thay thế mới được phát hiện) và các thay đổi tương ứng trong dialog map. Trước khi các yêu cầu được làm mịn và các screens được phác thảo, bạn có thể thiết kế các chi tiết của mỗi yêu tố giao diện người dùng để đạt được khả năng sử dụng tối ưu. Cách tiếp cận làm mịn lùi dần thì rẻ hơn là nhảy trực tiếp từ mô tả use case tới một thực thi giao diện người dùng đầy đủ và phát hiện ra các lỗi lớn trong yêu cầu.

Ngược lại với nguyên mẫu một lần, một *nguyên mẫu tiến hóa* cung cấp một cơ sở kiến trúc vững chắc để xây dựng sản phẩm theo cách tăng dần khi các yêu cầu dần dần trở nên sáng sửa theo thời gian. Nguyên mẫu tiến hoá là một component của mô hình phát triển lặp (Boehm 1988) và của một số quy trình phát triển phần mềm hướng đối tượng (Kruchten 1996). Ngược lại với bản chất dùng-xong-là-vứt đi của nguyên mẫu một lần, một nguyên mẫu tiến hoá cần phải được xây dựng một cách vững chắc. Vì vậy, một nguyên mẫu tiến hoá sẽ mất nhiều thời gian hơn để xây dựng so với một nguyên mẫu một lần, mặc dù cả hai cùng muốn chỉ mặt đặt tên cùng một số chức năng. Một nguyên mẫu tiến hoá cần được thiết kế sao cho dễ dàng tăng trưởng và nâng cấp thường xuyên, vậy bạn cần chú trọng các nguyên lý thiết kế bền vững và kiến trúc phần mềm. Không có lối tắt cho chất lượng trong khi làm nguyên mẫu tiến hoá.

Hãy nghĩ rằng vòng lặp đầu tiên của một nguyên mẫu tiến hoá chính là một phiên bản thử nghiệm (pilot release) cài đặt phần yêu cầu đã được hiểu biết tốt và ổn định. Các kinh nghiệm rút ra từ quá trình sử dụng và kiểm thử phiên bản đầu tiên này sẽ được dùng để chỉnh sửa vòng lặp tiếp theo, cứ thế tiến dần tới vòng lặp cuối cùng - chính là sản phẩm trọn vẹn – thông qua một dãy các nguyên mẫu.

Thực hiện nguyên mẫu tiến hoá là rất phù hợp với các dự án phát triển Web. Trong một dự án Web mà tôi đã quản lý, chúng tôi đã tạo ra một dãy bao gồm 4 nguyên mẫu, dựa trên các yêu cầu mà chúng tôi đã phát triển từ một phân tích use case. Một số người dùng đánh giá mỗi nguyên mẫu, căn cứ câu trả lời đối với các câu hỏi mà chúng tôi đã đặt ra, chúng tôi xem xét lại nguyên mẫu. Các chỉnh sửa sau sự đánh giá nguyên mẫu thứ tư dẫn tới kết quả sản phẩm Web cuối cùng.



Hình 12-2. Một số cách khả thi để tích hợp việc làm nguyên mẫu vào quy trình phát triển phần mềm

Hình 12-2 mô tả một số cách mà bạn có thể kết hợp các kiểu làm nguyên mẫu khác nhau thành quy trình phát triển phần mềm của bạn. Ví dụ, bạn có thể sử dụng tri thức thu được từ một dãy các nguyên mẫu một lần để làm mịn yêu cầu, các yêu cầu này lại được thực thi tăng dần thông qua một loạt nguyên mẫu tiến hóa. Một cách khác là bạn sử dụng một nguyên mẫu nằm ngang một lần (throwaway horizontal prototype) để làm sáng tỏ các yêu cầu nhằm chốt lại (finalizing) thiết kế giao diện người dùng, đồng thời một nguyên mẫu thẳng đứng (vertical prototyping) cũng được phát triển để xác nhận (validate) kiến trúc, các components và các thuật toán lõi.

Bảng 12-1 tóm tắt một số ứng dụng tiêu biểu của các nguyên mẫu một lần, tiến hóa, nằm ngang và thẳng đứng.

BẢNG 12-1. MỘT SỐ ỨNG DỤNG TIÊU BIỂU CỦA CÁC NGUYÊN MẪU		
	MỘT LẦN	TIẾN HÓA
NĂM	<ul style="list-style-type: none"> Làm sáng tỏ và làm mịn các 	<ul style="list-style-type: none"> Thi công các use cases lõi.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

NGANG	use cases, các yêu cầu chức năng. <ul style="list-style-type: none"> Xác định các chức năng bị khuyết. Duyệt qua các cách tiếp cận về giao diện người dùng. 	<ul style="list-style-type: none"> Thi công các use cases phụ dựa trên mức độ ưu tiên. Phát triển và làm mịn các Web sites.
THẮNG ĐÚNG	<ul style="list-style-type: none"> Chứng minh sự khả thi về mặt kỹ thuật. 	<ul style="list-style-type: none"> Thi công và tăng trưởng lặp chức năng client/server lõi và các communication layers. Thi công và tối ưu hóa các thuật toán lõi.

IV. NGUYÊN MẪU TRÊN GIẤY VÀ NGUYÊN MẪU ĐIỆN TỬ (PAPER AND ELECTRONIC PROTOTYPES)

Trong nhiều trường hợp, không cần thiết một nguyên mẫu có thể thực thi (executable prototype) để suy luận thông tin mà bạn cần nhằm phân giải các nhầm lẫn của yêu cầu. Nguyên mẫu giấy (đôi khi còn gọi là lo-fi prototype) sẽ rẻ, nhanh, đơn giản để duyệt qua những gì mà một phần của hệ thống sau khi được thực thi sẽ trông như thế nào (Rettig 1994; Hohmann 1997). Các nguyên mẫu giấy giúp bạn kiểm thử liệu người dùng và nhà phát triển có đạt được một hiểu biết chung về yêu cầu hay không. Chúng cho phép bạn từng bước một ít rủi ro đi đến một không gian giải pháp chung trước khi mã hóa sản phẩm.

Các nguyên mẫu giấy chỉ gồm các công cụ như giấy, thẻ chỉ dẫn (index cards), giấy ghi chú, kẹp giấy, bảng và bút đánh dấu. Bạn phác thảo ý tưởng về hình dáng của màn hình mà không cần phải bắn khoan về sự chính xác của các buttons hay widgets. Người dùng sẽ cung cấp phản hồi và bạn có thể sửa trên giấy luôn. Một tập hợp các screens được phác ra trên nhiều tờ giấy sẽ giúp bạn và người dùng hình dung toàn bộ màn hình của sản phẩm.

Nếu không thích dùng nguyên mẫu giấy, bạn có thể xây dựng một nguyên mẫu một lần điện tử (electronic throwaway prototype) bằng công cụ sau:

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- Các ngôn ngữ lập trình như MS Visual Basic, IBM VisualAge SmallTalk, Inprise Delphi.
- Các ngôn ngữ scripting như Perl, Python, Rexx.
- Các bộ công cụ thực hiện nguyên mẫu được thương mại hóa.

V. ĐÁNH GIÁ NGUYÊN MẪU (PROTOTYPE EVALUATION)

Bạn có thể cải tiến tính hiệu quả của việc đánh giá nguyên mẫu bằng cách tạo ra các kịch bản (scripts) hướng dẫn người dùng thông qua một dãy các bước và các câu hỏi cụ thể để suy luận thông tin mà bạn cần. Hoạt động này là sự hỗ trợ có giá trị cho mục tiêu “hãy nói tôi biết bạn cảm thấy như thế nào về nguyên mẫu này.” Dẫn xuất các kịch bản đánh giá từ các use cases hoặc chức năng đã được định vị trong nguyên mẫu. Kịch bản cần phải đề nghị người dùng thực hiện các tác vụ cụ thể và “lái” họ đi qua các phần của nguyên mẫu về hướng mà bạn không chắc chắn nhất. Sau mỗi tác vụ, kịch bản cần phải hỏi người đánh giá (evaluator) các câu hỏi liên quan đến tác vụ cụ thể. Ngoài ra, bạn cũng phải hỏi các câu hỏi tổng quan sâu:

- Nguyên mẫu này có thực thi các chức năng theo cách mà anh kỳ vọng?
- Có chức năng nào bị thiếu không?
- Anh có thể nghĩ về một tình huống lỗi nào đó (any error conditions) mà nguyên mẫu không chỉ ra không?
- Có chức năng không cần thiết nào được giới thiệu không?
- Các liên kết (navigation) mà anh thấy có logic và đầy đủ không?
- Anh có thể nghĩ về những cách dễ hơn để thực thi tác vụ này không?

Hãy chắc chắn rằng những người thích hợp đánh giá nguyên mẫu từ những hoàn cảnh thích hợp. Người đánh giá nguyên mẫu phải là đại diện của cộng đồng người dùng kỳ vọng. Hãy đưa vào nhóm đánh giá những người dùng có kinh nghiệm và không có kinh nghiệm từ các lớp người dùng sẽ sử dụng sản phẩm. Trước khi bạn giới thiệu nguyên mẫu cho những người đánh giá, hãy nhấn mạnh rằng nguyên mẫu không chứa tất cả các nghiệp vụ đang có, các nghiệp vụ đó sẽ được cài đặt khi phát triển sản phẩm.

Bạn sẽ học hỏi được nhiều hơn khi quan sát người dùng làm việc với nguyên mẫu so với việc chỉ đơn giản đề nghị họ đánh giá nguyên mẫu theo ý kiến của họ và nói

cho bạn biết những gì họ nghĩ. Các kiểm thử về khả năng sử dụng chính thức (formal usability tests) của các nguyên mẫu giao diện người dùng là cần thiết nhưng bạn cũng có thể học hỏi được nhiều chỉ bằng cách quan sát. Hãy quan sát nơi những ngón tay của người dùng cố gắng dò đếm theo bản năng. Các vị trí cần lưu ý (spot places) là vị trí mà cách thiết kế của nguyên mẫu không giống với thiết kế của các ứng dụng khác mà người dùng đang sử dụng. Hãy quan sát những khi người dùng nhăn trán vì có thể khi đó họ không biết làm cách nào để thực hiện được nghiệp vụ mong muốn.

Ghi lại thành tài liệu những gì bạn học hỏi được từ cuộc đánh giá nguyên mẫu. Đối với một nguyên mẫu nằm ngang, hãy sử dụng thông tin bạn thu thập được để làm mịn các yêu cầu trong SRS. Nếu cuộc đánh giá nguyên mẫu dẫn tới một số quyết định thiết kế giao diện người dùng hoặc lựa chọn một số kỹ thuật tương tác cụ thể, hãy ghi lại các kết luận đó và làm thế nào để đạt được chúng. Với các nguyên mẫu thẳng đứng, hãy ghi lại các đánh giá mà bạn thực hiện và các kết quả của chúng, từ đó bạn nhận định được về khả năng tồn tại (viability) của tiếp cận kỹ thuật của bạn.

VI. RỦI RO LỚN CỦA VIỆC LÀM NGUYÊN MẪU (THE BIG RISK OF PROTOTYPING)

Làm nguyên mẫu là một kỹ thuật giúp giảm bớt rủi ro thất bại của một dự án phần mềm. Tuy nhiên, nguyên mẫu cũng có rủi ro của nó. Rủi ro lớn nhất là một người dùng hoặc một nhà quản lý xem nguyên mẫu chạy và kết luận sản phẩm đã gần hoàn thành. “Ô, trông như sắp xong rồi đây nỉ!” người đánh giá thốt nên như vậy. “Trông có vẻ tốt rồi đấy. Anh có thể bàn giao sản phẩm chưa?”

Bạn trả lời một từ: KHÔNG! Nếu bạn đang mô tả hoặc đánh giá một nguyên mẫu một lần, sẽ chỉ có vấn đề gì nếu nguyên mẫu một lần trông giống như thật. Đó chỉ là một mô hình, một mô phỏng. Kỳ vọng của stakeholders là yếu tố chính ảnh hưởng đến sự thành công của nguyên mẫu, vì vậy hãy chắc chắn họ xem và hiểu vai trò của nguyên mẫu. Hãy từ chối áp lực chuyển giao nguyên mẫu một lần. Chuyển giao nguyên mẫu sẽ gây ra sự chậm tiến độ của dự án như đã định, do khi thiết kế và mã hóa nguyên mẫu bạn đã không quan tâm đến chất lượng hoặc độ bền vững. Để tránh áp lực chuyển giao nguyên mẫu, bạn có thể sử dụng nguyên mẫu giấy thay vì nguyên mẫu điện tử.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Hãy tiếp tục kiểm soát các kỳ vọng của người dùng khi đánh giá nguyên mẫu. Nếu họ thấy nguyên mẫu đáp ứng tức thì một truy vấn cơ sở dữ liệu thì họ sẽ nghĩ rằng sản phẩm thực cũng như thế. Hãy tạo một độ trễ thời gian để người dùng nghĩ chính xác hơn về hành vi kỳ vọng của sản phẩm.

VII. CÁC YẾU TỐ THÀNH CÔNG CỦA VIỆC LÀM NGUYÊN MẪU (PROTOTYPING SUCCESS FACTORS)

Làm nguyên mẫu là một kỹ thuật rất hữu ích giúp bạn rút ngắn lịch biểu phát triển, tăng cường đáp ứng mong muốn của khách hàng, sản xuất sản phẩm với chất lượng cao, giảm bớt các lỗi yêu cầu và khiêm khuyết của giao diện người dùng.

Để việc làm nguyên mẫu trở thành một phần hiệu quả trong quy trình phát triển yêu cầu, bạn có thể tuân theo các hướng dẫn sau:

- Đưa các công việc làm nguyên mẫu vào kế hoạch dự án của bạn. Lập lịch biểu để phát triển, đánh giá, chỉnh sửa nguyên mẫu.
- Lập kế hoạch phát triển nhiều loại hình nguyên mẫu, do bạn hiếm khi nhận được ý kiến đúng của khách hàng trong lần đầu tiên.
- Tạo ra các nguyên mẫu một lần càng nhanh, càng rẻ thì càng tốt. Đầu tư nguồn lực tối thiểu khi phát triển các nguyên mẫu chỉ để trả lời một số câu hỏi, hoặc phân giải các bất định của yêu cầu. Đừng cố gắng làm hoàn hảo giao diện người dùng của nguyên mẫu một lần.
- Đừng bao gồm các chú thích mã nguồn, các xử lý hợp lệ hóa dữ liệu đầu vào, các kỹ thuật mã hóa bảo vệ, các mã xử lý lỗi trong nguyên mẫu một lần.
- Đừng thực hiện nguyên mẫu về các yêu cầu mà bạn đã hiểu rõ.
- Chống lại sự quyền rũ - hoặc áp lực của người dùng - để thêm nhiều chức năng vào nguyên mẫu. Đừng để một nguyên mẫu một lần đơn giản vượt quá mục tiêu giúp bạn hiểu hơn một số yêu cầu nào đó.
- Đừng để người dùng luận ra bất cứ điều gì về hiệu năng của sản phẩm từ hiệu năng của nguyên mẫu nằm ngang. Nguyên mẫu không thể chạy trong môi trường sản xuất đã định, và bạn có thể tạo ra nguyên mẫu bằng các công cụ khác về hiệu quả hoạt động so với hệ thống thực được tạo ra bằng các công cụ sản xuất thực.

- Sử dụng dữ liệu giả hợp lý trong các thẻ hiện màn hình và báo cáo của nguyên mẫu. Người dùng đánh giá nguyên mẫu có thể bị rối trí vì dữ liệu không thực và thất bại nếu tập trung vào nguyên mẫu như một mô hình của hệ thống thực.
- Đừng kỳ vọng nguyên mẫu thay thế được các yêu cầu được viết thành văn. Rất nhiều chức năng “ẩn đằng sau sân khấu” và chỉ lộ ra trên nguyên mẫu, bạn hãy thu thập các yêu cầu này và hoàn thiện SRS.

SATA-APTECH

Các bước tiếp theo

- Chọn ra một phần dự án của bạn, ví dụ một use case đang chịu các phiền phức về yêu cầu. Hãy phác ra một giao diện người dùng có thể có của use case thể hiện hiểu biết của bạn về yêu cầu và hãy sử dụng một nguyên mẫu giấy chằng hạn. Một số người dùng sẽ xem xét nguyên mẫu để mô phỏng một tác vụ hoặc mô phỏng hoạt động của use case. Hãy xác định bất cứ vị trí nào mà hiểu biết ban đầu về yêu cầu là không đầy đủ hoặc không đúng đắn. Chính sửa nguyên mẫu theo những quan sát đó.
- Trình bày một tổng kết về chương này cho những người đánh giá nguyên mẫu của bạn. Điều này sẽ giúp họ hiểu được lý do tại sao phải làm nguyên mẫu và có những kỳ vọng thực tế hơn về các đánh giá nguyên mẫu.

SATA-APTECH

CHƯƠNG 13

THIẾT LẬP CÁC ƯU TIÊN CỦA YÊU CẦU

Sau khi phần lớn các yêu cầu của người dùng CTS đã được ghi lại tài liệu thì trưởng dự án, Dave, và nhà phân tích yêu cầu, Lori, có cuộc gặp với hai người đại diện sản phẩm. Tim đại diện cho cộng đồng các nhà hóa học và Roxanne đại diện nhân viên kho hóa chất.

“Như anh đã biết”, Dave bắt đầu, “những người đại diện sản phẩm đã tập hợp được nhiều yêu cầu cho CTS. Nhưng không thể đưa tất cả các yêu cầu vào ngay phiên bản đầu tiên. Do phần lớn các yêu cầu là từ các nhà hóa học và kho hóa chất, nên tôi muốn nói về việc xếp thứ tự ưu tiên các yêu cầu của anh.”

Tim bối rối, “Tại sao anh muốn xếp thứ tự ưu tiên các yêu cầu? Tất cả chúng đều quan trọng, hoặc chúng tôi sẽ không trao chúng cho anh.”

Dave giải thích, “Tôi biết tất cả chúng đều quan trọng, nhưng tôi không thể thực hiện tất cả mà vẫn có thể chuyển giao sản phẩm có chất lượng đúng thời hạn được. Không có nhiều nguồn lực sẵn sàng sử dụng, vì vậy chúng tôi muốn chắc chắn chỉ làm những yêu cầu quan trọng nhất trong phiên bản đầu tiên sẽ phát hành vào cuối quý tiếp theo. Chúng tôi đề nghị anh giúp chúng tôi phân rõ yêu cầu nào là cần làm ngay và yêu cầu nào có thể để sau.”

“Tôi biết các báo cáo sử dụng và huỷ bỏ hóa chất mà Văn phòng Sức khỏe và An toàn phải thực hiện gửi chính phủ cần được làm vào cuối mỗi quý,” Roxanne nói, “Chúng tôi có thể sử dụng hệ thống lưu trữ hiện tại của kho hóa chất trong một vài tháng nếu cần thiết. Nhưng các tính năng quét và dán mã vạch thì rất cần, chúng quan trọng hơn là các vendor catalogs có thể tìm kiếm cho các nhà hóa học.”

Tim bật ra lời phản kháng. “Tôi cam đoan chức năng tìm kiếm catalog trực tuyến cho các nhà hóa học là một cách để hệ thống tiết kiệm rất nhiều thời gian. Chức năng này cần phải được đưa vào phiên bản đầu tiên,” anh nhấn mạnh.

Lori, nhà phân tích, nói, “Trong khi tôi đang duyệt các use cases với các nhà hóa học, thì dường như có một số hóa chất được đặt mua rất thường xuyên, một số khác thì ít hơn và chỉ được một số ít người mua.”

Tim và Roxanne cho rằng nếu nhóm phát triển có thể thực hiện từng nhóm yêu cầu một và chuyển giao dần thì sẽ tốt hơn nếu mọi người có thể thoả thuận về tập hợp các yêu cầu được thực thi trong phiên bản đầu tiên.

Mỗi dự án phần mềm với các giới hạn nguồn lực cần phải hiểu về các ưu tiên tương đối của các tính năng được đề nghị, các use cases, các yêu cầu chức năng. Xếp thứ tự ưu tiên các yêu cầu giúp nhà quản lý dự án phân giải các xung đột, lập kế hoạch chuyển giao từng phần, thực hiện các đánh đổi cần thiết. Chương này thảo luận về việc xếp thứ tự ưu tiên các yêu cầu và đề xuất một sơ đồ xếp thứ tự ưu tiên dựa trên giá trị, chi phí, rủi ro.

I. TẠI SAO PHẢI XẾP THỨ TỰ ƯU TIÊN YÊU CẦU?

Khi các kỳ vọng của khách hàng cao, thời gian thực hiện ngắn, nguồn lực hạn chế thì bạn cần chắc chắn sản phẩm sẽ chuyển giao cho khách hàng những chức năng quan trọng nhất càng sớm, càng tốt. Xác lập tầm quan trọng tương đối của mỗi chức năng cho phép bạn lập kế hoạch thi công để cung cấp cho khách hàng giá trị lớn nhất với chi phí thấp nhất. Xếp thứ tự ưu tiên đặc biệt quan trọng nếu bạn phải thi công hệ thống trong một khung thời gian đã bị ấn định hoặc phát triển tăng dần theo một lịch biểu chặt chẽ, không thể xê xích và vì vậy, bạn cần tạm loại bỏ hoặc trì hoãn các chức năng ít quan trọng hơn.

Một nhà quản lý dự án cần cân bằng phạm vi mong muốn của dự án với các ràng buộc về lịch biểu, ngân sách, nguồn lực con người, các mục tiêu chất lượng. Một cách để đạt được điều đó là trì hoãn thực hiện ngay một số chức năng, để chúng vào những phiên bản sau, một số yêu cầu sẽ có độ ưu tiên thấp khi những yêu cầu mới có độ ưu tiên cao hơn được chấp nhận hoặc các điều kiện của dự án thay đổi. Nếu khách hàng không phân loại các yêu cầu của họ theo tầm quan trọng và mức độ khẩn cấp của nó thì nhà quản lý dự án phải làm điều này sau đó trao đổi với khách hàng để quyết định những yêu cầu nào làm trước, yêu cầu nào làm sau.

Thiết lập mức ưu tiên sớm trong dự án, khi bạn vẫn có nhiều lựa chọn sẵn sàng để đạt được một kết thúc thành công của dự án.

Thật sự là một thách thức khi xếp yêu cầu của một khách hàng nào đó thành yêu cầu quan trọng nhất do khó có được sự đồng thuận giữa rất nhiều khách hàng với những kỳ vọng trái ngược nhau. Con người có xu hướng tự nhiên là nhìn những mối quan tâm của mình dưới khía cạnh cảm xúc, ai cũng như vậy và vì thế rất khó dàn xếp được các yêu cầu của những nhóm lợi ích khác nhau. Tuy nhiên, xếp thứ tự ưu tiên các yêu cầu là một trong những trách nhiệm của khách hàng trong quan hệ khách hàng – nhà phát triển như đã thảo luận trong Chương 2.

Cả khách hàng và nhà phát triển cùng phải thực hiện xếp thứ tự ưu tiên. Các khách hàng đặt thứ tự ưu tiên cao cho các chức năng cung cấp lợi ích lớn nhất cho người dùng. Tuy nhiên, trước khi nhà phát triển chỉ ra chi phí, mức độ khó, rủi ro kỹ thuật, hoặc các đánh đổi khác (other trade-offs) với một yêu cầu cụ thể thì khách hàng có thể xác định lại yêu cầu đó không quan trọng như họ nghĩ. Nhà phát triển có thể cũng xác định một số chức năng ưu tiên thấp nào đó cần phải được thực thi sớm do ảnh hưởng của nó tới kiến trúc của hệ thống. Xếp thứ tự ưu tiên nghĩa là cân bằng các lợi ích nghiệp vụ (business benefits) của mỗi yêu cầu dựa trên chi phí và ảnh hưởng của nó tới kiến trúc và sự tiến hóa của sản phẩm.

II. CÁC CHỦ YÊU TIÊN KHI THỰC HIỆN **(GAMES PEOPLE PLAY WITH PRIORITIES)**

Phản xạ tức thời của khách hàng trước đề xuất xếp thứ tự ưu tiên là: “Tôi cần tất cả các tính năng đó. Hãy thực hiện chúng.” Rất khó để thuyết phục khách hàng cùng thiết lập thứ tự ưu tiên nếu họ biết các yêu cầu mức ưu tiên thấp có thể sẽ không bao giờ được thực hiện. Một nhà phát triển nói với tôi trước đây rằng các mức ưu tiên là không cần thiết, lý do là anh đã viết những gì trong SRS thì tức là anh phải thực hiện nó. Tuy nhiên, điều đó không có ý nói *khi nào* thì một số chức năng nào đó được thực hiện. Nhà phát triển có thể không thích xếp thứ tự ưu tiên, họ nghĩ rằng nếu làm điều đó thì sẽ mâu thuẫn với ý “chúng tôi có thể làm mọi thứ” mà họ thường bày tỏ với khách hàng và nhà quản lý của họ.

Trong thực tế, một số tính năng quan trọng hơn tính năng khác. Điều này là hoàn toàn thực tế khi dự án đã gần hết hạn theo lịch biểu và bạn phải chuyển giao cho khách hàng một cái gì đó. Thiết lập thứ tự ưu tiên ngay từ sớm trong dự án giúp bạn chủ động ra các quyết định về đánh đổi trên toàn bộ dự án, thay vì thực hiện các hành động chừa cháy vào cuối dự án. Thực hiện được một nửa rồi bạn mới xác định nó có mức ưu tiên thấp thì quả là rủi ro.

Nếu bỏ sang một bên các mưu chước thì thường khách hàng sẽ thiết lập mức ưu tiên cao cho 85% yêu cầu, trung bình cho 10% và thấp cho 5% còn lại. Cách thiết lập như vậy không mang lại cho nhà quản lý dự án nhiều sự linh hoạt. Huỷ bỏ các yêu cầu bằng cách loại bỏ bất cứ yêu cầu nào không cơ bản và làm đơn giản hóa các yêu cầu không cần thiết phải phức tạp đã được coi như là một best practice của phát triển phần mềm nhanh (McConnell 1996). Tuy nhiên, bạn cần lưu ý điểm này, khi huỷ bỏ các yêu cầu thì vẫn phải đảm bảo kế hoạch kinh doanh của doanh nghiệp bởi việc huỷ bỏ có thể dẫn tới giảm kích thước dự án và giảm doanh thu từ dự án.

III. THANG BẬC ƯU TIÊN (PRIORITY SCALES)

Một cách tiếp cận phổ biến để xếp thứ tự ưu tiên là nhóm các yêu cầu thành 3 loại. Bảng 13-1 trình bày một số cách thức phân loại yêu cầu. Chúng hoàn toàn chủ quan và không chính xác, vì vậy những ai liên quan đến việc phân loại cũng phải thỏa thuận về ý nghĩa mỗi loại trên thang phân loại đang được sử dụng. Nếu các khái niệm tương đối như *cao*, *trung bình*, *thấp* (*high*, *medium*, *low*) gây sự phân vân cho nhiều người thì bạn có thể sử dụng khái niệm như *thực hiện ngay*, *cho phép về mặt thời gian*, *để phiên bản sau* (*committed*, *time permitting*, *future release*).

BẢNG 13-1. MỘT SỐ THANG BẬC XẾP THỨ TỰ ƯU TIÊN YÊU CẦU		
TÊN	Ý NGHĨA	THAM CHIẾU
Cao (High)	Một yêu cầu đáp ứng một mục tiêu then chốt (mission – critical); đòi hỏi phải có để thực hiện phiên bản tiếp theo.	

Trung bình (Medium)	Hỗ trợ các hoạt động hệ thống cần thiết; đòi hỏi cuối cùng cũng phải có nhưng có thể đợi cho đến phiên bản sau nếu cần thiết.	
Thấp (Low)	Một nâng cấp về mặt chức năng hoặc chất lượng; có thể là tốt nếu có vào một lúc nào đó nếu nguồn lực cho phép.	
Nền tảng (Essential)	Phần mềm là không thể được chấp nhận trừ khi yêu cầu này được thực hiện theo một cách thức thỏa thuận nào đó.	IEEE 1998
Có điều kiện (Conditional)	Có thể nâng cao chất lượng sản phẩm, nhưng sản phẩm vẫn có thể được chấp nhận nếu không có.	
Lựa chọn (Optional)	Một lớp các chức năng có thể có hoặc không thể có cũng được.	
3	Cần phải được thi công hoàn hảo	Kovitz
2	Các nhu cầu cần phải làm, nhưng không nhất thiết phải làm tốt	1999
1	Có thể chứa bugs.	

Mức ưu tiên của mỗi yêu cầu cần phải được đưa vào SRS hoặc các mô tả use case. Thiết lập một quy ước cho SRS của bạn sao cho người đọc biết liệu mức ưu tiên gán cho yêu cầu mức cao (high-level requirement) có được dẫn xuất cho các yêu cầu cấp thấp tương ứng hay không, hoặc liệu mỗi yêu cầu riêng biệt có cần thuộc tính ưu tiên của riêng nó hay không.

Thậm chí một dự án kích cỡ trung bình có thể có hàng trăm hoặc hàng nghìn yêu cầu chức năng, quá nhiều để phân loại các yêu cầu sao cho đúng đắn và nhất quán. Để có thể quản lý được yêu cầu, bạn cần chọn một mức trừu tượng thích hợp để xếp thứ tự ưu tiên – đó là các use cases, các tính năng hoặc các yêu cầu chức năng chi tiết. Xét riêng trong một use case (single use case), một số tiến trình thay thế (alternative courses) có thể có mức ưu tiên cao hơn các tiến trình khác. Bạn có thể thực hiện xếp thứ tự ưu tiên ban đầu ở mức tính năng và sau đó xếp thứ tự ưu tiên theo mức yêu cầu chức năng trong từng tính năng. Cách làm này cho phép bạn phân biệt được chức năng lõi với việc làm mịn có thể được đưa vào ở các phiên

bản sau. Ghi chép thành tài liệu cả các yêu cầu có mức ưu tiên thấp do chúng có thể thay đổi sau này và có thể thay đổi mức ưu tiên trong tương lai.

IV. XẾP THỨ TỰ ƯU TIÊN DỰA TRÊN GIÁ TRỊ, CHI PHÍ, RỦI RO

Trong các dự án nhỏ các stakeholders có thể thỏa thuận về các ưu tiên của yêu cầu bằng một cách không chính thức. Trong các dự án lớn thì người ta đòi hỏi một cách tiếp cận vấn đề bài bản hơn, họ xóa bỏ tất cả những yếu tố cảm xúc, chính trị khỏi quy trình làm việc. Một số kỹ thuật phân tích và kỹ thuật toán học đã được đề xuất để hỗ trợ xếp thứ tự ưu tiên yêu cầu. Các phương pháp này bao gồm việc đánh giá giá trị tương đối, chi phí tương đối của mỗi yêu cầu. Các yêu cầu có mức ưu tiên cao nhất là các yêu cầu mang lại giá trị lớn nhất trên tổng chi phí thấp nhất (Karlsson and Ryan 1997); Jung 1998). Với số lượng yêu cầu lớn mà thực hiện ước lượng giá trị và chi phí một cách chủ quan thì không thực tế chút nào.

Một lựa chọn khác là sử dụng phương pháp Quality Function Deployment (QFD), phương pháp này đánh giá sự liên quan tương đối giữa giá trị khách hàng (customer value) và các tính năng của sản phẩm tương lai (Cohen 1995). Cách tiếp cận thứ ba, được vay mượn từ TQM, xếp loại mỗi yêu cầu dựa trên một số tiêu chuẩn thành công của dự án được đánh trọng số và tính toán một điểm (score) để xếp hạng (rank) mức ưu tiên của các yêu cầu. Tuy nhiên, ít doanh nghiệp phần mềm thực hiện 2 cách này do sự tính toán khó khăn của chúng.

Bảng 13-2 (Xem cuối chương) minh họa một mô hình bảng tính đơn giản giúp bạn ước lượng mức ưu tiên tương đối của một tập các use cases, các tính năng của sản phẩm, hoặc từng yêu cầu chức năng riêng. Ví dụ này lấy các tính năng từ CTS. Mô hình này vay mượn từ QFD khái niệm giá trị khách hàng (customer value) phụ thuộc vào cả lợi ích (benefit) được cung cấp cho khách hàng nếu một tính năng cụ thể được cài đặt và sự trừng phạt (penalty) nếu tính năng đó không được cài đặt (Pardee 1996). Sự thu hút của một tính năng tương xứng với giá trị mà nó cung cấp và ngược lại, sức đẩy của nó tương xứng với chi phí và rủi ro kỹ thuật khi thi công tính năng đó. Khi tất cả các yếu tố cân bằng nhau, các tính năng có tỷ lệ *giá trị/chi phí* được điều chỉnh rủi ro cao nhất (*highest risk – adjusted value/cost*) thì sẽ có mức ưu tiên cao nhất. Cách tiếp cận này phân bố một tập các mức ưu tiên được

ước lượng trên một continuum (thể liên tục), thay vì nhóm chúng lại thành một số ít mức ưu tiên.

Bạn ứng dụng sơ đồ xếp thứ tự ưu tiên chỉ cho các tính năng đang đàm phán, các tính năng không ở mức ưu tiên cao nhất. Ví dụ, bạn không thể đưa vào phân tích ưu tiên các mục (items) hình thành chức năng lõi của sản phẩm, bạn xem chúng như là yếu tố định vị sản phẩm lõi, hoặc bạn cũng không đưa vào đây những yêu cầu được thực hiện do tuân thủ các quy định của chính phủ.

Những người tham gia quy trình xếp thứ tự ưu tiên gồm:

- Nhà quản lý dự án, người lãnh đạo thực hiện quy trình, phân giải các xung đột, điều chỉnh ý kiến tham gia của các thành viên khác nếu cần thiết.
- Những người đại diện khách hàng chính, ví dụ người đại diện sản phẩm, là những người đưa ra xếp hạng lợi ích và trùng phạt.
- Đại diện những người phát triển, ví dụ trưởng nhóm kỹ thuật, người cung cấp thông tin về xếp hạng chi phí và rủi ro.

Thực hiện các bước sau khi sử dụng quy trình xếp thứ tự ưu tiên này:

1. Liệt kê trên bảng tính (spreadsheet) tất cả các yêu cầu, các tính năng, hoặc các use cases mà bạn muốn xếp thứ tự ưu tiên; trong ví dụ này, bạn sử dụng các tính năng. Tất cả các items đưa vào đều phải ở mức trừu tượng hóa như nhau; tránh tình trạng item này là yêu cầu, item kia lại là tính năng. Nếu một số tính năng có liên kết với nhau một cách logic – ví dụ, bạn chỉ thi công được tính năng B nếu và chỉ nếu tính năng A đã được thi công – thì chỉ liệt kê tính năng điều khiển (driving feature), tức tính năng A trong phân tích mà thôi. Mô hình này sẽ làm việc tốt với vài chục tính năng trước khi nó trở nên khó sử dụng do có quá nhiều tính năng. Nếu bạn có quá nhiều tính năng, hãy nhóm các tính năng có liên quan lẫn nhau để tạo thành một danh sách khởi đầu có thể quản lý. Bạn có thể thực hiện lượt phân tích thứ hai ở mức chi tiết hơn nếu bạn thấy cần thiết.
2. Ước lượng lợi ích tương đối mà mỗi tính năng mang lại cho khách hàng hoặc cho hoạt động nghiệp vụ trên thang từ 1 đến 9, trong đó 1 là lợi ích không đáng kể, 9 là lợi ích vô cùng lớn. Xếp hạng các lợi ích này sóng đôi

với các yêu cầu kinh doanh (business requirements) của sản phẩm. Các đại diện khách hàng của bạn là người tốt nhất để phân xử các lợi ích này.

3. Ước lượng các trùng phạt tương đối mà khách hàng hoặc nhu cầu kinh doanh bị vi phạm nếu tính năng này không được đưa vào. Cũng sử dụng một thang đánh giá từ 1 đến 9, 1 là không trùng phạt và 9 là trùng phạt nặng. Thất bại khi tuân thủ một chuẩn công nghiệp có thể bị trùng phạt nặng thậm chí ngay cả khi khách hàng có lợi ích thấp từ việc tuân thủ đó. Các yêu cầu có lợi ích thấp và trùng phạt thấp thì chỉ làm tăng chi phí nhưng giá trị đạt được thấp; chúng có thể là thể hiện của việc mạ vàng.
4. Cột Tổng giá trị (Total Value) là tổng lợi ích tương đối và trùng phạt tương đối. Một cách ngầm định, lợi ích và trùng phạt có trọng số như nhau. Như một cách làm mịn, bạn có thể thay đổi trọng số tương đối 2 yếu tố này. Trong Bảng 13-2, tất cả các xếp hạng lợi ích có trọng số gấp đôi xếp hạng trùng phạt. Cột Giá trị % là ảnh hưởng riêng tinh băng % của mỗi tính năng trên tổng giá trị của tất cả các tính năng.
5. Ước lượng chi phí tương đối để thực thi mỗi tính năng, căn cứ trên thang từ 1 (thấp) đến 9 (cao). Bảng tính sẽ tính toán lượng phần trăm của chi phí thực hiện tính năng đó trên tổng chi phí. Các nhà phát triển có thể ước lượng các xếp hạng chi phí dựa trên độ phức tạp của yêu cầu, sự mở rộng giao diện người dùng do công việc đòi hỏi, khả năng tiềm tàng để sử dụng lại mã nguồn, lượng công việc kiểm thử và tài liệu hóa...
6. Tương tự, các nhà phát triển ước lượng mức tương đối về rủi ro kỹ thuật hoặc rủi ro khác để thực hiện mỗi tính năng theo thang từ 1 đến 9. Xếp hạng 1 có nghĩa bạn có thể lập trình tính năng này trong khi ngủ gật, 9 chỉ các vấn đề nghiêm trọng liên quan đến sự khả thi (feasibility), việc thiếu nhân viên với các kỹ năng cần thiết, hoặc do phải sử dụng các tools và công nghệ không quen thuộc hoặc chưa được chứng minh hiệu quả. Bảng tính sẽ tính toán lượng phần trăm rủi ro của mỗi tính năng trên tổng rủi ro của tất cả các tính năng. Một cách ngầm định, lợi ích, trùng phạt, chi phí và rủi ro được đánh trọng số tương đương nhau, nhưng bạn cũng có thể điều chỉnh trọng số của chúng. Trong Bảng 13-2, yếu tố rủi ro có trọng số bằng một nửa của chi phí, nó có cùng trọng số như yếu tố trùng phạt. Nếu bạn hoàn toàn không muốn cân nhắc đến rủi ro trong mô hình thì có thể thiết lập trọng số của rủi ro thành zero.

7. Sau khi bạn nhập tất cả các ước lượng vào bảng tính, thì giá trị ưu tiên của mỗi tính năng được tính theo công thức sau:

$$\text{Ưu tiên} = (\text{Giá trị \%}) / (\text{Chi phí \%} * \text{Trọng số chi phí} + \text{Rủi ro \%} * \text{Trọng số rủi ro})$$

8. Sắp xếp danh sách tính năng theo thứ tự giảm dần của mức ưu tiên đã được tính toán. Tính năng ở đỉnh của danh sách có sự cân bằng tốt nhất của giá trị, chi phí, rủi ro và vì vậy cần có mức ưu tiên cao nhất.

Phương pháp bán định lượng này không quá nặng về toán học, sự chính xác của nó bị giới hạn bởi khả năng ước lượng của bạn về lợi ích, trùng phạt, chi phí và rủi ro cho mỗi item. Vì vậy, chỉ sử dụng phương pháp này như một gợi ý. Khách hàng và các đại diện phát triển cần soát xét bảng tính đã được tính để đạt được sự đồng thuận về các xếp hạng và kết quả sắp xếp thứ tự ưu.

Mô hình này có thể giúp bạn ra các quyết định đánh đổi khi bạn đánh giá các yêu cầu được đề xuất. Ước lượng các ưu tiên để chỉ ra làm thế nào chúng sống đôi nhau. Yêu cầu hiện có sao cho bạn có thể lựa chọn được một kế hoạch thực hiện khả thi. Mỗi khi bạn cố gắng xếp thứ tự ưu tiên các yêu cầu theo kiểu từ các hành động chính trị (dàn xếp và nhân nhượng lẫn nhau) thành các hành động phân tích có mục tiêu, có kỹ thuật sẽ giúp bạn cải thiện khả năng chuyển giao các chức năng quan trọng nhất cho khách hàng trong những điều kiện khả thi nhất.

Các bước tiếp theo

- Ứng dụng mô hình xếp thứ tự ưu tiên ở đây cho 10 hoặc 15 tính năng hoặc use cases trong dự án hiện tại của bạn. Hãy so sánh xem mức độ cải thiện hoạt động thực tế khi ước lượng mức độ ưu tiên theo cách này với cách mà bạn đang làm hiện tại? Hãy so sánh xem cách này tốt như thế nào so với sự ước lượng bằng trực giác?
- Nếu không có sự liên hệ nào giữa những gì mà mô hình dự đoán với những gì bạn nghĩ là đúng, hãy phân tích xem phần nào của mô hình không cho kết quả hợp lý. Hãy cố gắng định trọng số khác nhau cho các yếu tố lợi ích, trùng phạt, chi phí và rủi ro. Điều chỉnh mô hình cho đến khi nó cung cấp kết quả nhất quán với những gì bạn kỳ vọng.

- Sau khi bạn đã thích ứng và điều chỉnh mô hình này, hãy ứng dụng nó cho dự án mới của bạn. Tích hợp các ưu tiên đã tính toán vào quy trình ra quyết định và xem liệu dự án có đạt được các kết quả làm hài lòng các stakeholders hơn so với cách tiếp cận xếp thứ tự ưu tiên trước đây.

SATA-APTECH

CHƯƠNG 14

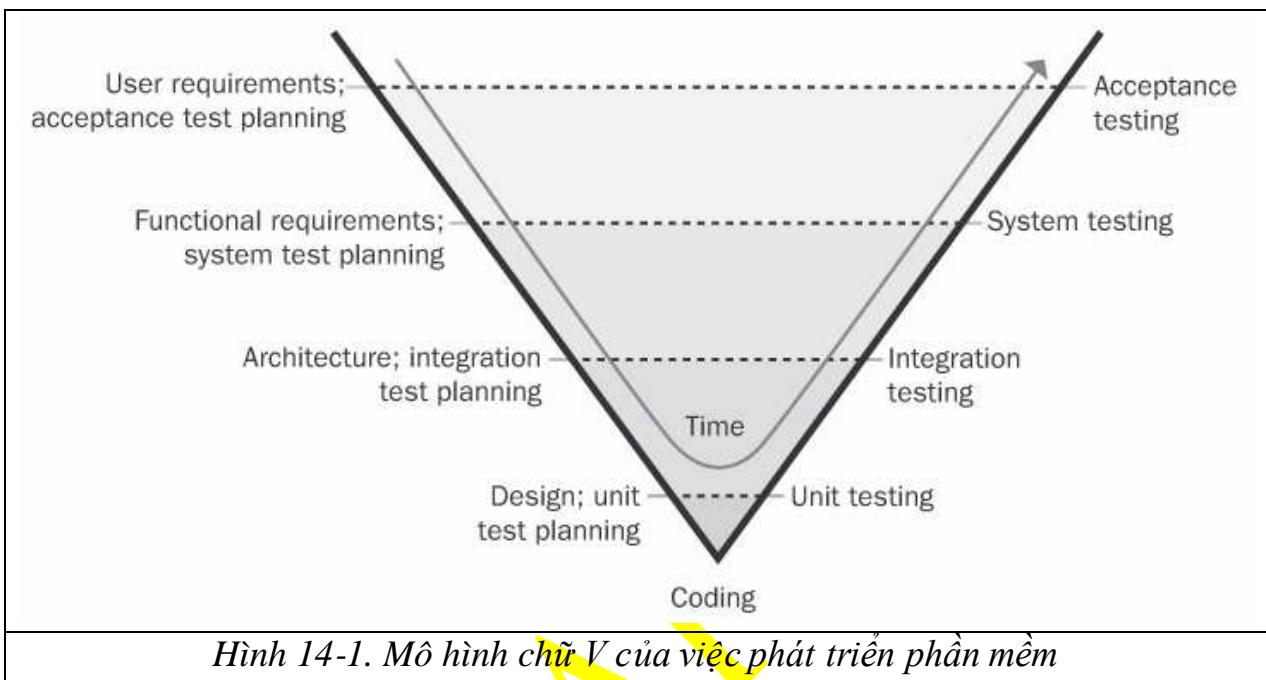
KIỂM TRA CHẤT LƯỢNG YÊU CẦU

Phần lớn các nhà phát triển đều có kinh nghiệm “đau thương” về việc phát hiện muộn các vấn đề thuộc yêu cầu trong quy trình phát triển phần mềm sau khi đã giao sản phẩm cho khách hàng. Cần rất nhiều công sức để sửa chữa các lỗi yêu cầu được phát hiện sau khi các công việc dựa trên các yêu cầu đó đã được hoàn thành. Các nghiên cứu đã chỉ ra rằng chi phí để sửa chữa sau gấp 68 đến 110 lần chi phí sửa chữa một lỗi yêu cầu được tìm ra trước trong quá trình phát triển yêu cầu. Một nghiên cứu khác cho biết rằng một lỗi được thấy khi phát triển yêu cầu sẽ lấy mất của chúng ta trung bình 30 phút để sửa chữa, trong khi một lỗi được tìm ra khi kiểm thử hệ thống sẽ lấy từ 5 đến 17 giờ để sửa chữa (Kelly, Sherif, and Hops 1992). Bất cứ việc làm theo tiêu chuẩn (measures) nào bạn có thể thực hiện để phát hiện các lỗi trong khi đặc tả yêu cầu cũng sẽ tiết kiệm rất nhiều thời gian và tiền bạc cho bạn sau này.

Trong nhiều dự án, bao gồm cả các dự án tuân theo chu trình thác nước cổ điển, kiểm thử luôn là hoạt động diễn ra cuối cùng. Các vấn đề liên quan đến yêu cầu sẽ tồn tại ngầm trong sản phẩm cho đến khi chúng lộ minh qua việc kiểm thử bởi chính các nhà phát triển hoặc bởi khách hàng, sau khi đã mất rất nhiều thời gian và tiền bạc. Nếu bạn bắt đầu lập kế hoạch kiểm thử và xây dựng các test cases ngay từ sớm trong quy trình phát triển thì bạn sẽ phát hiện được nhiều lỗi một cách nhanh chóng, như thế bạn sẽ tiết kiệm được tiền bạc và thời gian trong khi kiểm thử và bảo trì sản phẩm sau này.

Hình 14-1 thể hiện mô hình chữ V trong phát triển phần mềm, chúng thể hiện các hoạt động kiểm thử bắt đầu cùng với các hoạt động phát triển tương ứng (Davis 1993). Theo mô hình này, kiểm thử chấp nhận (acceptance testing) dựa trên yêu cầu người dùng, kiểm thử hệ thống dựa trên yêu cầu chức năng, kiểm thử tích hợp thì dựa trên kiến trúc hệ thống. Bạn phải bắt đầu lập kế hoạch cho hoạt động kiểm thử và phát triển các test cases sơ bộ tương ứng trong mỗi giai đoạn phát triển. Bạn không thể chạy bất cứ tests nào khi đang phát triển yêu cầu, do bạn không có một phần mềm nào cả. Tuy nhiên, bạn có thể tạo ra các test cases ý tưởng (conceptual

test cases) căn cứ trên các yêu cầu và sử dụng chúng để tìm kiếm các lỗi, sự nhập nhằng, các thiếu sót trong SRS và mô hình phân tích từ rất lâu trước khi nhóm phát triển bắt tay vào viết mã nguồn.



Kiểm tra yêu cầu (requirement verification)¹ là thành phần thứ tư của quy trình phát triển yêu cầu (3 thành phần trước là: suy luận, phân tích, đặc tả). Kiểm tra yêu cầu bao gồm các hoạt động nhằm đảm bảo:

- SRS mô tả đúng các hành vi và đặc tính của hệ thống mong muốn.
- Yêu cầu phần mềm được dẫn xuất một cách đúng đắn từ yêu cầu hệ thống và các nguồn khác.
- Yêu cầu đầy đủ và có chất lượng cao.
- Tất cả các quan điểm khác nhau về yêu cầu (views of requirements) đều nhất quán.
- Yêu cầu tạo ra cơ sở đầy đủ để thực hiện thiết kế, xây dựng và kiểm thử sản phẩm

¹Sự phân biệt giữa khái niệm verification và validation

Một số tác giả sử dụng khái niệm validation để gọi bước này trong quy trình công nghệ yêu cầu. Verification xác định liệu thành phẩm của một quá trình hoạt động

có đáp ứng các yêu cầu đã được đặt ra khi bắt đầu quá trình đó hay không (làm ra đúng cái cần làm). Validation đánh giá liệu một bản thành phẩm hoặc một thành phẩm cuối cùng có đáp ứng các yêu cầu đã định ở mức cao nhất hay không (làm ra cái đúng). Đối với yêu cầu phần mềm, sự phân biệt giữa 2 khái niệm này khá tinh tế và vẫn có những tranh cãi, vì vậy tôi đã sử dụng khái niệm verification theo định nghĩa của IEEE.

Kiểm tra yêu cầu đảm bảo yêu cầu tuân thủ các đặc tính của lời ~~thể~~ hiện yêu cầu tuyệt vời (excellent requirement statements) (đầy đủ, đúng đắn, khả thi, cần thiết, được xếp thứ tự ưu tiên, không nhập nhằng, có thể kiểm tra – complete, correct, feasible, necessary, prioritized, unambiguous, verifiable) và của đặc tả yêu cầu tuyệt vời (excellent requirements specifications) (đầy đủ, nhất quán, có thể chỉnh sửa, có thể làn vết – complete, consistent, modifiable, traceable). Tất nhiên, bạn có thể chỉ kiểm tra các yêu cầu đã được ghi chép thành tài liệu. Những yêu cầu chưa được mô tả, còn ẩn ý chỉ tồn tại trong tâm trí khách hàng hoặc nhà phát triển thì không thể kiểm tra.

Kiểm tra không phải là giai đoạn tách biệt một mình, bạn thực hiện giai đoạn công việc này sau khi đã thu thập và tài liệu hóa tất cả yêu cầu. Một số hoạt động kiểm tra như *soát xét tăng dần SRS tăng trưởng* (*incremental reviews of the growing SRS*), được thực hiện thông qua quy trình suy luận, phân tích, đặc tả lặp. Các bước kiểm tra khác, như *thanh tra chính thức SRS* (*formal inspection of the SRS*) là bình thường như một bộ lọc chất lượng cuối cùng nhằm vạch ranh giới SRS. Coi các hoạt động kiểm tra yêu cầu như là một nhiệm vụ tách biệt trong kế hoạch dự án hoặc cấu trúc phân việc (WBS) của bạn, hãy lập kế hoạch dành thời gian cho một số công việc tiếp theo thường xuất hiện sau mỗi hoạt động kiểm soát chất lượng.

Đôi khi ~~những~~ người tham gia dự án miễn cưỡng dành thời gian để soát xét và kiểm thử một SRS. Mặc dù có vẻ như thêm thời gian vào lịch biểu để cải thiện chất lượng yêu cầu sẽ làm trễ ngày giao hàng đã định đúng bằng lượng thời gian đó, kỳ vọng này giả định lợi ích bằng zero khi bạn đầu tư thời gian vào việc kiểm tra yêu cầu. Trên thực tế, thời gian đầu tư đó có thể làm rút ngắn thời gian giao hàng do việc giảm bớt những thứ cần làm lại và thúc đẩy nhanh hơn việc kiểm thử hệ thống. Báo cáo Capers Jones cho rằng mỗi 1 đồng chi ra để phòng ngừa các khiếm *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

khuyết sẽ làm giảm chi phí sửa chữa từ 3 đến 10 đồng (Jones 1994). Các yêu cầu tốt hơn cũng dẫn tới chất lượng sản phẩm cao hơn và đáp ứng khách hàng tốt hơn, làm giảm chi phí trên vòng đời của sản phẩm như bảo trì, nâng cấp, hỗ trợ khách hàng. Đầu tư vào chất lượng yêu cầu cũng luôn tiết kiệm cho bạn nhiều tiền hơn là số tiền bạn phải chi ra.

Các kỹ thuật khác nhau có thể giúp bạn kiểm tra tính đúng đắn và chất lượng của yêu cầu (Wallace and Ippolito 1997). Chương này tập trung vào 2 trong số nhiều kỹ thuật kiểm tra quan trọng nhất: soát xét yêu cầu chính thức (formal) và phi chính thức, các test cases ý tưởng (conceptual) được phát triển từ use cases và yêu cầu chức năng.

I. SOÁT XÉT YÊU CẦU (REVIEWING REQUIREMENTS)

Mỗi lần ai đó không phải tác giả của một bán thành phẩm phần mềm (software work product) kiểm tra bán thành phẩm để tìm kiếm khuyết thì việc đó gọi là soát xét kỹ thuật (technical review). Soát xét các tài liệu yêu cầu là kỹ thuật rất hữu hiệu để xác định các yêu cầu **nhập nhằng**, hoặc không thể kiểm tra, các yêu cầu không sáng sửa đủ để làm cơ sở cho việc thiết kế, và “yêu cầu” đó trên thực tế chính là các đặc tả thiết kế (design specifications).

Soát xét yêu cầu cũng là một cách để các stakeholders thảo luận về mức độ hệ trọng của một chi tiết cụ thể nào đó. Đồng nghiệp Barry của tôi trước đây đã lãnh đạo một nhóm soát xét SRS, nhóm đó gồm các đại diện từ 4 lớp người dùng. Một người dùng đề nghị đưa một chi tiết bên ngoài vào phạm vi xem xét: chi tiết này tạo một thay đổi hệ trọng đối với các yêu cầu. Sau cuộc họp, người phân tích yêu cầu và trưởng dự án đã rất tức giận vì chi tiết đó đã không được đưa ra trong cuộc họp 2 tháng trước đó. Sau một số cuộc họp, chi tiết này lại bị đưa ra bên ngoài phạm vi SRS nhưng người dùng đó không biết. Khi một số người dùng thảo luận trong cuộc soát xét về tầm mức hệ trọng của chi tiết đó thì người phân tích yêu cầu và trưởng dự án nhận ra rằng họ đã không biết về điều đó sớm hơn.

Các loại hình soát xét kỹ thuật khác nhau có rất nhiều tên gọi khác nhau. Cách tiếp cận phi chính thức là chuyển các bán thành phẩm (work products) cho một số người đánh giá (peers) để xem xét đánh giá. Soát xét phi chính thức rất tốt để *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

truyền thông cho những người khác biết về sản phẩm và nhận được các phản hồi phi chính thức, nhưng cách làm này không hệ thống, không được thực thi một cách nhất quán. Soát xét phi chính thức không đòi hỏi các ghi chép thành văn.

Trong khi các soát xét phi chính thức được thực hiện theo một cách nào đó thuận tiện thì một kiểu soát xét chính thức tuân thủ một quy trình được định nghĩa chặt chẽ bằng một dãy các bước đã định. Một soát xét chính thức cần phải ghi lại kết quả trong một báo cáo thành văn, báo cáo đó xác định vấn đề chính của buổi soát xét, những người tham gia soát xét, cơ sở lý luận của nhóm soát xét khi nhận định sản phẩm có đầy đủ hay không, hoặc cần phải làm thêm, cuối cùng là một tổng kết về các khuyết điểm được tìm thấy. Thành viên của một nhóm soát xét chính thức chia sẻ trách nhiệm về chất lượng sau soát xét, mặc dù các tác giả có trách nhiệm chính về sản phẩm mà họ tạo ra (Freedman and Weinberg 1990).

Cách được cho là tốt nhất để soát xét kỹ thuật chính thức gọi là *thanh tra* (*inspection*) (Ebenau and Strauss 1994; Gilb and Graham 1993). Thanh tra tài liệu yêu cầu hiện là kỹ thuật kiểm tra chất lượng phần mềm luôn được ứng dụng rộng rãi nhất. Một số công ty đã cho rằng họ tiết kiệm được 10 giờ làm việc cho mỗi giờ dành cho việc thanh tra tài liệu yêu cầu và các bán thành phẩm khác (Grady 1994). Tôi cho rằng không có cách quản lý chất lượng hoặc quy trình phát triển phần mềm nào có thể sinh ra mức thu hồi trên vốn đầu tư (ROI) là 1000% như vậy.

Nếu bạn thấy rất khó khăn khi tìm cách nâng cao chất lượng phần mềm, thì bạn hãy thanh tra mỗi dòng của mỗi tài liệu yêu cầu mà bạn đã viết. Mặc dù sự thanh tra chi tiết một khối lượng tài liệu lớn là rất mệt mỏi và mất thời gian thì bạn hãy chú ý rằng mỗi giờ thực hiện việc đó sẽ tiết kiệm cho bạn gấp mười lần số thời gian sau này. Nếu bạn nghĩ bạn không có thời gian để thanh tra thì hãy thực hiện một phân tích rủi ro đơn giản để phân biệt đâu là phần tài liệu cần thanh tra, đâu là phần tài liệu chỉ cần thực hiện một soát xét phi chính thức là đủ đáp ứng các mục tiêu chất lượng.

Trong trường hợp CTS, các nhóm đại diện cho nhiều lớp người dùng khác nhau soát xét phi chính thức các ý kiến của họ đóng góp vào SRS sau mỗi cuộc họp suy luận yêu cầu, họ không phát hiện tất cả các lỗi ngay lúc đó. Sau khi suy luận đã
Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

hoàn tất, một phân tích viên kết hợp tất cả các phần khác nhau của SRS từ các lớp người dùng khác nhau thành một tài liệu duy nhất khoảng 50 trang cùng một số phụ lục. Hai phân tích viên, một người phát triển phần mềm, ba người đại diện sản phẩm, nhà quản lý dự án, một kiểm thử viên đã thanh tra SRS này trong 3 cuộc họp, mỗi cuộc họp 2 giờ. Nhóm thanh tra đã tìm ra 223 lỗi, kèm thêm vài chục khiếm khuyết lớn (major defects). Nhóm thanh tra thỏa thuận về thời gian họ cần để xử lý các vấn đề trên của SRS, thời gian được dành để làm việc đó sẽ tiết kiệm cho họ rất nhiều giờ sau này trong khi xây dựng hệ thống.

QUY TRÌNH THANH TRA (INSPECTION PROCESS)

Michael Fagan đã xây dựng quy trình thanh tra tại IBM từ giữa những năm 1970 (Fagan 1976), hiện quy trình này được tổ chức lại thành một hướng dẫn thực hành tốt nhất (best practice) trong công nghiệp phần mềm (Brown 1996). Bất cứ bản thành phẩm phần mềm (software work product) nào cũng có thể được thanh tra, đó có thể là tài liệu yêu cầu, tài liệu thiết kế, mã nguồn, tài liệu kiểm thử, kế hoạch dự án, ... Thanh tra là một quy trình nhiều bước được thiết kế chặt chẽ để một nhóm nhỏ các thành viên được đào tạo (về quy trình) tập trung tìm kiếm các khiếm khuyết của bản thành phẩm. Quy trình tạo ra một cổng chất lượng (quality gate) nơi mà các tài liệu phải vượt qua trước khi chúng được vạch ranh giới. Trong khi vẫn có những tranh luận liệu phương pháp Fagan có hiệu quả hay không đối với việc thanh tra (Glass 1999), thì không hề có chút nghi ngờ nào khi cho rằng thanh tra là một kỹ thuật quản lý chất lượng đầy sức mạnh.

1. Những người tham gia (Participants)

Những người tham gia một cuộc thanh tra cần phải là đại diện từ 3 nơi:

- *Tác giả của bản thành phẩm (work product) và có thể thêm các đồng nghiệp (peers) của tác giả.* Nhà phân tích viết tài liệu yêu cầu.
- *Tác giả của mỗi bản thành phẩm trước đó (predecessor work product) hoặc bản đặc tả của bản thành phẩm đang được thanh tra.* Điều này có nghĩa là một kỹ sư hệ thống hoặc một kiến trúc sư hệ thống có thể kiểm tra SRS về khả năng lần lượt thích hợp của mỗi yêu cầu đối với một đặc tả hệ thống. Trong trường hợp thiếu tài liệu yêu cầu mức cao hơn (higher level requirement document), thì cuộc thanh tra phải có mặt cả các khách hàng để đảm bảo SRS mô tả chính xác và đầy đủ yêu cầu của họ.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- *Phải làm việc dựa trên tài liệu được thanh tra.* Để thanh tra một SRS, bạn có thể tập hợp một nhà phát triển, một kiểm thử viên, một quản trị dự án, một người viết tài liệu người dùng, và tất cả những ai thực hiện các công việc dẫn xuất từ SRS. Các thanh tra viên sẽ nhìn SRS từ nhiều góc độ khác nhau và phát hiện các vấn đề khác nhau. Một kiểm thử viên cần phải bắt được một yêu cầu không thể kiểm tra, trong khi một nhà phát triển cần định vị được các yêu cầu không khả thi về mặt kỹ thuật.

Chỉ nên thành lập nhóm thanh tra gồm nhiều nhất 7 người. Các nhóm lớn dễ bị sa lầy vào các thảo luận bên lề, sa lầy vào việc đề ra cách giải quyết vấn đề và các tranh luận vượt quá những gì thật sự là lỗi, vì thế mà làm chậm tốc độ bao quát vấn đề, làm tăng chi phí tìm kiếm mỗi khiếm khuyết.

2. Các vai trò trong cuộc thanh tra (Inspection Roles)

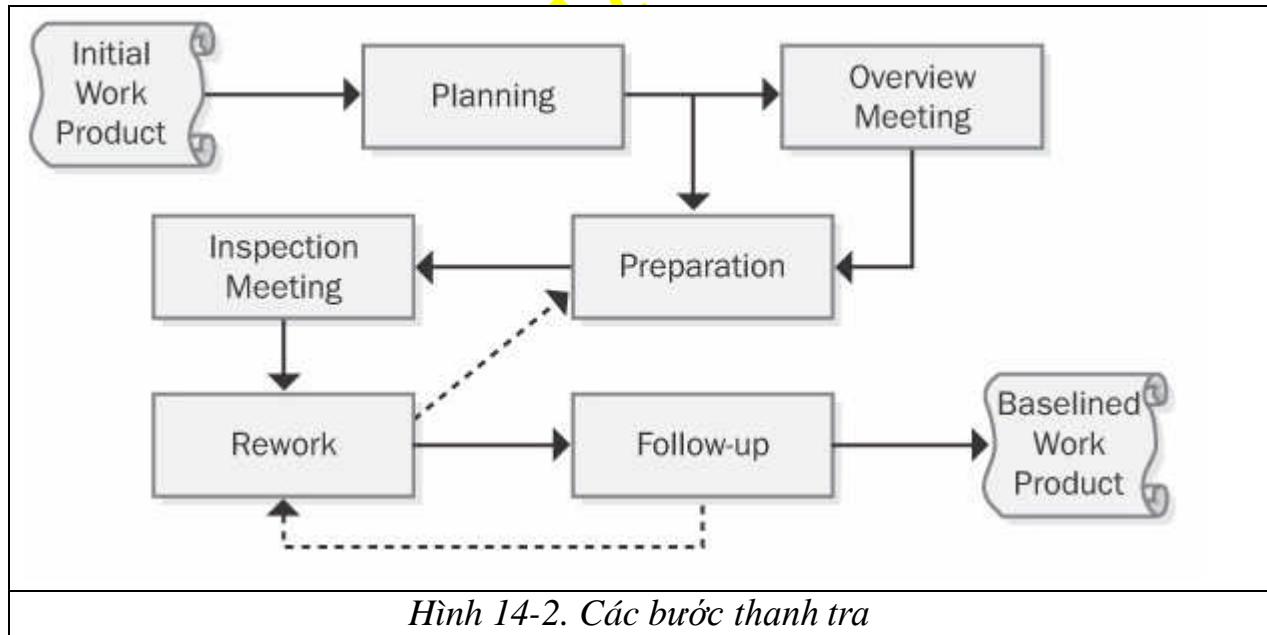
Một số thành viên nhóm thanh tra thực hiện các vai trò cụ thể trong cuộc thanh tra như sau.

- **Tác giả (Author).** Người tạo ra và duy trì sản phẩm đang được thanh tra. Tác giả của một SRS thường là phân tích viên, người thu thập yêu cầu từ khách hàng và viết đặc tả. Trong các cuộc soát xét phi chính thức như duyệt qua (walkthroughs), tác giả thường dẫn dắt cuộc thảo luận. Tuy nhiên, tác giả nên đóng vai trò bị động (passive role) trong một cuộc thanh tra và không nên đóng bất cứ vai trò nào khác như – người điều tiết, người đọc, người ghi chép. Bằng cách đóng một vai trò tích cực và đứng ngoài lề, tác giả có thể nghe được các ý kiến từ những thanh tra viên khác, trả lời – chứ không tranh luận – các câu hỏi của họ, sau đó thì suy nghĩ. Tác giả thường dễ che dấu các lỗi (the author will often spot errors) mà các thanh tra viên khác không nhìn thấy.
- **Người điều tiết (Moderator).** Người điều tiết, hoặc lãnh đạo cuộc thanh tra lập kế hoạch thanh tra cùng với tác giả, điều phối các hoạt động, thúc đẩy cuộc họp thanh tra. Người điều tiết phân bổ các tài liệu (materials) được thanh tra cho những người tham gia vài ngày trước khi cuộc họp diễn ra, bắt đầu cuộc thanh tra đúng thời hạn, khuyến khích sự đóng góp của các thành viên, hướng các cuộc họp vào mục tiêu tìm kiếm các khiếm khuyết thay vì phân giải các chi tiết không hợp lý. Báo cáo kết quả thanh tra tới cấp quản lý

hoặc cho người tổng hợp dữ liệu từ nhiều cuộc thanh tra cũng do người điều tiết chịu trách nhiệm. Một vai trò cuối mà người điều tiết phải nắm là theo dõi các thay đổi đã đề nghị với tác giả, nhằm đảm bảo chắc chắn các khiếm khuyết và các chi tiết không hợp lý tìm kiếm được sẽ được sửa chữa.

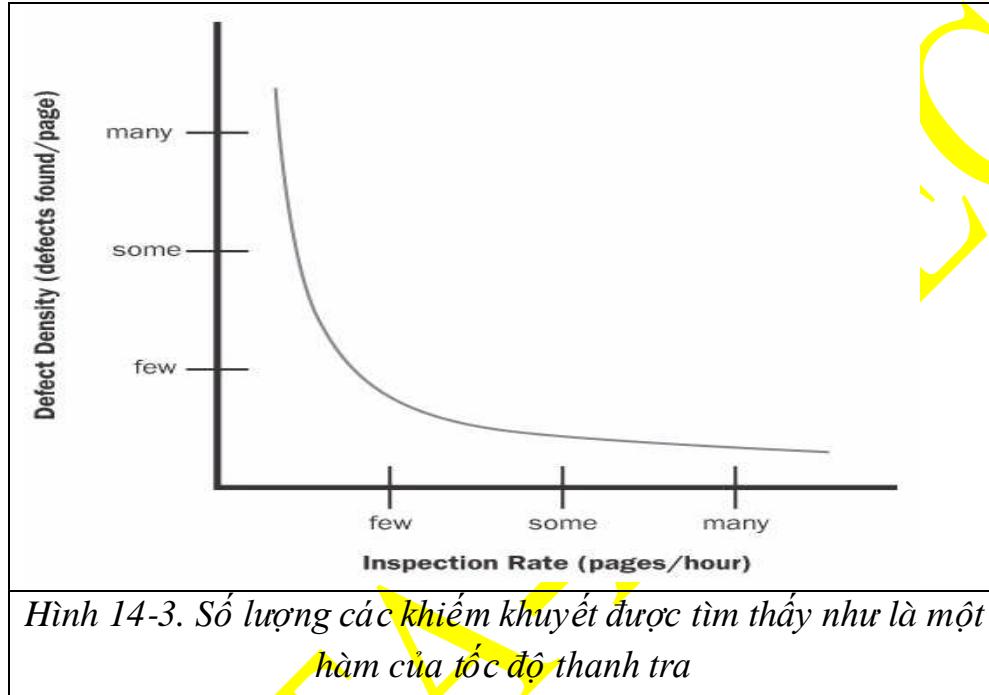
- **Người đọc (Reader).** Một thanh tra viên đóng vai trò người đọc. Trong cuộc họp thanh tra, người đọc diễn giải từng đoạn tài liệu ngắn cần thanh tra và để cho những thành viên khác phát biểu cách hiểu của họ. Đối với một đặc tả yêu cầu, người đọc cần trình bày một yêu cầu được gán nhãn hoặc từng đoạn ngắn một. Bằng cách mô tả các yêu cầu trong ngôn ngữ của riêng mình, người đọc cung cấp một sự diễn giải có thể khác với cách diễn giải của những thanh tra viên khác, đó chính là cách để lộ ra sự nhập nhằng hoặc một khiếm khuyết.
- **Người ghi chép (Recorder).** Người ghi chép ghi lại thành văn các vấn đề phát sinh và các khiếm khuyết được tìm thấy trong khi thanh tra theo một khuôn dạng văn bản chuẩn. Các thanh tra viên cần giúp người ghi chép nắm bắt được cốt lõi của mỗi vấn đề này sinh theo một cách thuyết phục để truyền thông cho tác giả về vị trí và bản chất của vấn đề.

3. Các bước thanh tra (Inspection stages)



Hình 14-2 minh họa một quy trình thanh tra (theo Ebenau and Strauss 1994). Mục đích của bước trong quy trình thanh tra được tóm tắt như sau.

Lập kế hoạch (Planning). Tác giả và người điều phối cùng lập kế hoạch thanh tra, xác định những ai tham gia, tài liệu nào mà các thanh tra viên cần có trước khi họp thanh tra, bao nhiêu cuộc họp thanh tra là cần thiết. Tốc độ thanh tra ảnh hưởng lớn đến số lượng khiếm khuyết được phát hiện (Gilb and Graham 1993).



Hình 14-3. Số lượng các khiếm khuyết được tìm thấy như là một hàm của tốc độ thanh tra

Như được mô tả trong Hình 14-3, bạn càng soát xét SRS chậm bao nhiêu thì càng để lộ nhiều khiếm khuyết bấy nhiêu. Do không ai có thể xác định trước được thời gian dành cho việc thanh tra yêu cầu, lựa chọn một tốc độ thích hợp căn cứ trên rủi ro của việc bỏ sót các khiếm khuyết chính. Từ 4 đến 6 trang trong mỗi giờ là một hướng dẫn thực tế, nhưng có thể điều chỉnh tốc độ này dựa trên các yếu tố sau:

- Lượng chữ viết (text) trên mỗi trang.
- Độ phức tạp của đặc tả.
- Mức độ quan trọng của tài liệu được thanh tra đối với thành công của dự án.
- Dữ liệu thanh tra quá khứ của bạn.
- Mức độ kinh nghiệm của tác giả SRS.

Họp tổng quan (Overview Meeting).

Họp tổng quan để thông tin cho các thanh tra viên về bối cảnh (background) của tài liệu sẽ được thanh tra, bất cứ giả định (assumptions) nào mà tác giả dựa vào, và các mục tiêu thanh tra cụ thể của tác giả. Bạn có thể bỏ qua cuộc họp này nếu tất cả các thanh tra viên đã quen thuộc với các vấn đề cần thanh tra.

Chuẩn bị (Preparing). Trong khi chuẩn bị cho cuộc thanh tra chính thức, mỗi thanh tra viên kiểm tra sản phẩm để xác định các khiếm khuyết và vấn đề có thể phát sinh, sử dụng các bảng kiểm tra (checklists) như là một hướng dẫn thực hiện để tìm kiếm các khiếm khuyết điển hình (sẽ được mô tả sau trong chương này). Trên 75% các khiếm khuyết được tìm thấy trong mỗi cuộc thanh tra được phát hiện chính trong quá trình chuẩn bị, do vậy đừng bỏ qua bước này. Các thanh tra viên không được chuẩn bị đầy đủ sẽ khiến các cuộc họp thanh tra không hiệu quả, từ đó có thể có nhận định các cuộc thanh tra là vô bổ. Hãy nhớ rằng thời gian mà bạn sử dụng để kiểm tra công việc của một đồng nghiệp là một sự đầu tư giúp tăng chất lượng sản phẩm của cả nhóm.

Họp thanh tra (Inspection meeting). Trong khi họp thanh tra, người đọc dẫn dắt nhóm thanh tra căn cứ trên SRS, đọc mỗi yêu cầu một lần. Các thanh tra viên khui ra các khiếm khuyết và các vấn đề khác, người ghi chép (recorder) ghi lại chúng trong một tài liệu với khuôn dạng đã định, tài liệu này trở thành một danh sách các hành động cần thiết (action item list) cho tác giả của yêu cầu. Mục đích của cuộc họp là xác định càng nhiều càng tốt các khiếm khuyết chính trong tài liệu yêu cầu. Rất dễ dàng cho các thanh tra viên chỉ thảo luận các vấn đề bề mặt hoặc tranh cãi liệu một vấn đề có thật sự là khiếm khuyết hay không, hoặc tranh cãi về phạm vi của dự án, hoặc tìm kiếm các giải pháp cho bài toán. Các hoạt động trên là bình thường song chúng làm giảm sự chú ý tới mục tiêu lõi là tìm kiếm các khiếm khuyết quan trọng và các cơ hội cải tiến chúng. Cuộc họp thanh tra không nên kéo dài quá 2 giờ, nếu bạn cần thêm thời gian thì hãy lập lịch các cuộc họp sau đó. Khi kết luận cuộc họp, nhóm quyết định liệu tài liệu yêu cầu có được chấp nhận như hiện tại, được chấp nhận với một số sửa đổi nhỏ, không được chấp nhận do cần phải soát xét lại, thanh tra lại.

Một số nhà nghiên cứu cho rằng các cuộc họp thanh tra là quá thiên về việc đánh giá chủ quan của các thanh tra viên, nhưng tôi lại thấy rằng các cuộc họp này để lộ ra các khiếm khuyết bỗng nhiên mà không một thanh tra viên nào khi làm việc một mình có thể phát hiện ra. Như tất cả các hoạt động chất lượng, bạn cần phải quyết định rằng bạn cần dành bao nhiêu công sức để cải tiến chất lượng yêu cầu trước khi bạn thực hiện việc thiết kế và thi công.

Làm lại (Rework). Gần như mỗi hoạt động kiểm soát chất lượng mà tôi đã tham gia đều tìm thấy khiếm khuyết của yêu cầu. Do vậy, tác giả cần lập kế hoạch tiêu tốn thêm thời gian để làm lại tài liệu theo yêu cầu từ cuộc họp thanh tra. Các yêu cầu không được chỉnh sửa lại sẽ đây chi phí nên rất cao khi dự án được thực hiện sau này. Nếu bạn không định chỉnh sửa các yêu cầu sau đó thì không có nhiều lý do để tổ chức các cuộc họp thanh tra.

Giám sát (Follow-up). Trong bước thanh tra cuối cùng này, người điều tiết hoặc một ai đó được chỉ định sẽ giám sát tác giả chỉnh sửa các khiếm khuyết. Giám sát đảm bảo tất cả các vấn đề để ngỏ đều được phân giải và các lỗi yêu cầu được sửa chữa hoàn toàn. Giám sát là bước kết thúc quy trình thanh tra và tạo điều kiện cho người điều tiết xác định liệu tiêu chuẩn đi ra (exit criteria) của thanh tra có được đáp ứng.

4. Tiêu chuẩn đi vào và tiêu chuẩn đi ra (Entry and Exit Criteria)

Bạn đang thanh tra một tài liệu yêu cầu phần mềm xem nó có đáp ứng các điều kiện tiên quyết cụ thể. *Tiêu chuẩn đi vào* (*entry criteria*) thiết lập một số kỳ vọng rõ ràng mà các tác giả phải tuân thủ khi chuẩn bị một cuộc thanh tra. Chúng giúp nhóm thanh tra khỏi mất thời gian vào các chi tiết cần phải được phân giải. Người điều tiết sử dụng tiêu chuẩn đi vào như một checklist trước khi quyết định có thực hiện cuộc thanh tra hay không. Tham khảo [tài liệu tiêu chuẩn đi vào](#) khi thanh tra tài liệu yêu cầu.

Số	Tiêu chuẩn	Đáp ứng
1	Tài liệu tuân theo template chuẩn	
2	Tài liệu đã được kiểm tra chính tả và, nếu thích hợp, thì đã kiểm tra ngữ pháp	

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

3	Tác giả đã kiểm tra tài liệu bằng cảm quan (visually) để sửa chữa các lỗi về trình bày.	
4	Tất cả các tài liệu liên quan hoặc tài liệu tham khảo mà các thanh tra viên cần để thực hiện công việc, ví dụ tài liệu đặc tả yêu cầu hệ thống, đều đã có sẵn.	
5	Số dòng (để trống) đã được in thêm vào tài liệu để thuận tiện cho việc tham chiếu tới các vị trí cụ thể khi thanh tra.	
6	Tất cả các vấn đề còn để ngỏ đều được đánh dấu TBD (cần làm rõ - to be determined)	
7	Một bảng thuật ngữ giải thích các từ chung được in kèm theo tài liệu	

Tương tự, bạn cũng nên định nghĩa một tài liệu *tiêu chuẩn đi ra* (*exit criteria*) cần phải được hoàn thành trước khi người điều tiết kết thúc cuộc thanh tra. Tham khảo [tài liệu tiêu chuẩn đi ra](#) khi thanh tra tài liệu yêu cầu.

Số	Tiêu chuẩn	Đáp ứng
1	Tất cả các chi tiết phát sinh trong khi thanh tra đều đã được chỉ rõ	
2	Mọi thay đổi của tài liệu đều được thực hiện đúng đắn	
3	Tài liệu sau khi soát xét đã được kiểm tra chính tả và, nếu thích hợp, thì đã kiểm tra ngữ pháp	
4	Tất cả các TBDs đều đã được phân giải, hoặc quy trình phân giải mỗi TBD, ngày thực hiện, và người thực hiện đều được ghi nhận vào tài liệu.	
5	Tài liệu đã được check in vào hệ thống quản lý cấu hình của dự án.	
6	Các số liệu thanh tra đã được báo cáo tới người tổng hợp số liệu.	

5. Các checklists thanh tra yêu cầu (Requirements Inspection Checklists)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Để giúp các thanh tra viên tìm kiếm các kiểu lỗi trong sản phẩm mà họ phải thanh tra, bạn hãy xây dựng một checklist cho mỗi kiểu tài liệu mà tổ chức của bạn tạo ra. Các checklist đó lôi kéo sự chú ý của các thanh tra viên vào các vấn đề về yêu cầu vẫn thường xảy ra trong quá khứ.

Hình 14-4 minh họa một checklist để thanh tra một SRS.

Số thứ tự	Tiêu chuẩn	Đáp ứng
KẾT CẤU CỦA SRS VÀ TÍNH ĐẦY ĐỦ		
1	Tất cả các tham chiếu chéo nội bộ tới các yêu cầu khác là đúng đắn?	
2	Tất cả các yêu cầu đều được mô tả ở cùng một mức chi tiết nhất quán và thích hợp?	
3	Các yêu cầu tạo cơ sở đầy đủ cho việc thiết kế?	
4	Mức ưu tiên thực thi của mỗi yêu cầu đều được xác định?	
5	Tất cả các giao diện phần cứng, phần mềm và truyền thông đều được định nghĩa?	
6	Các thuật toán để cài đặt các yêu cầu chức năng đều được định nghĩa?	
7	SRS đã mô tả tất cả các khách hàng hoặc hệ thống đã biết cần thiết của phần mềm?	
8	Một yêu cầu bị khuỷu thông tin nào đó? Nếu đúng như vậy hãy đánh dấu yêu cầu đó là TBD.	
9	Hành vi được kỳ vọng đã được mô tả đầy đủ cho tất cả các điều kiện lỗi tiên đoán?	
TÍNH ĐÚNG ĐĂN		
10	Có những yêu cầu nào xung đột với nhau hay không, hay mô tả cùng một nội dung hay không?	
11	Mỗi yêu cầu đều được viết bằng một ngôn ngữ sáng sủa, chính xác, không nhập nhằng?	
12	Mỗi yêu cầu đều có thể được kiểm tra bằng kiểm thử, chứng minh, soát xét hoặc phân tích?	
13	Mỗi yêu cầu đều thuộc phạm vi của dự án?	

14	Mỗi yêu cầu đều không có lỗi về nội dung hoặc lỗi về ngữ pháp?	
15	Có thể tất cả các yêu cầu đều được thi công trong các ràng buộc đã biết?	
16	Tất cả các messages báo lỗi cụ thể đều là duy nhất và có ý nghĩa?	
CÁC THUỘC TÍNH CHẤT LƯỢNG		
17	Tất cả các mục tiêu hiệu năng đều được đặc tả một cách hợp lý?	
18	Tất cả các cản nhắc về an ninh và an toàn đều được đặc tả một cách hợp lý?	
19	Tất cả các mục tiêu chất lượng khác đều được mô tả và định lượng sáng sủa, với các đánh đổi có thể chấp nhận và được đặc tả?	
KHẢ NĂNG LÀN VÉT		
20	Mỗi yêu cầu đều được định danh đúng đắn và duy nhất?	
21	Mỗi yêu cầu chức năng đều được lẩn vết tối một yêu cầu mức cao hơn? (Ví dụ yêu cầu hệ thống hoặc use case)	
CÁC VẤN ĐỀ ĐẶC BIỆT		
22	Tất cả các yêu cầu đang có đều không kèm theo các giải pháp thiết kế hoặc thi công?	
23	Tất cả các chức năng then chốt về mặt thời gian (time-critical functions) đều được định danh và tiêu chuẩn thời gian được mô tả rõ?	
24	Các vấn đề quốc tế hoá đều được xác định rõ?	
<i>Hình 14-4. Checklist thanh tra tài liệu SRS</i>		

Hình 14-5 minh họa một [checklist để thanh tra use cases](#).

Số	Tiêu chuẩn	Đáp ứng
1	Mỗi use case là một tác vụ độc lập và tách rời?	
2	Mục tiêu hoặc giá trị có thể đo lường của use case được xác định rõ?	

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

3	Actor tương tác với use case được xác định rõ?	
4	Use case được mô tả ở mức cơ bản (trừu tượng), thay vì như một kịch bản cụ thể?	
5	Use case không xác định các chi tiết thiết kế và thi công?	
6	Tất cả các tiến trình thay thế dự đoán (anticipated alternative courses) đều được mô tả đầy đủ?	
7	Tất cả các điều kiện loại trừ đã biết (known exception conditions) đều được mô tả đầy đủ?	
8	Có chuỗi hành động chung nào có thể được tách ra từ các use cases? (Để hình thành use case mới)	
9	Chuỗi đối thoại của mỗi tiến trình được mô tả sáng sủa, không nhập nhằng và đầy đủ?	
10	Mỗi actor và bước trong use case đều thích hợp để thực thi tác vụ?	
11	Mỗi tiến trình được định nghĩa trong use case đều khả thi?	
12	Mỗi tiến trình được định nghĩa trong use case đều có thể kiểm tra?	

Hình 14-5. Checklist thanh tra tài liệu use-case

Không ai có thể nhớ được tất cả các mục trong một checklist dài. Hãy suy nghĩ và chỉnh sửa mỗi checklist trên để phù hợp nhất với nhu cầu của tổ chức của bạn. Bạn có thể đề nghị các thanh tra viên khác nhau sử dụng các tập con khác nhau của checklist tổng thể để tìm kiếm các khiếm khuyết. Một người có thể kiểm tra xem tất cả các tham chiếu chéo trong nội bộ tài liệu có đúng hay không, người khác thì xác định liệu các yêu cầu có tạo cơ sở vững chắc cho việc thiết kế hay không, người thứ ba thì chỉ đánh giá xem liệu các yêu cầu có thể được kiểm tra hay không. Một số nghiên cứu đã chỉ ra nếu trao cho các thanh tra viên trách nhiệm phát hiện các khiếm khuyết cụ thể thì cần cung cấp cho họ các quy trình làm việc cụ thể hoặc các kịch bản cụ thể để họ săn tìm các kiểu lỗi cụ thể. Cách này hiệu quả hơn là để tất cả các thanh tra viên cùng xử lý một checklist (Porter, Votta, and Basili 1995).

CÁC THÁCH THỨC KHI SOÁT XÉT YÊU CẦU (REQUIREMENTS REVIEW CHALLENGES)

Một số thách thức khi soát xét yêu cầu được nêu ra ở đây với các khuyến nghị về cách xử lý.

1. Các tài liệu yêu cầu lớn (Large requirements documents)

Viễn cảnh phải thanh tra một SRS dày vài trăm trang thật sự tạo ra cảm giác rát mệt mỏi. Bạn có thể bị cảm dỗ bỏ qua việc thanh tra toàn thể mà chỉ thanh tra cấu trúc tài liệu, nhưng đó không phải một lựa chọn tốt. Thậm chí với một SRS có kích thước trung bình, tất cả các thanh tra viên kiểm tra kỹ càng phần đầu tiên, kiểm tra hơi kỹ phần giữa nhưng không có gì chắc chắn họ sẽ kiểm tra kỹ phần cuối. Để tránh việc nhóm thanh tra bị tràn, đừng đợi để bắt đầu soát xét SRS cho đến khi bạn sẵn sàng vạch ranh giới cho nó. Hãy thực hiện việc soát xét phi chính thức, cứ soát xét dần dần trong khi bạn vẫn đang phát triển SRS, trước khi thanh tra tài liệu đầy đủ. Yêu cầu một số thanh tra viên bắt đầu làm việc từ những vị trí khác nhau của tài liệu để chắc chắn họ sẽ chú ý vào phần được phân công. Nếu bạn có đủ các thanh tra viên, hãy chia họ thành các nhóm nhỏ để thanh tra các phần khác nhau của tài liệu.

2. Các nhóm thanh tra lớn (Large inspection teams)

Nhiều thành viên và khách hàng tham gia dự án đều ít nhiều có liên quan đến yêu cầu, vì vậy bạn có một danh sách những thành viên tiềm năng cho vị trí thanh tra yêu cầu. Tuy nhiên, các nhóm thanh tra lớn sẽ gây khó cho việc lập lịch biểu họp hành, khi họp lại hay nghiêng về các cuộc thảo luận bên lề, nhóm lớn cũng khó hơn trong việc đạt được các thỏa thuận chung. Hãy cố gắng thực hiện các tiếp cận sau để giải quyết những vấn đề tiềm tàng của một nhóm thanh tra lớn:

- Hãy chắc chắn rằng mỗi thành viên của nhóm tham gia với mục đích tìm kiếm các khuyết điểm của tài liệu chứ không phải với mục đích bảo vệ một vị trí làm việc. Nếu mong muốn am hiểu vấn đề là mục đích của một số thành viên thì hãy mời họ tham gia các cuộc họp tổng quan chứ không phải các cuộc họp thanh tra.
- Hiểu được vai trò mà mỗi thành tra viên đại diện (khách hàng, người phát triển, người kiểm thử), lịch sự từ chối sự tham gia của người khác vào vị trí đã có người đại diện.

- Thành lập một số nhóm nhỏ để thanh tra song song SRS và kết hợp danh sách các khiếm khuyết mà mỗi nhóm tìm được, xoá bỏ những trùng hợp nếu có. Các nghiên cứu đã chỉ ra nhiều nhóm thanh tra sẽ tìm được nhiều lỗi hơn trong tài liệu yêu cầu so với một nhóm lớn (Martin and Tsai 1990; Schneider, Martin and Tsai 1992; Kosman 1997). Các nhóm khác nhau có khuynh hướng nhìn thấy các khía cạnh khác nhau của các khiếm khuyết mà họ tìm thấy, do vậy mà kết quả của các cuộc thanh tra song song về cơ bản là bổ sung cho nhau hơn là dư thừa.

3. Sự xa cách về địa lý giữa các thanh tra viên (Geographic separation of inspectors)

Ngày càng nhiều các tổ chức phát triển đang cùng nhau xây dựng các sản phẩm thông qua sự hợp tác giữa các nhóm bị phân tách về địa lý. Sự phân tách này khiến các cuộc soát xét yêu cầu trở nên rủi ro. Videoconferencing có thể là một giải pháp hiệu quả, nhưng teleconferencing không cho bạn thấy được ngôn ngữ cơ thể và sự diễn giải của các thành viên tham gia soát xét như các cuộc họp mặt đối mặt. Tất cả các cuộc họp từ xa đều khó điều tiết hơn so với họp mặt đối mặt.

Soát xét tài liệu điện tử được đặt trong một mạng dùng chung là một sự lựa chọn thay thế cho họp truyền thống. Trong cách tiếp cận này, các thành viên sử dụng các tính năng của bộ xử lý từ để thêm các chú thích (comments) vào văn bản. Mỗi chú thích của một thành viên được gán nhãn tên của thành viên đó, mỗi thành viên đều có thể xem người khác nói gì. Các cuộc họp từ xa có thể bị mất đi 25% hiệu quả so với họp truyền thống.

II. KIỂM THỬ YÊU CẦU (TESTING REQUIREMENTS)

Thường rất khó để có thể hình dung cụ thể về cách hành vi của một hệ thống trong các hoàn cảnh cụ thể chỉ bằng cách đọc SRS. Các test cases dựa trên yêu cầu chức năng hoặc test cases được dẫn từ các use cases sẽ giúp làm sáng tỏ hành vi của hệ thống trong cái nhìn của những người tham gia dự án. Làm các test cases sẽ giúp làm phát lộ nhiều vấn đề của yêu cầu, thậm chí cả khi không cần thực thi tests trên hệ thống thực (Beizer 1990). Nếu bạn bắt đầu xây dựng các test cases ngay từ sớm cho các phần yêu cầu đã ổn định thì bạn có thể thấy các vấn đề trong khi vẫn còn đủ sớm để sửa chữa chúng mà không tốn quá nhiều tiền.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Việc viết các black box test cases và các functional test cases thúc đẩy các suy nghĩ của bạn về hành vi của hệ thống sẽ như thế nào trong các điều kiện cụ thể. Các yêu cầu mờ ảo và nhập nhằng sẽ bị phát hiện do chúng không thể giúp bạn mô tả đáp ứng kỳ vọng của hệ thống. Khi nhà phân tích, nhà phát triển, khách hàng cùng duyệt qua (walk through) các test cases, họ sẽ chia sẻ cùng nhau một cái nhìn chung về hành vi kỳ vọng của hệ thống.

Bạn có thể dẫn ra các functional test cases mức ý tưởng từ các use cases ngay từ đầu quy trình phát triển (Ambler 1995; Collard 1999). Khi đó bạn có thể sử dụng các test cases đó để kiểm tra các đặc tả yêu cầu bằng ngôn ngữ tự nhiên và phân tích các mô hình (ví dụ dialog maps), đánh giá các nguyên mẫu. Các test cases như vậy, dựa trên các kịch bản sử dụng đã dự đoán (anticipated usage scenarios), có thể được dùng như cơ sở cho kiểm thử chấp nhận của khách hàng. Bạn cũng có thể soạn thảo chúng thành các test cases chi tiết và các thủ tục để kiểm thử hệ thống chính thức (Hsia, Kung, and Sell 1997). Câu hỏi cơ bản mà bạn muốn khách hàng trả lời khi định nghĩa tiêu chuẩn chấp nhận là “Anh làm thế nào để nhận biết được nếu phần mềm thực sự làm những gì anh tìm kiếm?” Nếu khách hàng không có câu trả lời cho mỗi tính năng hoặc mỗi use case thì bạn cần làm sáng tỏ yêu cầu.

Ý tưởng kiểm thử yêu cầu có thể khá trừu tượng đối với bạn. Một ví dụ có thể minh họa khái niệm này lấy từ CTS. Dưới đây là một yêu cầu kinh doanh (business requirement), một use case, một yêu cầu chức năng, một phần của dialog map, một test case liên quan đến tác vụ đề xuất một hóa chất.

Business requirement. Một trong những động lực chính để xây dựng hệ thống là yêu cầu sau:

CTS sẽ giảm bớt chi phí mua sắm hóa chất bằng cách thúc đẩy việc sử dụng lại các công-ten-nơ hóa chất đã có sẵn trong công ty.

Use case. Một use case sóng đôi với yêu cầu kinh doanh trên là “Đề xuất một hóa chất” (Request a chemical), use case bao hàm một chuỗi hành động cho phép người dùng đề xuất một công-ten-nơ hóa chất đã có sẵn trong kho hóa chất. Dưới đây là use-case description (Xem Hình 8-3 để thấy chi tiết hơn):

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Use case ID: USE CASES-5					
Tên use case: Request a Chemical					
Người tạo: Tim	Người cập nhật lần cuối: Janice				
Ngày tạo: 4/10	Ngày cập nhật lần cuối: 27/10				
Actor: Requester					
<p>Mô tả: Requester đặc tả hóa chất đề nghị, hoặc nhập chemical ID number của hóa chất đó, hoặc nhập cấu trúc của hóa chất từ một công cụ vẽ cấu trúc hoá chất. Hệ thống có thể đáp ứng đề nghị hoặc bằng cách đưa ra cho Requester một hộp hoá chất mới, hoặc một hộp chứa hoá chất dùng rồi mà trong kho đã có sẵn hoặc một thông báo Requester tự đặt hàng từ một nhà cung cấp bên ngoài.</p>					
<p>Tiền tình huống:</p> <ol style="list-style-type: none"> 3. User ID được cấp quyền. 4. CSDL kho hóa chất được trực tuyến. 					
<p>Hậu tình huống:</p> <ol style="list-style-type: none"> 3. Các đề nghị được lưu trữ đầy đủ trong Chemical Tracking System 4. Đề nghị được gửi qua email tới Chemical Stockroom hoặc Bộ phận mua sắm để thực hiện. 					
Ưu tiên: Cao					
Tần suất sử dụng: Xấp xỉ 5 lần 1 tuần trên mỗi nhà hoá học, 100 lần mỗi tuần trên mỗi nhân viên quản lý kho hóa chất.					
Tiến trình chuẩn: 5.0 Request a Chemical from a Vendor					
<table border="1"> <thead> <tr> <th>Hành động của Actor</th><th>Đáp ứng của hệ thống</th></tr> </thead> <tbody> <tr> <td> 1. Nhập Chemical ID number hoặc tên của file chứa cấu trúc hoá học 4. Xác định nhà cung cấp (tiếp tục) hoặc kho hóa chất </td><td> 2. Kiểm tra tính hợp lệ của Chemical ID 3. Hỏi Requester là muốn đặt hàng từ một nhà cung cấp mới hoặc muốn một hộp hoá chất từ kho 5. <tiếp tục đối thoại cho đến khi đề nghị được thực hiện> </td></tr> </tbody> </table>		Hành động của Actor	Đáp ứng của hệ thống	1. Nhập Chemical ID number hoặc tên của file chứa cấu trúc hoá học 4. Xác định nhà cung cấp (tiếp tục) hoặc kho hóa chất	2. Kiểm tra tính hợp lệ của Chemical ID 3. Hỏi Requester là muốn đặt hàng từ một nhà cung cấp mới hoặc muốn một hộp hoá chất từ kho 5. <tiếp tục đối thoại cho đến khi đề nghị được thực hiện>
Hành động của Actor	Đáp ứng của hệ thống				
1. Nhập Chemical ID number hoặc tên của file chứa cấu trúc hoá học 4. Xác định nhà cung cấp (tiếp tục) hoặc kho hóa chất	2. Kiểm tra tính hợp lệ của Chemical ID 3. Hỏi Requester là muốn đặt hàng từ một nhà cung cấp mới hoặc muốn một hộp hoá chất từ kho 5. <tiếp tục đối thoại cho đến khi đề nghị được thực hiện>				
Tiến trình thay thế: 5.1 Request a Chemical from the Chemical Stockroom (rẽ nhánh kề từ 5.0.4)					
<table border="1"> <thead> <tr> <th>Hành động của Actor</th><th>Đáp ứng của hệ thống</th></tr> </thead> <tbody> <tr> <td> 2. Tùy chọn (optionally), yêu cầu cho xem tình hình (history) của bất </td><td> 1. Hiển thị một danh sách các hộp chứa hóa chất mong muốn còn ở trong kho. </td></tr> </tbody> </table>		Hành động của Actor	Đáp ứng của hệ thống	2. Tùy chọn (optionally), yêu cầu cho xem tình hình (history) của bất	1. Hiển thị một danh sách các hộp chứa hóa chất mong muốn còn ở trong kho.
Hành động của Actor	Đáp ứng của hệ thống				
2. Tùy chọn (optionally), yêu cầu cho xem tình hình (history) của bất	1. Hiển thị một danh sách các hộp chứa hóa chất mong muốn còn ở trong kho.				

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

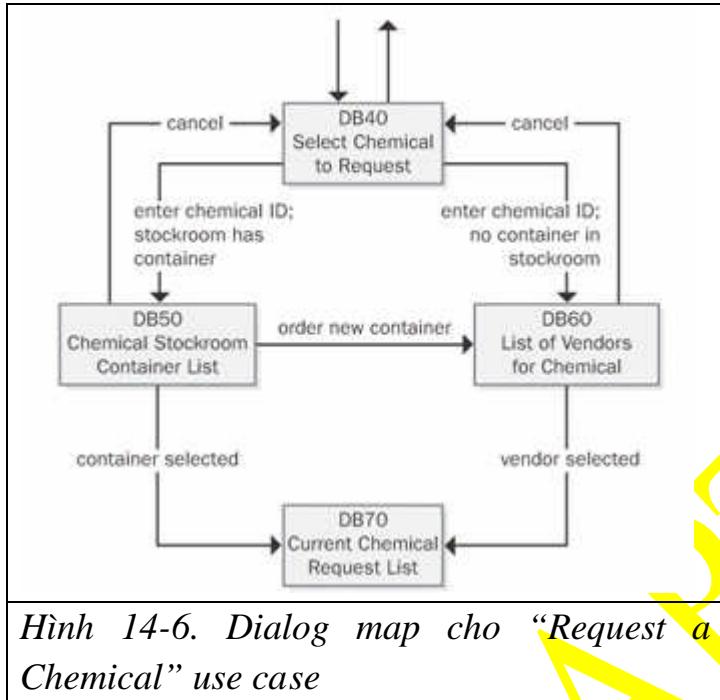
ky hộp hoá chất nào	
3. Chọn một hộp cũ thê hoặc yêu cầu đặt hàng từ một nhà cung cấp	
Các loại trừ: 5.E.1 Chemical Is Not Commercially Available	
Hành động của Actor	Đáp ứng của hệ thống
3. Đề nghị một hoá chất khác.	<ol style="list-style-type: none"> 1. Hiển thị message: Không Nhà cung cấp. 2. Hỏi Requester liệu có đề nghị một hoá chất khác hoặc thoát khỏi chương trình. 4. Bắt đầu lại Tiến trình Chuẩn.
Includes: UC-12 Enter Change Number	
Yêu cầu đặc biệt: Hệ thống có thể nhập một cấu trúc hoá học theo một khuôn dạng đã được mã hoá chuẩn từ bất cứ thiết bị vẽ cấu trúc hoá học nào.	
Giả định: Cấu trúc hoá học được nhập vào hệ thống được giả định hợp lệ	
Ghi chú: Tim sẽ tìm hiểu xem liệu sự chấp thuận của cấp quản lý có cần thiết khi một hoá chất trên mức nguy hiểm cấp 1 được đề nghị hay không. Hạn ngày 4/11 (Due date).	

Requester đặc tả hóa chất để xuất, hoặc bằng cách nhập chemical ID number hoặc bằng cách import cấu trúc của nó từ một công cụ vẽ cấu trúc hóa chất. Hệ thống có thể đáp ứng request này hoặc bằng cách đưa ra cho Requester một công-ten-nơ mới hoặc công-ten-nơ đã sử dụng rồi chia hóa chất đó từ chemical stockroom hoặc bằng cách cho phép Requester đặt hàng từ một nhà cung cấp bên ngoài.

Functional requirement. Dưới đây là một phần của chức năng liên quan đến việc cho phép người dùng lựa chọn một hóa chất đã có thay vì đặt hàng một hóa chất đã có từ bên ngoài.

Nếu stockroom có các công-ten-nơ hóa chất đang được yêu cầu, hệ thống phải hiển thị một danh sách các công-ten-nơ sẵn sàng đó. Người dùng hoặc là lựa chọn một công-ten-nơ, hoặc đề nghị đặt hàng mua một công-ten-nơ mới.

Dialog map. Hình 14-6 minh họa một phần của Dialog map cho “Request a Chemical” use case gắn liền với chức năng này. Các box biểu diễn cho các phần tử đối thoại giữa người dùng và hệ thống, các mũi tên biểu diễn các đường đi có thể từ một phần tử dialog này tới một phần tử dialog khác.



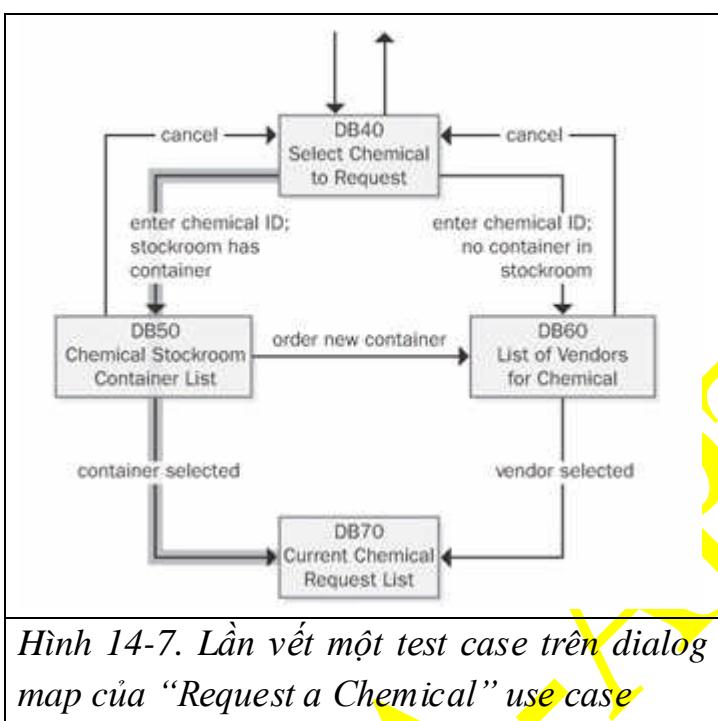
Test case. Do use case này có một số đường thực thi có thể (possible execution paths), nên bạn có thể hình dung nhanh một số ít test cases để kiểm thử tiến trình chính, các tiến trình thay thế, các loại trừ. Dưới đây chỉ là một test case, được thiết kế dựa trên path thể hiện cho người dùng thấy các công-ten-no sẵn có trong kho. Test case này được suy ra từ các use-case description của tác vụ và từ dialog map trong Hình 14-6.

Tại dialog box DB40, nhập một chemical ID hợp lệ; chemical stockroom có hai công-ten-no của hóa chất này. Dialog box DB50 xuất hiện thể hiện số lượng 2 công-ten-no. Lựa chọn công-ten-no thứ hai. DB50 đóng lại và công-ten-no 2 được thêm vào cuối của Current Chemical Request List trong DB70.

Ramesh, test leader của CTS đã viết một số test cases như vậy, căn cứ trên hiểu biết đã có về cách mà người dùng tương tác với hệ thống khi đề xuất một hóa chất. Anh ta vạch ra mỗi test case căn cứ trên yêu cầu chức năng tương ứng để chắc

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

chắn rằng mỗi test case có thể được “thực thi” bởi tập các yêu cầu hiện có và chắc chắn ít nhất mỗi test case bao phủ một yêu cầu chức năng. Tiếp theo, Ramesh lần theo đường thực thi (execution path) của mỗi test case trên dialog map bằng một đường đậm. Đường đậm trên Hình 14-7 thể hiện test case mẫu trên làn vết trên dialog map.



Bằng cách lần theo đường thực thi của mỗi test case, bạn có thể tìm thấy các yêu cầu không đúng hoặc yêu cầu bị khuyết (incorrect or missing requirements), sửa chữa các lỗi trên dialog map, làm mịn các test cases. Ví dụ, giả sử sau khi “thực thi” tất cả các test cases theo cách này mà đường đi có nhãn “Order new container” từ DB50 đến DB60 không được tô đậm thì có 2 cách diễn giải sau:

- Đường đi (navigation) này không phải là hành vi được phép của hệ thống, do vậy mà phải bị xóa khỏi dialog map, và nếu SRS chưa một yêu cầu đặc tả sự biến chuyển từ DB50 đến DB60 thì yêu cầu này phải bị xóa bỏ.
- Đường đi này là một hành vi hệ thống hợp lệ nhưng test case chứng minh hành vi này bị khuyết.

Tương tự, giả sử rằng một test case xác định rằng người dùng có thể thực hiện một hành động nào đó để di chuyển trực tiếp từ DB40 đến DB70. Tuy nhiên, dialog map này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com.

map trên Hình 14-6 lại không chứa một đường như vậy, vì vậy test case không thể được thực thi với yêu cầu hiện có (existing requirement) đó. Thêm nữa, có 2 cách diễn giải có thể có, và bạn sẽ cần để xác định nhận định đúng hay không:

- Đường đi từ DB40 đến DB70 không phải là một hành vi được phép của hệ thống, vì vậy test case đó sai.
- Đường đi từ DB40 đến DB70 là một hành vi hợp lệ, nhưng dialog map và có lẽ SRS bị khuyết yêu cầu cho phép bạn thực thi test case.

Trong các ví dụ trên, nhà phân tích và người kiểm thử kết hợp các yêu cầu, mô hình phân tích, test cases để phát hiện các yêu cầu khuyết, sai, không cần thiết rất lâu trước khi mã nguồn được viết. Kiểm thử mức ý tưởng yêu cầu phần mềm là một kỹ thuật mạnh để kiểm soát chi phí và lịch biểu của dự án bằng cách tìm các lỗi yêu cầu ngay từ rất sớm.

Tập hợp và tài liệu hóa các yêu cầu của bạn là điểm bắt đầu tốt nhất cho sự thành công của dự án. Bạn cũng cần phải chắc chắn chúng là các yêu cầu đúng (right requirements) và chúng thể hiện tất cả các đặc tính của lời phát biểu yêu cầu tuyệt vời (excellent requirement statements). Nếu bạn kết hợp ngay từ sớm cách thiết kế kiểm thử hợp lý với việc soát xét yêu cầu không chính thức, thanh tra SRS và các kỹ thuật kiểm tra yêu cầu khác, thì bạn sẽ xây dựng được các hệ thống chất lượng cao với ít thời gian hơn và chi phí hơn so với trước đây.

Các bước tiếp theo

- Chọn ngẫu nhiên một trang mô tả yêu cầu chức năng từ SRS của bạn. Triệu tập mọi người thành một nhóm nhỏ đại diện cho các stakeholders khác nhau và kiểm tra cẩn thận yêu cầu để tìm bất cứ một độ lệch nào so với các đặc tính của lời phát biểu yêu cầu tuyệt vời (excellent requirement statements).
- Nếu bạn tìm đủ các vấn đề bằng cách soát xét ngẫu nhiên khiến cho những người tham gia soát xét thấy lo lắng về chất lượng tổng thể của yêu cầu thì hãy thúc đẩy người dùng và những người đại diện phát triển thanh tra toàn bộ SRS. Đào tạo các thành viên của nhóm về quy trình thanh tra để công việc đạt hiệu quả cao nhất.

- Định nghĩa các test cases mức ý tưởng cho một use case hoặc một phần của SRS chưa được mã hóa. Hãy xác định liệu các stakeholders có đồng ý rằng các test cases phản ánh đúng hành vi kỳ vọng của hệ thống. Chắc chắn bạn đã định nghĩa tất cả các yêu cầu chức năng cho phép test cases được “thực thi” và không có yêu cầu nào thừa.

SATA-APTECH

CHƯƠNG 15

NHÌN XA HƠN VIỆC PHÁT TRIỂN YÊU CẦU

Mặc dù với một sự bắt đầu không chắc chắn lắm, CTS vẫn đang được tiến hành suôn sẻ. Người bảo trợ dự án, Gerhard và người đại diện sản phẩm kho hóa chất, Roxanne, vẫn còn hoài nghi về sự cần thiết phải dành một lượng thời gian lớn cho việc thu thập yêu cầu. Tuy nhiên, họ đang muốn tổ chức một buổi học 1 ngày về yêu cầu phần mềm cho nhóm phát triển và những người đại diện sản phẩm. Lớp học này sẽ nhấn mạnh đến tầm quan trọng của việc các stakeholders đạt được sự đồng thuận về yêu cầu trước khi mã hóa. Lớp học trình bày cho tất cả mọi người các thuật ngữ về yêu cầu, các khái niệm, các thực hành cần thiết, từ đó thúc đẩy họ sử dụng các kỹ thuật làm yêu cầu.

Khi dự án đang được tiến hành, Gerhard đã nhận được sự phản hồi rất tốt từ những đại diện của người dùng về quy trình phát triển yêu cầu. Gerhard đã mời nhóm phát triển và các đại diện người dùng đi ăn trưa để cảm ơn họ về việc đạt được một mốc quan trọng là các yêu cầu đã được vạch ranh giới của dự án CTS. Tại bữa ăn, Gerhard cảm ơn những người thực hiện công việc suy luận yêu cầu và nói, “Bây giờ chúng ta đã có yêu cầu rồi, tôi mong sớm nhìn thấy nhóm phát triển thực hiện công việc mã hóa của mình.”

“Chúng tôi chưa sẵn sàng để viết mã đâu,” quản trị dự án nói với Gerhard, “Chúng tôi đã lập kế hoạch để có thể phát hành nhiều phiên bản của sản phẩm, chúng tôi cần suy nghĩ về cách tốt nhất để thiết kế hệ thống sao cho có thể dễ dàng điều chỉnh sau này. Nguyên mẫu của chúng tôi cho một số ý tưởng tốt về cách tiếp cận yêu cầu và giúp chúng tôi nắm bắt tốt hơn cách mà người dùng muốn các đặc tính mà giao diện phải có. Nếu chúng tôi dành thời gian để thiết kế phần mềm thì chúng tôi sẽ không phải đổi đầu với những vấn đề như thêm bớt các chức năng sau này.”

Gerhard cảm thấy hơi chút bức bối. Một lần nữa, có vẻ như nhóm phát triển không làm công việc thật sự của mình là lập trình. Hay là Gerhard đã nôn nóng quá?

Những nhà quản trị dự án và người phát triển có kinh nghiệm đều hiểu tầm quan trọng của việc biến đổi bản yêu cầu thành một bản thiết kế tốt và bản kế hoạch dự án hợp lý. Chương này giới thiệu ngắn gọn về một số cách tiến hành những công việc giữa phát triển yêu cầu và phát hành các phiên bản sản phẩm bằng cách kết nối yêu cầu với kế hoạch dự án, thiết kế hệ thống, mã hóa và kiểm thử sản phẩm.

I. TỪ YÊU CẦU ĐẾN KẾ HOẠCH DỰ ÁN (FROM REQUIREMENTS TO PROJECT PLANS)

Do các yêu cầu định nghĩa kết quả mong muốn của dự án, nên kế hoạch dự án, các ước lượng, lịch biểu cần phải hoàn toàn dựa trên yêu cầu.

YÊU CẦU VÀ LỊCH BIỂU

Nhiều dự án phần mềm lập lịch biểu theo kiểu *right-to-left*, theo đó ngày bàn giao sản phẩm đã được định ra và sau đó thì các yêu cầu mới được định nghĩa. Cách làm này thường khiến các nhà phát triển không thể đáp ứng được việc bàn giao sản phẩm đúng ngày mà sản phẩm vẫn có đầy đủ các chức năng ở mức chất lượng kỳ vọng. Sẽ thực tế hơn nếu định nghĩa yêu cầu phần mềm trước khi lập kế hoạch chi tiết và các cam kết khác. Tuy nhiên, một chiến lược *thiết kế-tới-lịch biểu (design-to-schedule)* cũng có thể được thực hiện nhằm cung cấp cho bạn một khoảng rộng để đàm phán về những gì có thể cắt bớt hoặc thêm vào cho phù hợp với những ràng buộc của lịch biểu.

Đối với các hệ thống phức tạp trong đó phần mềm chỉ là một phần của sản phẩm cuối cùng, các lịch biểu mức cao thường được thiết lập sau khi các yêu cầu mức sản phẩm (mức hệ thống) đã được sinh ra. Khi đó yêu cầu hệ thống được phân rã và được phân bổ vào các hệ thống con phần mềm và phần cứng khác nhau. Tại thời điểm này, những ngày bàn giao chính được được thiết lập và thỏa thuận căn cứ trên đầu vào từ các nguồn khác nhau, gồm marketing, sales, dịch vụ khách hàng, phát triển. Nếu lịch biểu đã bị ràng buộc thì nhóm phát triển liên chức năng cần phải thực hiện các quyết định đánh đổi về chức năng, chất lượng và chi phí.

Bạn cần suy nghĩ về việc lập kế hoạch và giải ngân cho dự án thành nhiều giai đoạn. Giai đoạn đầu tiên, duyệt qua yêu cầu (requirements exploration), sẽ cung cấp đủ thông tin để cho phép bạn lập các kế hoạch thực tế và các ước lượng cho Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

một hoặc nhiều giai đoạn thi công. Các dự án có những yêu cầu không chắc chắn có thể sử dụng chu trình phát triển lặp và tăng dần. Định nghĩa các ưu tiên của yêu cầu cho phép bạn xác định chức năng nào cần phải được thực hiện ngay và cứ thế cho các chức năng tiếp theo.

Các dự án phần mềm thường thất bại trong việc đáp ứng các mục tiêu là do các nhà phát triển và những người tham gia khác là những người lập kế hoạch tồi, chứ không phải do họ là những kỹ sư phần mềm tồi. Các lỗi chính khi lập kế hoạch thường là việc lờ đi các công việc cần làm khi lập kế hoạch một dự án thông thường, như ước lượng dưới mức cần thiết thời gian và nỗ lực (effort), sai khi tính toán các rủi ro, không kỳ vọng sẽ dành thời gian cho những gì cần làm lại. Lập kế hoạch dự án chính xác thường đòi hỏi các yếu tố sau:

- Kích thước của sản phẩm đã được ước lượng cẩn cứ trên việc hiểu rõ yêu cầu.
- Năng suất đã được biết trước của nhóm phát triển căn cứ trên hiệu năng trong quá khứ.
- Một danh sách đã được biết rõ về các tác vụ cần để thi công và kiểm tra một tính năng hoặc một use case.
- Quy trình ước lượng và lập kế hoạch hiệu quả.
- Kinh nghiệm của những người lập kế hoạch.

YÊU CẦU VÀ ƯỚC LƯỢNG

Bước đầu tiên khi ước lượng dự án là tìm mối liên quan giữa yêu cầu và kích thước của sản phẩm. Bạn có thể ước lượng kích thước căn cứ trên các yêu cầu thành văn, các mô hình phân tích bằng hình ảnh (graphical analysis model), các nguyên mẫu, hoặc các thiết kế giao diện người dùng. Mặc dù không có phép đo kích thước nào là hoàn hảo, nhưng dưới đây là một số gợi ý:

- Điểm chức năng hoặc điểm tính năng (Jones 1996b), hoặc điểm chức năng 3-D (Whitmire 1995).
- Số lượng, kiểu và độ phức tạp của các phần tử giao diện người dùng đồ họa.
- Số lượng dòng lệnh cần thiết để thực thi các yêu cầu cụ thể.
- Đếm các lớp đối tượng, hoặc các phép đo khác cho các hệ thống hướng đối tượng (Whitmire 1997).

- Số lượng các yêu cầu có thể kiểm thử riêng biệt (Wilson 1995).

Tất cả các phương pháp trên có thể được dùng để ước lượng kích thước, nhưng bất cứ cách nào thì bạn cũng vẫn phải dựa trên kinh nghiệm của mình.

Các yêu cầu mờ ảo, nhập nhằng sẽ dẫn tới sự không chắc chắn khi ước lượng kích thước sản phẩm, do vậy mà bạn ước lượng sai nỗ lực (effort) và lịch biểu. Do sự không chắc chắn của yêu cầu là không thể tránh được ngay từ sớm trong dự án, nên hãy định kỳ ước lượng lại lịch biểu, ngân sách và các nỗ lực cần thiết như một sự hiệu chỉnh.

II. TỪ YÊU CẦU ĐẾN THIẾT KẾ VÀ MÃ HÓA (FROM REQUIREMENTS TO DESIGNS AND CODE)

Có một vùng xám giữa yêu cầu và thiết kế, nhưng hãy cố gắng giữ các đặc tả độc lập với việc thực thi chừng nào có thể. Tốt nhất là sự mô tả về những gì mà hệ thống cần làm không bị làm lệch đi bởi những cân nhắc về thiết kế (Jackson 1995). Phát triển và đặc tả yêu cầu cần tập trung vào tìm hiểu và mô tả các hành vi có thể quan sát được từ bên ngoài của hệ thống. Lôi kéo những người thiết kế và người phát triển vào các phiên thanh tra yêu cầu để chắc chắn rằng yêu cầu có thể tạo cơ sở cho việc thiết kế.

Cần có một loạt thiết kế đáp ứng được yêu cầu mục tiêu, các thiết kế biến thiên về mặt hiệu năng (performance), hiệu quả (efficiency), khả năng bền vững, và các biện pháp kỹ thuật được sử dụng. Nếu bạn nhảy trực tiếp từ yêu cầu tới mã hóa thì về cơ bản bạn đang làm hỏng dự án của mình. Hãy suy ngẫm kỹ về những cách hiệu quả để xây dựng hệ thống trước khi bắt tay xây dựng nó. Suy nghĩ về các lựa chọn thiết kế sẽ giúp bạn đảm bảo rằng các nhà phát triển tôn trọng các ràng buộc thiết kế đã xác định hoặc các đặc tả chất lượng liên quan đến thiết kế.

Trước đây tôi đã làm việc trong một dự án mà việc phân tích yêu cầu được thực hiện rất chu đáo, chúng tôi xây dựng chi tiết sơ đồ luồng dữ liệu (DFD, data flow diagram) minh họa một dãy 8 quy trình biến đổi hành vi của một hệ thống xử lý ảnh. Sau khi đã hoàn thành việc phân tích, chúng tôi không bị lôi cuốn bởi việc phải mã hóa hệ thống ngay. Thay vì vậy, chúng tôi dành thời gian để tạo ra một mô *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

hình thiết kế (design model), mô hình này lại được biểu diễn bởi một DFD. Chúng tôi nhanh chóng nhận ra rằng 3 trong số các bước trong mô hình sử dụng các thuật toán giống hệt nhau, 3 bước khác sử dụng một tập hợp các phương trình khác nhau, 2 bước còn lại thì dùng chung một tập phương trình.

Các mô hình phân tích biểu diễn hiểu biết về bài toán cần giải quyết của khách hàng và nhóm phát triển, trong khi các mô hình thiết kế minh họa cách chúng tôi dự định xây dựng hệ thống phần mềm cho bài toán trên. Bằng cách bỏ qua giai đoạn thiết kế, chúng tôi đã đơn giản hóa bài toán lõi đi khoảng 60%, rút gọn 8 bước tính toán phức tạp xuống còn có 3. Nếu chúng tôi mã hóa ngay sau khi phân tích thì không nghi ngờ gì nữa, chúng tôi sẽ phải sửa lại mã nguồn tại một thời điểm nào đó khi thi công. Tuy nhiên, chúng tôi đã tiết kiệm được thời gian (và do đó là tiền) bằng cách phát hiện cách làm đơn giản ngay từ sớm khi đang thiết kế hệ thống. Làm lại thiết kế thì chắc là tốn ít tiền bạc và thời gian hơn là mã hóa lại.

Các yêu cầu của một sản phẩm phần mềm và thuộc tính chất lượng của chúng xác định những cách tiếp cận kiến trúc thích hợp khi thiết kế (Bass, Clements, and Kazman). Nghiên cứu và đánh giá một kiến trúc được đề xuất là một cách khác để làm sáng tỏ yêu cầu. Đó là cách tiếp cận bottom-up để phân tích yêu cầu, tương tự để xây dựng nguyên mẫu. Cả hai cách này đều xoay quanh tư duy sau: “Nếu tôi hiểu yêu cầu một cách đúng đắn, thì cách tiếp cận này là một phương thức đúng để đáp ứng chúng. Giờ thì tôi có một kiến trúc sơ bộ (hoặc một nguyên mẫu) trong tay, liệu nó có thể giúp tôi hiểu yêu cầu sâu hơn không?”

Bạn không phải phát triển một thiết kế đầy đủ, chi tiết cho toàn bộ sản phẩm trước khi bạn bắt tay thi công một phần yêu cầu nào đó. Tuy nhiên, bạn phải thiết kế mỗi component trước khi mã hóa nó. Lập kế hoạch cho công việc thiết kế mang lại lợi ích lớn cho các dự án khó, các hệ thống với nhiều giao diện và tương tác giữa các components bên trong hệ thống, các dự án mà phần đông những người phát triển thiếu kinh nghiệm cần thiết (McConnell). Tuy nhiên, mọi dự án đều có được lợi ích khi bạn thực hiện các bước sau:

- Xây dựng một kiến trúc vững chắc bao gồm các hệ thống con và các components sẽ được dùng khi bảo trì.

- Xác định các object classes hoặc các functional modules mà bạn phải xây dựng, định nghĩa giao diện giữa chúng, vai trò và sự tương tác của chúng.

Khi người phát triển biến đổi yêu cầu thành thiết kế và mã nguồn thì họ phải đổi mới với những sự nhập nhằng và rối rắm. Một cách lý tưởng là người phát triển chuyển các vấn đề đó cho khách hàng để họ giải quyết. Nếu một vấn đề không thể được giải quyết ngay, thì bất cứ giả định nào hoặc cách diễn giải nào mà người phát triển đề xuất cũng phải được khách hàng chấp nhận. Nếu người phát triển phải đổi mới với nhiều vấn đề như vậy thì tức là yêu cầu đã không được làm đầy đủ (đủ sáng sủa, đủ chi tiết) trước khi thi công.

III. TỪ YÊU CẦU ĐẾN KIỂM THỬ (FROM REQUIREMENTS TO TESTS)

Yêu cầu được đặc tả là cơ sở để kiểm thử hệ thống, việc kiểm thử xác định liệu hệ thống có đáp ứng yêu cầu hay không. Bạn phải kiểm thử sản phẩm cuối cùng dựa trên những gì nó được mong muốn phải làm đã ghi trong SRS, chứ không phải dựa trên thiết kế hoặc mã nguồn. Kiểm thử dựa trên mã nguồn tương tự một lời tiên tri tự thực hiện. Sản phẩm có thể cho thấy đúng tất cả các hành vi đã mô tả trong test cases nhưng điều đó không có nghĩa nó đã thực hiện đúng những gì khách hàng cần. Nếu bạn không có các yêu cầu được ghi thành văn, bạn sẽ phải suy luận ra chúng căn cứ trên việc xây dựng các test cases phù hợp - một cách tiếp cận không đầy đủ và không chính xác. Mời các testers tham gia vào các phiên thanh tra yêu cầu để đảm bảo chắc chắn rằng mọi yêu cầu có thể kiểm tra và có thể coi là cơ sở cho việc kiểm thử hệ thống.

Khi mỗi yêu cầu đã ổn định thì các testers phải tài liệu hóa cách họ sẽ kiểm tra nó – thông qua kiểm thử (testing), thanh tra (inspection), chứng minh (demonstration), hoặc phân tích. Việc tư duy về cách bạn sẽ kiểm tra mỗi yêu cầu như thế nào là một hành động đơn giản mà tự nó đã là một hướng dẫn thực hành chất lượng tốt. Sử dụng các kỹ thuật phân tích như đồ thị nguyên nhân-kết quả để dẫn ra các test cases căn cứ trên logic được mô tả trong mỗi yêu cầu. Cách làm này sẽ để lộ ra các nhập nhằng, các điều kiện “else” (trong if...then) bị khuyết và nhiều vấn đề khác. Mỗi yêu cầu cần được lắn vết tối ít nhất một test case trong bộ test hệ thống của bạn sao cho không một hành vi kỳ vọng nào của hệ thống không được kiểm tra. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Bạn có thể đo lường tiến độ kiểm thử thông qua lượng phần trăm các yêu cầu đã vượt qua các test cases. Những testers kinh nghiệm sẽ xây dựng các test chỉ dựa trên yêu cầu với hiểu biết của họ về chức năng, đặc tính chất lượng sản phẩm.

Việc kiểm thử dựa trên đặc tả yêu cầu ứng dụng một số chiến lược thiết kế test sau: định hướng hành động (action driven), định hướng dữ liệu (data driven) (gồm phân tích giá trị biên, phân lớp tương đương), định hướng logic, định hướng sự kiện, định hướng trạng thái (Poston). Đó là cách tự động để sinh ra các test cases từ đặc tả hình thức, nhưng bạn phải xây dựng bằng tay các test cases từ các đặc tả yêu cầu bằng ngôn ngữ tự nhiên.

Cuối cùng phải nhấn mạnh rằng, kiểm thử hệ thống dựa trên các yêu cầu người dùng là cần thiết nhưng không đầy đủ để đáp ứng nhu cầu sử dụng thực của hệ thống.

IV. TỪ YÊU CẦU TỚI THÀNH CÔNG (FROM REQUIREMENTS TO SUCCESS)

Tôi đã từng làm một dự án mà một nhóm mới được đưa đến để xây dựng hệ thống mà một nhóm khác đã phát triển yêu cầu. Nhóm mới này nhìn vào một tá tài liệu mỗi cuốn dày 3 inch chứa SRS, phát hoảng lên và bắt đầu coding. Họ chẳng hề tham chiếu đến SRS khi xây dựng hệ thống, thay vì vậy họ xây dựng cái mà họ nghĩ cần phải làm, căn cứ trên một hiểu biết không chính xác và không đầy đủ về hệ thống. Tôi không ngạc nhiên khi dự án này phải đổi mặt với nhiều vấn đề. Có gắng hiểu được một số lượng không lồ các yêu cầu (chúng có thể hoặc không được viết ra một cách đầy đủ) đôi khi làm nản chí nhóm phát triển, nhưng không biết về chúng thì chắc chắn dẫn dự án tới thất bại. Sản phẩm phần mềm cuối cùng được chuyển giao phải đáp ứng mọi nhu cầu và kỳ vọng của khách hàng thì dự án thành công. Yêu cầu là bước cơ bản trên con đường đi từ ý tưởng sản phẩm tới sự hài lòng của khách hàng.

PHẦN III QUẢN LÝ YÊU CẦU PHẦN MỀM

SATA-APTECH

CHƯƠNG 16

CÁC NGUYÊN LÝ VÀ THỰC HÀNH VỀ QUẢN LÝ YÊU CẦU

Chương 1 đã phân chia công nghệ yêu cầu thành hai giai đoạn là *phát triển yêu cầu* và *quản lý yêu cầu*. Phát triển yêu cầu gồm các công đoạn suy luận, phân tích, đặc tả, kiểm tra yêu cầu của một dự án. Các sản phẩm điển hình của giai đoạn phát triển yêu cầu như: tài liệu về tầm nhìn và phạm vi dự án, tài liệu use-case, tài liệu SRS, các mô hình phân tích có liên quan.

Trước khi được soát xét và chấp thuận, phải định nghĩa được requirement baseline (ranh giới yêu cầu) trong các tài liệu đó để tập trung các nỗ lực phát triển phiên bản sản phẩm tương ứng. Baseline tạo một thỏa thuận giữa khách hàng và nhóm phát triển về các yêu cầu chức năng và yêu cầu phi chức năng của sản phẩm đã định. Dự án sau đó sẽ có các thỏa thuận thêm về những gì chuyển giao, các ràng buộc, lịch biểu, ngân sách, cam kết bằng hợp đồng, nhưng các chủ đề này không thuộc phạm vi cuốn sách này.

Thỏa thuận requirement baseline bắc cầu nối giữa phát triển yêu cầu và quản lý yêu cầu. Quản lý yêu cầu gồm tất cả các hoạt động duy trì sự toàn vẹn và chính xác của thỏa thuận yêu cầu trong khi dự án tiến triển, như Hình 16-1 chỉ ra.

Quản lý yêu cầu (Requirement Management)			
Kiểm soát thay đổi (Change Control)	Kiểm soát phiên bản (Version Control)	Giám sát trạng thái yêu cầu (Requirement Status Tracking)	Lần vết yêu cầu (Requirement Tracing)
<ul style="list-style-type: none">Đề xuất thay đổiPhân tích ảnh hưởngRa quyết định	<ul style="list-style-type: none">Xác định phiên bản của tài liệu yêu cầuXác định phiên soát xét từng yêu cầu	<ul style="list-style-type: none">Định nghĩa các liên kết với các yêu cầu khácĐịnh nghĩa các liên kết với các phần tử hệ thống	<ul style="list-style-type: none">Định nghĩa trạng thái của yêu cầuGiám sát mỗi yêu cầu đã định nghĩa trạng thái

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

<ul style="list-style-type: none"> • Truyền thông • Tích hợp • Đo lường độ ổn định của yêu cầu 		khác	
<i>Hình 16-1. Các hoạt động chính để quản lý yêu cầu</i>			

Quản lý yêu cầu nhấn mạnh:

- Kiểm soát thay đổi đối với requirement baseline.
- Giữ các kế hoạch dự án phù hợp với tính trạng yêu cầu hiện tại.
- Kiểm soát các phiên bản của từng yêu cầu riêng biệt và của các tài liệu yêu cầu.
- Quản lý mối quan hệ giữa yêu cầu, các liên kết hoặc phụ thuộc giữa các yêu cầu riêng biệt và các phần tử được chuyển giao của dự án.
- Giám sát trạng thái của yêu cầu trong baseline.

Chương này chỉ ra các nguyên lý cơ bản của quản lý yêu cầu. Các chương khác trong Phần III sẽ mô tả cụ thể các thực hành quản lý yêu cầu ở mức chi tiết hơn, gồm kiểm soát thay đổi (Chương 17), lần vết yêu cầu (Chương 18), phân tích ảnh hưởng của thay đổi (cũng được thảo luận trong Chương 18). Phần III kết luận bằng một thảo luận về các công cụ thương mại giúp bạn quản lý yêu cầu (Chương 19).

I. QUẢN LÝ YÊU CẦU VÀ CMM

(Lời người dịch). Mục này không được dịch do CMMI (phiên bản hiện tại của CMM) sẽ được giới thiệu trong một tài liệu riêng, bạn nào có nhu cầu có thể xem trong tài liệu gốc.

II. CÁC THỦ TỤC QUẢN LÝ YÊU CẦU

Bạn cần định nghĩa các bước mà nhóm dự án sẽ thực hiện để quản lý yêu cầu của họ. Ghi chép thành tài liệu các bước này sẽ cho phép các thành viên của tổ chức thực hiện các hoạt động chính của dự án một cách nhất quán và hiệu quả. Các nội dung cần suy nghĩ gồm:

- Công cụ, kỹ thuật, quy ước để kiểm soát các phiên bản của các tài liệu yêu cầu khác nhau hoặc của bản thân mỗi yêu cầu.
- Những cách mà các yêu cầu mới hoặc những thay đổi đối với các yêu cầu cũ được đề xuất, xử lý, đàm phán, truyền thông tới tất cả người có liên quan với dự án.
- Các yêu cầu được vạch ranh giới như thế nào.
- Các trạng thái của yêu cầu mà bạn sẽ sử dụng, cũng như ai được phép thay đổi chúng.
- Các thủ tục báo cáo và giám sát trạng thái của yêu cầu.
- Các bước cần tuân theo để phân tích ảnh hưởng của các thay đổi được đề xuất.
- Các thay đổi trong yêu cầu sẽ được phản ánh như thế nào trong kế hoạch và cam kết của dự án, và dưới những điều kiện nào.

Bạn có thể gói gọn tất cả các thông tin trên trong một tài liệu duy nhất. Hay cách khác, bạn có thể thích viết các thủ tục riêng biệt hơn như, thủ tục kiểm soát một thay đổi, thủ tục phân tích một ảnh hưởng, thủ tục giám sát một trạng thái. Các thủ tục đó sẽ rất có ích cho nhiều dự án do đã quy trình hóa các chức năng chung mà mọi dự án đều phải tuân theo.

III. KIỂM SOÁT PHIÊN BẢN CỦA ĐẶC TẢ YÊU CẦU

“Cuối cùng tôi đã hoàn thành việc thực thi tính năng báo cáo sắp xếp lại kho hàng”, Shari nói tại cuộc họp hàng tuần của dự án.

*“Ồ, khách hàng đã hủy tính năng đó cách đây 2 tuần rồi,” trưởng dự án trả lời.
“Anh không nhận được SRS đã được cập nhật à?”*

Nếu bạn từng nghe một cuộc đối thoại như vậy, thì bạn đã biết được sự thiệt hại, sự mất tinh thần của nhóm dự án như thế nào khi các thành viên phải dành thời gian, công sức phát triển một tính năng mà sau đó thì được biết nó đã bị loại từ lâu. Tôi biết trường hợp một nhóm phát triển đã nhận được một lô báo cáo lỗi sau khi đã chuyển giao một phiên bản mới cho nhóm kiểm thử. Các kiểm thử viên đã làm việc trên một SRS lỗi thời so với phiên bản mới này. Nhóm đã tiêu tốn thời gian để cố

gắng tìm kiếm lỗi theo phiên bản SRS cũ, sau đó họ lại lặp lại công việc đó cho phiên bản SRS mới.

Kiểm soát phiên bản là một khía cạnh cơ bản của việc quản lý yêu cầu. Mỗi phiên bản của các tài liệu yêu cầu cần được định danh duy nhất. Mỗi thành viên của nhóm phải truy nhập phiên bản hiện thời của yêu cầu, các thay đổi cần được tài liệu hóa rõ ràng và được truyền thông đầy đủ tới tất cả mọi người. Để tối thiểu hóa sự lộn xộn, sự xung đột, và sự rối loạn truyền thông nên chỉ những cá nhân được chỉ định trước mới được quyền cập nhật tài liệu yêu cầu. Hành động tương tự cũng được áp dụng cho tất cả các tài liệu quan trọng của dự án.

Mỗi phiên bản đã được phát hành của tài liệu yêu cầu cần có thông tin về lịch sử soát xét để cho người đọc biết những thay đổi nào đã diễn ra, ngày mỗi thay đổi diễn ra, cá nhân thực hiện sự thay đổi, lý do của mỗi thay đổi. Bạn có thể sử dụng các đánh dấu soát xét tiêu chuẩn, như gạch ngang để chỉ đoạn bị xoá, gạch dưới để chỉ sự sửa chữa, dòng gạch thẳng đứng bên lề để chỉ vị trí mỗi thay đổi. Do các ký hiệu này có thể làm lộn xộn tài liệu, nên các bộ xử lý từ có hỗ trợ soát xét sẽ cho bạn in hoặc văn bản đang được soát xét hoặc văn bản cuối cùng. Hãy suy nghĩ cách lập một số phiên bản cho mỗi yêu cầu riêng biệt, chúng có thể tăng ngay khi yêu cầu được chỉnh sửa.

Cách đơn giản nhất để kiểm soát phiên bản là gán nhãn (bằng tay) cho mỗi phiên soát xét SRS phù hợp với một quy ước tiêu chuẩn. Các sơ đồ giúp phân biệt giữa các phiên bản của tài liệu dựa trên ngày soát xét, hoặc ngày tài liệu được in dễ dẫn đến nhầm lẫn, vì vậy tôi không khuyên bạn sử dụng chúng. Tôi đã sử dụng cách gán nhãn là “version 1.0 draft 1” cho phiên bản đầu tiên của một tài liệu mới. Bản thảo thứ hai là “version 1.0 draft 2”, số bản thảo cứ tăng dần với mỗi vòng lặp cho đến khi tài liệu được chấp thuận và được vạch ranh giới. Tại thời điểm này, nhãn sẽ là “version 1.0 approved”. Phiên bản tiếp theo sẽ là “version 1.1 draft 1” cho một soát xét nhỏ hoặc “version 2.0 draft 1” cho một thay đổi lớn (dĩ nhiên “nhỏ”, “lớn” ở đây chỉ là tương đối). Cách gán nhãn này phân biệt rõ giữa các phiên bản tài liệu là bản thảo hay đã được vạch ranh giới, nhưng bù lại nó yêu cầu phải thực hiện mọi thứ bằng tay.

Một mức kiểm soát phiên bản phức tạp hơn liên quan đến việc lưu trữ tài liệu yêu cầu trong một công cụ kiểm soát phiên bản, ví dụ các công cụ được dùng để kiểm soát mã nguồn thông qua các thủ tục check-in và check-out. Nhiều công cụ quản lý thương mại đã đáp ứng mục đích này. Tôi biết một dự án lưu trữ hàng trăm tài liệu use-case được viết trong MS Word như một công cụ kiểm soát phiên bản. Công cụ này cho phép các thành viên truy nhập mỗi phiên bản trước đó của mỗi tài liệu use-case, công cụ cũng tạo ra một log để mô tả lịch sử của các thay đổi của mỗi tài liệu. Các nhà phân tích yêu cầu của dự án phải truy nhập read/write vào các tài liệu được lưu trữ trong công cụ, trong khi các thành viên khác của nhóm chỉ được truy nhập read-only. Cách kiểm soát này cũng khá tốt.

Cách tốt nhất để kiểm soát phiên bản là lưu trữ yêu cầu trong cơ sở dữ liệu của một công cụ có sẵn cho mục đích này như mô tả trong Chương 19. Các công cụ đó có thể giám sát và báo cáo lịch sử đầy đủ của các thay đổi đối với mỗi yêu cầu, chúng rất có giá trị nếu bạn muốn quay lại phiên bản trước đó của một yêu cầu. Các lời dẫn giải mô tả lý do đằng sau một quyết định thêm, thay đổi, xóa bỏ, hoặc từ chối một yêu cầu được hỗ trợ đầy đủ nếu yêu cầu gây nên tranh cãi sau này.

IV. THUỘC TÍNH CỦA YÊU CẦU

Để làm sáng tỏ mô tả bằng lời của mỗi yêu cầu chức năng, người ta cần một số thông tin khác nữa - gọi là thuộc tính của yêu cầu. Các thuộc tính đó tạo lập một bối cảnh và nền tảng cho mỗi yêu cầu có thể thực thi tốt chức năng mong muốn. Các giá trị của thuộc tính có thể được lưu trữ trong một bảng tính (spreadsheet), một cơ sở dữ liệu, hoặc một công cụ quản lý yêu cầu. Các công cụ thương mại đã cung cấp một số thuộc tính được hệ thống sinh ra, ngoài ra bạn có thể định nghĩa các thuộc tính khác với các kiểu dữ liệu khác nhau. Các công cụ như vậy cho phép bạn lọc, sắp xếp, và truy vấn cơ sở dữ liệu để xem các tập con yêu cầu được chọn dựa trên giá trị của thuộc tính.

Một tập hợp phong phú các thuộc tính là đặc biệt quan trọng đối với các dự án lớn, phức tạp. Hãy cân nhắc đặc tả các thuộc tính như sau đối với mỗi yêu cầu của bạn:

- Ngày yêu cầu được tạo ra.
- Số phiên bản của yêu cầu.
- Tác giả của yêu cầu.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- Người có trách nhiệm đảm bảo rằng yêu cầu được thỏa mãn.
- Trạng thái của yêu cầu.
- Nguồn gốc của yêu cầu hoặc cơ sở lý luận ẩn đằng sau nó (hoặc một tham chiếu tới vị trí thông tin này có thể được tìm thấy).
- Hệ thống con nào mà yêu cầu được phân bổ.
- Phiên bản sản phẩm nào mà yêu cầu được phân bổ.
- Phương pháp kiểm tra nào được sử dụng hoặc tiêu chuẩn kiểm thử nào được chấp nhận.
- Mức ưu tiên hoặc tầm quan trọng đối với sản phẩm (đặc tả thành mức cao, trung bình, thấp, hoặc bạn có thể định nghĩa các thuộc tính riêng theo 4 chiều kích về sự ưu tiên được mô tả ở Chương 13: lợi ích, trùng phạt, chi phí, rủi ro).
- Độ ổn định của yêu cầu (một chỉ số về mức độ thay đổi của yêu cầu trong tương lai, yêu cầu không ổn định có thể chỉ ra rằng bạn đang cố gắng tự động hóa các quy trình nghiệp vụ không lặp lại, đang bị rối loạn, được định nghĩa kém).

Định nghĩa và cập nhật các giá trị thuộc tính đó là một phần của chi phí quản lý yêu cầu. Lựa chọn tập con tối thiểu các thuộc tính sẽ giúp bạn quản lý dự án hiệu quả hơn. Ví dụ, bạn có thể không cần phải ghi lại tên của người chịu trách nhiệm đảm bảo yêu cầu đã được thỏa mãn và hệ thống con mà yêu cầu được phân bổ về đó. Nếu bất kỳ thông tin nào đã được ghi lại ở đâu đó, thì có lẽ trong hệ thống giám sát phát triển tổng thể, bạn không nên ghi lặp lại thông tin đó trong cơ sở dữ liệu yêu cầu.

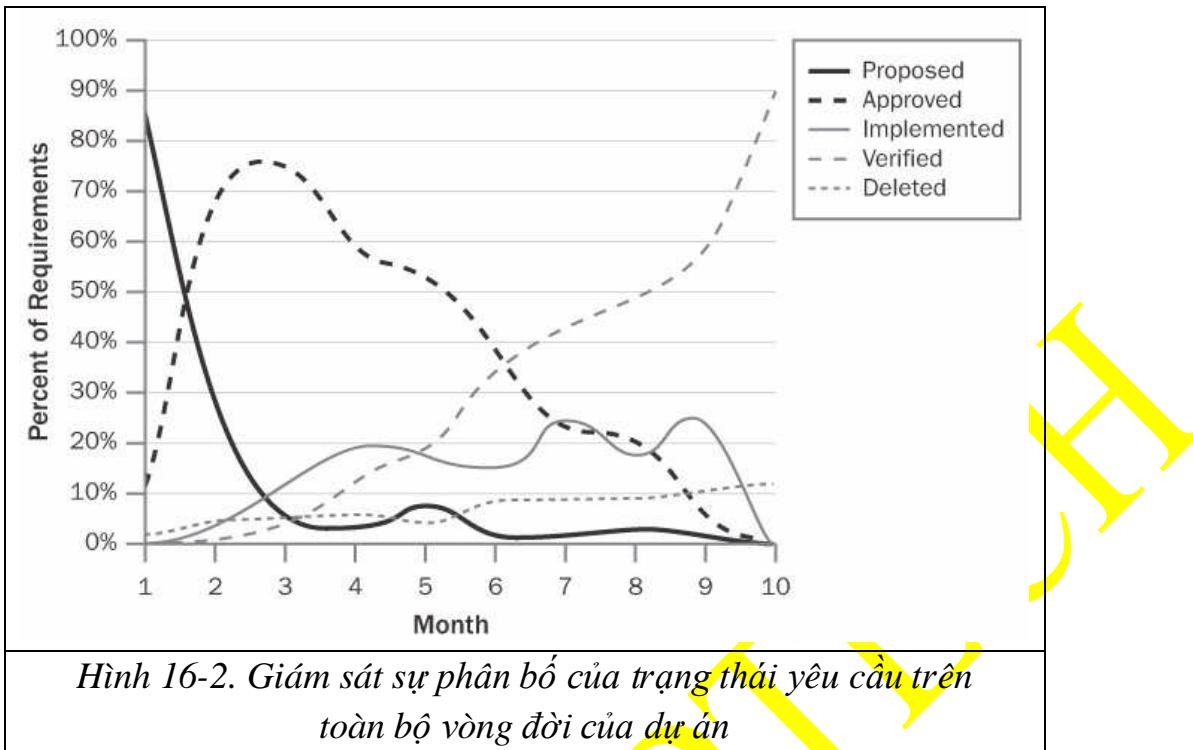
Giám sát trạng thái mỗi yêu cầu trên toàn bộ tiến trình phát triển là một khía cạnh quan trọng của việc quản lý yêu cầu (Caputo 1998). Giám sát dự án tổng thể sẽ được cải thiện nếu bạn có thể định kỳ báo cáo lượng phần trăm của toàn bộ yêu cầu, bao nhiêu phần trăm ở trạng thái nào trong số các trạng thái đã định nghĩa. Giám sát trạng thái yêu cầu sẽ hoạt động tốt chỉ khi bạn đã tạo lập các kỳ vọng rõ ràng và định nghĩa ai là người được phép sửa đổi thông tin trạng thái và các điều kiện cần phải được đáp ứng mỗi khi có bất kỳ sự thay đổi trạng thái nào. Một công cụ có thể giúp bạn giám sát ngay mỗi khi một thay đổi trạng thái xuất hiện.

Bảng 16-1 gợi ý một số trạng thái của yêu cầu.

BẢNG 16-1. CÁC TRẠNG THÁI YÊU CẦU ĐƯỢC GỌI Ý	
TRẠNG THÁI	ĐỊNH NGHĨA
Được đề xuất (Proposed)	Yêu cầu đã được đề xuất bởi một nguồn có quyền cung cấp yêu cầu
Được chấp thuận (Approved)	Yêu cầu đã được phân tích, ảnh hưởng của nó trên phần còn lại của dự án đã được ước lượng (gồm chi phí và sự giao thoa của nó với các phần khác của sản phẩm), yêu cầu đã được phân bổ tới một phiên bản cụ thể của sản phẩm. Nhóm phát triển đã cam kết thực thi yêu cầu.
Được thực thi (Implemented)	Mã thực thi yêu cầu đã được thiết kế, được viết và được kiểm thử đơn vị (unit testing).
Được kiểm tra (Verified)	Yêu cầu được thực thi rồi và đã được kiểm tra theo một cách lựa chọn từ trước, ví dụ kiểm thử hoặc thanh tra. Yêu cầu đã được lắn vết tới các test cases thích hợp. Giờ đây, yêu cầu đang được cân nhắc hoàn thành.
Bị xoá bỏ (Deleted)	Một yêu cầu đã được lập kế hoạch từ trước và bây giờ bị xoá khỏi baseline. Kèm theo đó một giải thích lý do và ai ra quyết định xoá yêu cầu.

Một số người còn thêm vào trạng thái *Được thiết kế* (nghĩa là các phần tử thiết kế cho yêu cầu chức năng đã được tạo ra và được soát xét) hoặc trạng thái *Được chuyển giao* (nghĩa là phần mềm mà trong đó yêu cầu được cài đặt đã ở trong tầm tay của người dùng, tức đang được kiểm thử beta). Sẽ tốt hơn nếu chúng ta ghi chép lại các yêu cầu được đề xuất nhưng không bao giờ được chấp thuận (nghĩa là trạng thái *Bị từ chối*) do các yêu cầu đó có thể sẽ tìm được cách khác để len vào trong khi dự án đang được phát triển.

Hình 16-2 minh họa một cách giám sát các yêu cầu trong quá trình phát triển của một dự án kéo dài 10 tháng.



Hình này thể hiện lượng phần trăm của tất cả các yêu cầu với giá trị trạng thái tương ứng vào cuối mỗi tháng. ~~Giám sát sự phân bố bằng lượng phần trăm không thể hiện liệu số lượng yêu cầu trong baseline có thay đổi vượt quá thời gian thực hiện dự án hay không, nhưng nó minh họa cách bạn tiếp cận mục tiêu kiểm tra được sự hoàn thành của tất cả các yêu cầu “Được chấp thuận”.~~

Phân loại các yêu cầu thành các nhóm riêng biệt như vậy thì thực tế hơn là cố gắng giám sát lượng phần trăm hoàn thành trọn vẹn của mỗi yêu cầu. Các nhà phát triển phần mềm có thể lạc quan thái quá khi họ báo cáo lượng phần trăm các tác vụ (tasks) được hoàn thành, thường thì điều này chỉ có nghĩa là báo cáo về những công việc đã bắt đầu nhưng chưa hoàn thành. Khuynh hướng “chạy quanh” tiến độ của dự án dẫn tới tình trạng tất-cả-đều-bình-thường (all-too-common) của dự án phần mềm, hoặc các tác vụ chính được báo cáo là “90% đã hoàn thành” sau một thời gian dài thực hiện công việc. Bạn chỉ nên thay đổi trạng thái của một yêu cầu khi các điều kiện thay đổi cụ thể đã được đáp ứng. Một số thay đổi trạng thái cũng dẫn tới việc cập nhật ma trận lần lượt để chỉ ra các phần tử thiết kế, mã hóa, kiểm thử nào liên quan đến yêu cầu, Chương 18 sẽ thảo luận về vấn đề này.

V. ĐO CÁC NỖ LỰC QUẢN LÝ YÊU CẦU (MEASURING REQUIREMENTS MANAGEMENT EFFORT)

Các hoạt động quản lý yêu cầu cần phải xuất hiện như là các tác vụ chính thức trong mỗi cấu trúc phân việc (WBS) của dự án và được phân bổ nguồn lực thích hợp. Đo lường chi phí quản lý yêu cầu trong dự án hiện tại của bạn là cách tốt nhất để biết nên dành bao nhiêu tiền để quản lý yêu cầu trong các dự án sau này.

Một tổ chức không bao giờ đo lường bất cứ khía cạnh nào trong công việc của mình thì sẽ rất khó để nắm được những gì đã tiêu phí thời gian của tổ chức. Đo lường các nỗ lực quản lý dự án và phát triển yêu cầu đòi hỏi một sự thay đổi về mặt văn hóa tổ chức và kỷ luật cá nhân để xây dựng thói quen ghi chép lại các hoạt động thường ngày. Chú ý rằng các nỗ lực dành cho công việc trên thực tế không giống như những gì bạn hoạch định trên trang lịch biểu. Các tác vụ thực tế có thể đan xen nhau, hoặc chúng đòi hỏi sự tương tác với khách hàng và do đó bị trễ. Tổng nỗ lực cho một tác vụ, theo đơn vị giờ làm việc, không nhất thiết thay đổi do các yếu tố như vậy, nhưng độ dài thời gian trên thực tế thì dài hơn thời gian được lập kế hoạch.

Giám sát nỗ lực quản lý yêu cầu hiện tại cũng giúp bạn nhìn thấu đáo để đánh giá liệu nhóm của bạn có đang thực hiện các hành động đã định để quản lý yêu cầu hay không. Thất bại khi thực hiện các hoạt động quản lý yêu cầu sẽ tăng các rủi ro của dự án từ phía các thay đổi không được kiểm soát, sự biến đổi của phạm vi và các yêu cầu bị bỏ sót khi thi công. Tính toán nỗ lực dành cho các hoạt động sau là một phần của việc quản lý yêu cầu:

- Đệ trình các thay đổi yêu cầu và đệ trình các yêu cầu mới đã được chấp thuận.
- Dánh giá các thay đổi đã được chấp thuận, bao gồm cả phân tích ảnh hưởng.
- Các hoạt động của ban kiểm soát thay đổi.
- Cập nhật tài liệu yêu cầu hoặc cơ sở dữ liệu.
- Truyền thông các thay đổi của yêu cầu tới các nhóm và cá nhân bị ảnh hưởng.
- Giám sát và báo cáo trạng thái yêu cầu.
- Định nghĩa và cập nhật thông tin lần vết yêu cầu.

Quản lý yêu cầu của dự án sẽ giúp bạn đảm bảo nỗ lực bạn dành để thu thập, tài liệu hóa, phân tích yêu cầu không bị lãng phí, dự án không bị vượt quá thời hạn. Các thực hành quản lý yêu cầu hiệu quả sẽ giúp giảm bớt khoảng cách kỳ vọng bằng cách luôn thông báo cho các thành viên dự án về trạng thái hiện tại của yêu cầu trên toàn bộ quy trình phát triển.

Các bước tiếp theo

- Lựa chọn các trạng thái mà bạn muốn sử dụng để mô tả vòng đời của các yêu cầu chức năng trong dự án của bạn. Định nghĩa tình trạng hiện thời cho mỗi yêu cầu trong SRS và theo sát sự diễn biến của yêu cầu trong phần còn lại của dự án.
- Định nghĩa một sơ đồ kiểm soát phiên bản để định danh tài liệu yêu cầu của bạn. Tài liệu hóa sơ đồ này như là một phần của quy trình quản lý yêu cầu của bạn.
- Viết một mô tả quy trình về các bước mà tổ chức của bạn sẽ thực hiện để quản lý các yêu cầu của mỗi dự án. Khuyến khích các nhà phân tích soạn thảo, soát xét, làm dự án thử nghiệm, chấp thuận các hoạt động của quy trình và các sản phẩm được chuyển giao của quy trình. Hãy chắc chắn rằng các bước của quy trình mà bạn lựa chọn là có tính thực hành và thực tế, chúng giúp bạn tăng thêm giá trị của dự án.

CHƯƠNG 17

QUẢN LÝ ĐỀ XUẤT THAY ĐỔI

Trước đây khi đánh giá một quy trình phần mềm, tôi đã hỏi các thành viên nhóm dự án xem họ tích hợp như thế nào các thay đổi của yêu cầu vào sản phẩm đang phát triển. Một người nói, “Bất cứ khi nào đại diện marketing muốn thực hiện một thay đổi, tôi yêu cầu Bruce hoặc Sandy do họ luôn nói “vâng”, việc còn lại của chúng tôi là đưa ra thời hạn thực hiện thay đổi cho đại diện marketing biết.” Những gì tôi nghe không hề gây ấn tượng cho tôi như khi tôi thấy các quy trình hợp lý quản lý các thay đổi của yêu cầu. Thay đổi không được kiểm soát là nguồn gốc phổ biến của các hỗn độn, trượt lịch biểu, các vấn đề về chất lượng. Để tránh các vấn đề nghiêm trọng xuất hiện khi các yêu cầu thay đổi, tổ chức cần:

- Các thay đổi đã đề xuất đều được đánh giá cẩn thận.
- Các cá nhân thích hợp ra quyết định về các thay đổi.
- Các thay đổi được truyền thông tới tất cả những thành viên có liên quan.
- Dự án tích hợp các thay đổi yêu cầu vào sản phẩm theo một cách làm có kiểm soát.

Trừ khi các stakeholders kiểm soát các thay đổi trong khi phát triển, còn không thì họ không thật sự biết cái gì sẽ được chuyển giao, cái gì có thể dẫn tới một khoảng cách kỳ vọng (expectation gap). Càng xa thời điểm bắt đầu dự án thì sức chống lại các thay đổi càng mạnh do bây giờ các thay đổi sẽ gây ra những vấn đề nghiêm trọng. Các thay đổi cần được ghi nhận vào tài liệu yêu cầu của dự án. *Triết lý của bạn phải là các tài liệu yêu cầu mô tả chính xác những gì có trong sản phẩm được chuyển giao.* Nếu bạn không kiểm soát chặt chẽ SRS khi sản phẩm đang được phát triển thì SRS sẽ trở nên ít hữu ích và nhóm dự án sẽ hành động như thể là một tập hợp các cá nhân rời rạc.

Khi bạn phải thực hiện một thay đổi, hãy bắt đầu từ mức cao nhất của tài liệu yêu cầu bị ảnh hưởng bởi thay đổi và lan dần ảnh hưởng xuống các yêu cầu mức dưới và các yêu cầu khác có liên quan. Ví dụ, một thay đổi được đề xuất có thể ảnh hưởng đến một use case và các yêu cầu chức năng của use case đó nhưng không ảnh hưởng gì đến các yêu cầu kinh doanh cả. Một yêu cầu hệ thống mức cao được *Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com*

chỉnh sửa có thể ảnh hưởng đến nhiều yêu cầu phần mềm. Nếu bạn thực hiện các thay đổi chỉ ở mức bị ảnh hưởng thấp nhất của yêu cầu (điển hình là một yêu cầu chức năng), thì yêu cầu có thể trở nên không nhất quán với yêu cầu cấp cao hơn của nó.

I. KIỂM SOÁT SỰ BIẾN ĐỔI CỦA PHẠM VI (CONTROLLING SCOPE CREEP)

Capers Jones (1994) nhận thấy rằng sự biến đổi của phạm vi gây ra rủi ro cho 80% các dự án làm về hệ thông tin quản lý và gây ra rủi ro cho 70% các dự án phần mềm trong lĩnh vực quân sự. Sự biến đổi của yêu cầu có thể là thêm các chức năng mới, các sửa đổi lớn được đưa ra sau khi yêu cầu dự án đã được vạch ranh giới. Vấn đề không chỉ là thay đổi yêu cầu, mà còn là các thay đổi muộn sẽ ảnh hưởng to lớn đến các công việc đã được thực hiện. Nếu mỗi thay đổi đề xuất đều được chấp thuận, thì các thay đổi có thể đến từ các nhà tài trợ dự án, các thành viên, các khách hàng, và dự án sẽ chẳng bao giờ được hoàn thành – vì vậy, không phải mỗi thay đổi đề xuất đều được chấp thuận.

Một sự tiến hóa nào đó của yêu cầu là hợp lệ và không thể tránh khỏi đối với phần lớn các dự án. Các quy trình nghiệp vụ, các cơ hội thị trường, cạnh tranh và công nghệ phần mềm có thể thay đổi theo thời gian, chúng tác động đến sự phát triển một hệ thống. Lịch biểu dự án của bạn phải bao hàm một số thời gian dệm để dự phòng cho sự thay đổi của yêu cầu. Sự biến đổi không được kiểm soát của phạm vi mà không điều chỉnh tương ứng nguồn lực, lịch biểu hoặc mục tiêu chất lượng của dự án sẽ dẫn dự án tới sự thất bại trước.

Bước đầu tiên để quản lý sự biến đổi của phạm vi là ghi thành tài liệu rõ ràng về tầm nhìn, phạm vi và các giới hạn của hệ thống mới như là một phần của yêu cầu kinh doanh, theo cách như đã mô tả trong Chương 6. Dánh giá mỗi yêu cầu hoặc tính năng được đề xuất dựa trên phạm vi và tầm nhìn để xem liệu nó có thật sự thuộc về sản phẩm hay không. Các kỹ thuật suy luận yêu cầu hiệu quả nhấn mạnh rằng sự tham gia của khách hàng có thể làm giảm số lượng các yêu cầu bị thiêu ngay từ sớm, các thay đổi chỉ được thêm vào kế hoạch của nhóm dự án sau khi các cam kết đã được đưa ra và nguồn lực đã được phân bổ (Jones 1996a). Một kỹ thuật Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

hiệu quả khác để kiểm soát sự biến đổi của yêu cầu là làm nguyên mẫu, nguyên mẫu trình bày trước người dùng một cái nhìn trước (preview) về một phương án có thể nhằm giúp người dùng và nhà phát triển cùng đạt được một sự hiểu biết chung về nhu cầu của người dùng (Jones 1994).

Kỹ thuật hiệu quả nhất để kiểm soát sự biến đổi phạm vi là có thể nói không (Weinberg 1995). Phần lớn con người không thích nói không, và nhà phát triển có thể bị áp lực phải tích hợp mỗi yêu cầu được đề xuất vào dự án. Triết lý “khách hàng luôn luôn đúng” hoặc “chúng ta sẽ đạt được trọn vẹn sự hài lòng của khách hàng” chỉ tốt đẹp một cách trừu tượng, bạn sẽ phải trả giá dài dài nếu làm theo triết lý này. Lờ đi cái giá phải trả cũng không làm mất được sự thật: thay đổi không phải cứ muốn là được. Tôi biết một công ty phát triển thương mại thành công nói mà vị giám đốc điều hành đã quen gọi ý một tính năng mới và nghe nhà quản lý phát triển nói “không phải bây giờ”. “Không phải bây giờ” thì dễ chấp nhận hơn là sự từ chối do nó để ngỏ lời hứa sẽ thêm tính năng này vào phiên bản sau.Thêm mỗi tính năng được đề xuất bởi một khách hàng, bộ phận marketing, cấp quản lý, hoặc một nhà phát triển có thể dẫn tới các cam kết bị lỡ hẹn, chất lượng bị giảm sút, nhà phát triển mệt nhoài. Ngay cả khi khách hàng không phải lúc nào cũng đúng thì có thể họ cũng có lý ở một góc độ khác, vì vậy bạn hãy lưu giữ ý kiến của họ lại để có một đánh giá khả dĩ hơn trong các chu trình phát triển sau.

Trong một thế giới lý tưởng, bạn sẽ thu thập được tất cả yêu cầu của một hệ thống mới trước khi bắt tay xây dựng nó, và yêu cầu sẽ ổn định suốt quá trình bạn xây dựng hệ thống. Đó chính là ngũ ý đằng sau quy trình phát triển tuần tự hoặc còn gọi là quy trình “thác nước”, nhưng cách làm này hoàn toàn không thực tế chút nào. Tất nhiên, tại một thời điểm nào đó, bạn phải cố định lại các yêu cầu cho một phiên bản cụ thể. Tuy nhiên, chúng ta đã bỏ qua một thực tế là đôi khi khách hàng không chắc chắn về những gì họ cần, nhu cầu của người dùng sẽ thay đổi, và nhà phát triển muốn đáp ứng các thay đổi đó. Để đối đầu với thực tế này, bạn cần một quy trình sẽ tích hợp các thay đổi phù hợp nhất vào dự án theo một cách có kiểm soát.

II. QUY TRÌNH KIỂM SOÁT THAY ĐỔI (THE CHANGE CONTROL PROCESS)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

Một quy trình kiểm soát thay đổi được thiết kế tốt sẽ cung cấp cho các stakeholders một cơ chế hình thức để đề xuất các thay đổi đối với yêu cầu. Quy trình này cho phép các nhà quản lý dự án thực hiện các quyết định nghiệp vụ với thông tin đầy đủ nhằm cung cấp giá trị lớn nhất cho khách hàng trong khi vẫn kiểm soát được chi phí vòng đời (life cycle costs) của sản phẩm. Quy trình kiểm soát thay đổi cho phép bạn giám sát trạng thái của tất cả các thay đổi đã được đề xuất, đảm bảo không thay đổi được gợi ý (suggested changes) nào bị mất hoặc bị bỏ qua. Trước khi bạn vạch ranh giới cho một tập các yêu cầu, bạn cần tuân theo quy trình kiểm soát thay đổi đối với tất cả các thay đổi đã được đề xuất.

Một quy trình kiểm soát thay đổi không hàm ý trở thành một trở ngại để thực hiện các thay đổi. Thay vì vậy, nó tạo ra một cơ chế lọc để đảm bảo hầu hết các yêu cầu thay đổi thích hợp được đáp ứng và các ảnh hưởng xấu đến dự án được làm cho tối thiểu. Quy trình thay đổi của bạn cần phải được tài liệu hóa tốt, càng đơn giản càng tốt, và – trên tất cả – cần phải hiệu quả. Nếu quy trình thay đổi không hiệu quả, vướng víu, quá phức tạp, thì mọi người sẽ có xu hướng quay lại các cách làm cũ của họ để thực hiện các thay đổi (và có lẽ họ nên làm thế).

Kiểm soát các thay đổi yêu cầu được ràng buộc với các thực hành quản lý cấu hình khác của dự án. Quản lý các thay đổi yêu cầu tương tự với quy trình giám sát và ra quyết định về báo cáo lỗi (bugs), và các công cụ tương tự có thể hỗ trợ cả hai loại hình hoạt động này. Hãy ghi nhớ rằng một công cụ thì không phải là một quy trình. Sử dụng một công cụ giám sát để quản lý các chỉnh sửa được đề xuất đối với yêu cầu thì không có nghĩa là thay thế một thủ tục thành văn mô tả nội dung và xử lý một đề xuất thay đổi.

CHÍNH SÁCH KIỂM SOÁT THAY ĐỔI (CHANGE CONTROL POLICY)

Cấp quản lý dự án cần truyền thông rõ ràng một chính sách mô tả các kỳ vọng của họ về cách các đề xuất thay đổi yêu cầu sẽ được xử lý như thế nào. Chính sách chỉ có ý nghĩa nếu chúng thực tế, tăng giá trị cho yêu cầu theo một cách nào đó, và được làm cho có hiệu lực. Tôi nhận thấy các yếu tố sau là cần thiết đối với một chính sách kiểm soát thay đổi:

- Tất cả các thay đổi yêu cầu cần phải tuân thủ quy trình. Nếu một đề xuất thay đổi không được chấp nhận theo các bước của quy trình thì đề xuất này sẽ không được quan tâm nữa.
- Chỉ đề xuất một thay đổi thì không có gì đảm bảo rằng nó sẽ được thực thi. Ban kiểm soát thay đổi của dự án (CCB, change control board) sẽ quyết định thay đổi nào được thực thi. (Chúng ta sẽ thảo luận về CCB sau trong chương này).
- Nội dung của cơ sở dữ liệu chứa các thay đổi có thể được xem bởi tất cả các stakeholders của dự án.
- Văn bản gốc (original text) của một đề xuất thay đổi thì không được phép chỉnh sửa gì cả hoặc không được phép xóa khỏi cơ sở dữ liệu.
- Mỗi thay đổi yêu cầu được tích hợp cần được lần vết tới một đề xuất thay đổi được chấp thuận.

Tất nhiên, có những thay đổi nhỏ hầu như không ảnh hưởng gì đáng kể tới dự án, có những thay đổi lớn ảnh hưởng nghiêm trọng tới dự án. Về nguyên tắc, bạn sẽ xử lý tất cả các thay đổi bằng quy trình kiểm soát thay đổi. Về thực hành, bạn có thể lựa chọn để bỏ qua một số quyết định yêu cầu chi tiết (detail requirements decisions) trong sự suy xét của các nhà phát triển. Không thay đổi nào ảnh hưởng đến hơn một người mà bị bỏ qua quy trình kiểm soát thay đổi.

Tôi biết một dự án với hai components lớn, một là ứng dụng giao diện người dùng và một là cơ sở tri thức bên trong giao diện đó, nhưng không hề có quy trình kiểm soát thay đổi. Dự án đã phải đối mặt với nhiều vấn đề khi các nhà phát triển cơ sở tri thức chọn một số giao diện ngoài khác nhưng không thông báo sự thay đổi đó cho các nhà phát triển ứng dụng. Một dự án khác, các nhà phát triển giới thiệu chức năng mới được sửa đổi mà phần còn lại của nhóm không hề hay biết cho đến khi hệ thống được kiểm thử, do đó phải làm lại các thủ tục kiểm thử và tài liệu người dùng. Các thực hành kiểm soát thay đổi thông nhất giúp tránh các vấn đề như vậy và các tác hại liên quan khác như nhiều công việc phải làm lại, thời gian kiểm thử trước đây bị bỏ lỡ.

THỦ TỤC KIỂM SOÁT THAY ĐỔI (CHANGE CONTROL PROCEDURE)

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- | |
|--|
| <ol style="list-style-type: none"> 1. GIỚI THIỆU <ol style="list-style-type: none"> 1.1. Mục đích 1.2. Phạm vi 1.3. Các định nghĩa |
| <ol style="list-style-type: none"> 2. VAI TRÒ VÀ TRÁNH NHIỆM 3. TRẠNG THÁI CỦA ĐỀ XUẤT THAY ĐỔI (CHANGE REQUEST STATUS) 4. TIÊU CHUẨN BẮT ĐẦU (ENTRY CRITERIA) 5. CÁC NHIỆM VỤ (TASKS) <ol style="list-style-type: none"> 5.1. Tạo đề xuất thay đổi 5.2. Đánh giá đề xuất thay đổi 5.3. Ra quyết định 5.4. Truyền thông về thay đổi 6. KIỂM TRA (VERIFICATION) 7. TIÊU CHUẨN KẾT THÚC (EXIT CRITERIA) 8. BÁO CÁO TRẠNG THÁI KIỂM SOÁT THAY ĐỔI (CHANGE CONTROL STATUS REPORTING) |

PHỤ LỤC: CÁC MỤC DỮ LIỆU ĐƯỢC LƯU TRỮ (DATA ITEMS STORED)

HÌNH 17-1. Template thủ tục kiểm soát thay đổi mẫu

Hình 17-1 minh họa một template mô tả một thủ tục kiểm soát thay đổi có thể ứng dụng để kiểm soát các chỉnh sửa yêu cầu và các thay đổi dự án khác. Thảo luận sau liên quan chủ yếu đến cách thủ tục xử lý như thế nào các thay đổi yêu cầu. Tôi nhận thấy nên có 4 thành phần sau trong thủ tục và các mô tả quy trình:

- *Tiêu chuẩn bắt đầu (Entry criteria)* – các điều kiện cần được đáp ứng trước khi thực thi quy trình hay thủ tục.
- *Các nhiệm vụ (tasks)* khác nhau bao gồm trong quy trình hoặc thủ tục và vai trò (người) chịu trách nhiệm thực thi mỗi nhiệm vụ.
- *Các bước để kiểm tra (verify)* xem liệu các nhiệm vụ đã được hoàn thành đúng đắn.
- *Tiêu chuẩn kết thúc (Exit criteria)* – các điều kiện chỉ ra khi nào quy trình hoặc thủ tục được hoàn thành.

1. GIỚI THIỆU

Mục này mô tả mục đích của thủ tục và xác định phạm vi ứng dụng của thủ tục. Nếu thủ tục này kiểm soát các thay đổi chỉ trong một số bán thành phẩm (work products) nào đó, thì hãy chỉ chúng ra ở đây. Cũng chỉ ra bất cứ loại hình thay đổi nào được miễn xem xét bằng thủ tục này. Ví dụ, bạn có thể miễn xem xét các thay đổi của các bán thành phẩm tạm thời (temporary work products) được tạo ra trong tiến trình phát triển của một dự án. Mục này cũng phải định nghĩa bất cứ khái niệm nào cần thiết để hiểu phần còn lại của tài liệu.

2. VAI TRÒ VÀ TRÁNH NHIỆM (ROLES AND RESPONSIBILITIES)

Liệt kê các thành viên dự án (theo vai trò, không theo tên) tham gia vào các hoạt động kiểm soát thay đổi và mô tả trách nhiệm của họ. Bảng 17-1 gợi ý một số trong các vai trò này.

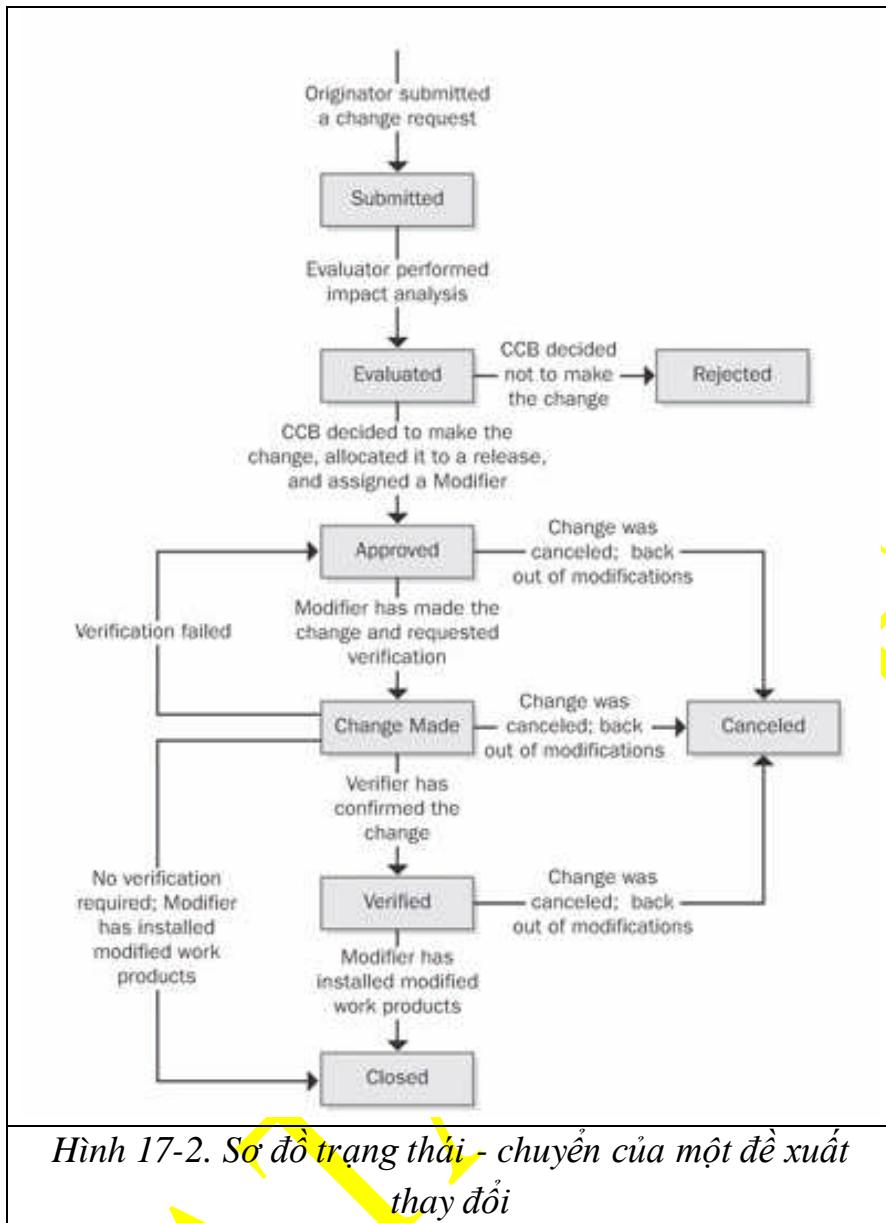
BẢNG 17-1. CÁC VAI TRÒ TRONG DỰ ÁN CÓ THỂ THAM GIA CÁC HOẠT ĐỘNG QUẢN LÝ THAY ĐỔI	
VAI TRÒ	MÔ TẢ VÀ TRÁNH NHIỆM
Trưởng CCB	Người phụ trách ban kiểm soát thay đổi, có trách nhiệm ra quyết định cuối cùng nếu CCB không đạt được sự đồng thuận
CCB	Nhóm quyết định chấp thuận hoặc từ chối các thay đổi được đề xuất trong một dự án cụ thể
Người đánh giá (Evaluator)	Người được nhà quản lý dự án yêu cầu phân tích ảnh hưởng của một thay đổi được đề xuất
Người điều chỉnh (Modifier)	Người có trách nhiệm thực hiện các thay đổi trong một bán thành phẩm (work product) nhằm đáp ứng một đề xuất thay đổi được chấp thuận; cập nhật trạng thái của đề xuất theo thời gian
Người đề xuất (Originator)	Người đề trình một đề xuất thay đổi mới
Nhà quản lý dự án	Người yêu cầu ai đóng vai trò đánh giá mỗi đề xuất thay đổi và ai là người điều chỉnh cho mỗi đề xuất thay đổi được chấp thuận

Người nhận đề xuất	Người tiếp nhận các đề xuất thay đổi mới
Người kiểm tra	Người có trách nhiệm xác định liệu sự thay đổi có được thực hiện đúng đắn hay không

Hãy làm thích ứng các vai trò và trách nhiệm trên môi trường và nhu cầu của riêng bạn, giữ các quy trình ở mức đơn giản nhất có thể nhưng vẫn hiệu quả. Một cá nhân có thể nắm giữ một vài vai trò một lúc. Với các dự án nhỏ, một vài, có thể là tất cả, vai trò được nắm giữ bởi cùng một người.

3. TRẠNG THÁI CỦA ĐỀ XUẤT THAY ĐỔI (CHANGE REQUEST STATUS)

Một đề xuất thay đổi đi qua một quy trình đã định nghĩa trước thì có những trạng thái khác nhau ở những bước khác nhau của quy trình. Bạn có thể biểu diễn các thay đổi trạng thái đó bằng cách sử dụng một sơ đồ trạng thái - chuyển (state – transition), như minh họa trong Hình 17-2. Cập nhật trạng thái của một đề xuất chỉ khi các tiêu chuẩn cụ thể được đáp ứng.



4. TIÊU CHUẨN BẮT ĐẦU (ENTRY CRITERIA)

Tiêu chuẩn bắt đầu cơ bản cho thủ tục kiểm soát thay đổi của bạn là:

- Một đề xuất thay đổi hợp lệ được nhận qua một kênh thích hợp.

Tất cả những người phát sinh (originators) đề xuất tiềm năng cần phải biết làm thế nào để trình một đề xuất thay đổi, đề xuất nên được thực hiện trên giấy hay qua một web-based form, gửi một email hay một message, hoặc sử dụng một công cụ kiểm soát thay đổi. Tất cả các đề xuất nên được gửi về một nơi duy nhất và được gán một định danh duy nhất.

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

5. CÁC NHIỆM VỤ (TASKS)

Sau khi nhận được một đề xuất thay đổi mới, bước tiếp theo là đánh giá đề xuất về tính khả thi kỹ thuật, chi phí, sự song hành với các yêu cầu kinh doanh của dự án và các ràng buộc nguồn lực khác. Trưởng CCB có thể yêu cầu người đánh giá thực hiện phân tích ảnh hưởng một cách hệ thống (được thảo luận trong chương 18), phân tích rủi ro, phân tích các nguy hiểm (hazard), hoặc các đánh giá khác. Phân tích này đảm bảo các hậu quả tiềm tàng khi chấp nhận thay đổi đều được hiểu rõ. Người đánh giá và CCB cũng cần cân nhắc các hàm ý (implications) về mặt kinh doanh và kỹ thuật nếu từ chối thay đổi.

Những người ra quyết định thích hợp trong CCB sẽ lựa chọn nên chấp thuận hay từ chối đề xuất thay đổi. CCB gán cho mỗi thay đổi được chấp thuận một mức ưu tiên hoặc ngày bắt đầu thi công, hoặc phân bổ đề xuất thay đổi vào một phiên bản cụ thể nào đó. CCB truyền thông quyết định bằng cách cập nhật trạng thái của đề xuất và thông báo tới tất cả các thành viên của dự án - những người phải thay đổi các work products như SRS, một cơ sở dữ liệu yêu cầu, các mô hình thiết kế, các components giao diện người dùng, mã nguồn, tài liệu kiểm thử, tài liệu người dùng. Những người chỉnh sửa có thể cập nhật các work products bị ảnh hưởng khi cần thiết.

6. KIỂM TRA (VERIFICATION)

Các thay đổi yêu cầu được kiểm tra bằng một cuộc thanh tra để đảm bảo SRS, tài liệu use-case, các mô hình phân tích đều được cập nhật đúng và phản ánh tất cả các khía cạnh của thay đổi. Sử dụng thông tin có khả năng lẩn vét để tìm kiếm tất cả các phần của hệ thống mà thay đổi tác động tới và vì thế cần được kiểm tra (Chương 18 sẽ thảo luận chi tiết hơn). Nhiều thành viên của nhóm cần tham gia kiểm tra các thay đổi được thực hiện dần dần xuống tới các work products bằng kiểm thử hoặc thanh tra. Sau khi kiểm tra, người điều chỉnh cài đặt (installs) các updated work products để chúng sẵn sàng cho những người còn lại của nhóm.

7. TIÊU CHUẨN KẾT THÚC (EXIT CRITERIA)

Các tiêu chuẩn kết thúc sau cần được đáp ứng để hoàn thành hợp lệ một phiên thực thi thủ tục kiểm soát thay đổi:

Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

- Trạng thái của đề xuất là Bị từ chối, Bị đóng lại, hoặc Bị loại (Rejected, Closed, Canceled).
- Tất cả các bản thành phẩm đã chỉnh sửa (modified work products) đều được cài đặt (install) vào vị trí đúng.
- Người đề xuất thay đổi, trưởng CCB, quản trị dự án, các thành viên có liên quan khác đều được thông báo về các chi tiết của thay đổi và trạng thái hiện tại của thay đổi.
- Ma trận lằn vết yêu cầu đã được cập nhật (thảo luận chi tiết trong Chương 18).

8. BÁO CÁO TRẠNG THÁI KIỂM SOÁT THAY ĐỔI (CHANGE CONTROL STATUS REPORTING)

Xác định các báo cáo và biểu đồ mà bạn sẽ sử dụng để tổng kết nội dung của cơ sở dữ liệu kiểm soát thay đổi và số lượng các đề xuất thay đổi cùng với trạng thái của mỗi đề xuất. Mô tả thủ tục sinh báo cáo. Quản trị dự án sẽ thường xuyên sinh các báo cáo đó để hỗ trợ việc giám sát tình trạng dự án.

PHỤ LỤC: CÁC MỤC DỮ LIỆU ĐƯỢC LUU TRỮ (DATA ITEMS STORED)
Bảng 17-2 liệt kê một số mục dữ liệu để cân nhắc lưu trữ mỗi đề xuất thay đổi.

MỤC DỮ LIỆU (ITEM)	MÔ TẢ (DESCRIPTION)
Nguồn gốc của thay đổi (Change origin)	Vùng chức năng cần thay đổi; các nhóm đề xuất có thể gồm marketing, cấp quản lý, khách hàng, công nghệ phần mềm, công nghệ phần cứng, kiểm thử.
Định danh của đề xuất thay đổi (Change Request ID)	Số định danh được gán cho đề xuất.
Ngày đệ trình (Date Submitted)	Ngày mà đề xuất thay đổi được đệ trình.
Ngày cập nhật (Date Updated)	Ngày mà đề xuất thay đổi được cập nhật.
Mô tả (Description)	Mô tả bằng ngôn ngữ tự nhiên thay đổi được đề xuất.

Ưu tiên thực thi (Implementation priority)	Tầm quan trọng tương đối của việc thực thi thay đổi theo quyết định của CCB: thấp, trung bình, cao.
Người điều chỉnh (Modifier)	Tên người chịu trách nhiệm chính thực thi thay đổi.
Originator (Người đề xuất)	Tên người đề trình đề xuất thay đổi này; bạn có thể ghi vào đây thông tin liên lạc của người đề xuất.
Mức ưu tiên của người đề xuất (Originator Priority)	Tầm quan trọng tương đối của việc thực thi thay đổi từ quan điểm của người đề xuất: thấp, trung bình, cao.
Phiên bản được lập kế hoạch (Planned Release)	Phiên bản sản phẩm hoặc build number mà đề xuất thay đổi đã được chấp thuận sẽ được cài đặt.
Dự án	Tên của dự án mà một thay đổi được đề xuất.
Đáp ứng (Response)	Mô tả bằng ngôn ngữ tự nhiên cách thực thi thay đổi; nhiều đáp ứng có thể bị tràn thời gian; không thay đổi các đáp ứng đã có khi nhập một đáp ứng mới.
Tình trạng (Status)	Tình trạng hiện tại của đề xuất thay đổi, được chọn từ các options trong Hình 17-2.
Tên đề xuất (Title)	Tóm tắt cỡ một dòng về thay đổi được đề xuất.
Người kiểm tra	Tên của người chịu trách nhiệm xác định liệu thay đổi có được thực thi đúng hay không.

Khi bạn định nghĩa danh sách của riêng bạn, hãy xác định mục dữ liệu nào là bắt buộc, mục nào là tùy chọn. Cũng cần chỉ ra nội dung mục nào được tự động cập nhật bởi công cụ kiểm soát thay đổi, mục nào được cập nhật bằng tay bởi một hoặc nhiều thành viên quản lý thay đổi. Bạn cũng có thể sẽ phải đánh giá lại danh sách dữ liệu khi bạn đã có kinh nghiệm hơn, vì vậy đừng tự buộc chặt mình vào một công cụ quản lý tự động cho đến khi bạn thử nghiệm thủ tục với một bảng tính đơn giản hoặc thậm chí ghi trên giấy.

CÔNG CỤ KIỂM SOÁT THAY ĐỔI

Các công cụ tự động có thể giúp bạn thực hiện quy trình kiểm soát thay đổi hiệu quả hơn. Nhiều nhóm sử dụng các công cụ thương mại để thu thập, lưu trữ, quản lý các thay đổi yêu cầu. Bạn có thể sử dụng các công cụ đó để sinh ra một danh sách. Cuốn sách này thuộc “Tủ sách Công nghệ thông tin”, tủ sách do SATA-APTECH tuyển chọn và giới thiệu. Bạn có thể xem và tải về trên www.sata-aptech.edu.vn, hoặc satablog2.wordpress.com

các đề xuất thay đổi được đệ trình để làm đầu vào lập lịch biểu cho các cuộc họp của CCB. Hãy lưu ý các đặc tính sau trong một công cụ để hỗ trợ quy trình kiểm soát thay đổi yêu cầu của bạn:

- Cho phép bạn định nghĩa các mục dữ liệu (data items) bạn muốn đưa vào một đề xuất thay đổi.
- Cho phép bạn định nghĩa một sơ đồ chuyển-trạng thái của chu trình đề xuất thay đổi.
- Ràng buộc sơ đồ chuyển - trạng thái sao cho chỉ những người được cấp quyền mới được phép thay đổi trạng thái của đề xuất.
- Ghi lại ngày tháng của mỗi thay đổi trạng thái và định danh của người thực hiện thay đổi.
- Cho phép bạn nhận các ghi chú bằng email tự động khi một người đề xuất (Originator) đệ trình một đề xuất thay đổi mới hoặc khi một trạng thái của đề xuất được cập nhật.
- Cho phép bạn sinh ra các báo cáo tiêu chuẩn hoặc được tùy biến và các biểu đồ bạn cần.

III. BAN KIỂM SOÁT THAY ĐỔI (THE CHANGE CONTROL BOARD)

Ban kiểm soát thay đổi hoặc CCB (đôi khi còn được gọi là ban kiểm soát cấu hình), được coi là một hướng dẫn thực hành tốt nhất (best practice) cho phát triển phần mềm (McConnell 1996). CCB có thể là một cá nhân hoặc là một nhóm, ra quyết định chấp thuận hay không về các thay đổi yêu cầu được đề xuất và các tính năng sản phẩm mới được gợi ý. CCB cũng ra quyết định về các khuyết (defect) đã phát hiện cần được sửa chữa và được phát hành bản sửa chữa ở phiên bản nào. Thiết lập một CCB đòi hỏi bạn xác định những người như thế nào có thể được đưa vào nhóm này và quy chế hoạt động của nhóm ra sao.

Theo nghĩa rộng nhất, một CCB soát xét và chấp thuận các thay đổi của các baselined work products thuộc một dự án, thay đổi tài liệu yêu cầu ở đây chỉ được lấy làm mẫu. Các dự án lớn có thể có nhiều mức độ kiểm soát, nơi thì chịu trách nhiệm ra các quyết định về nghiệp vụ (ví như thay đổi yêu cầu), nơi thì chịu trách nhiệm ra quyết định về kỹ thuật (Sorensen 1999). Một số CCBs được trao quyền để ra quyết định và thông báo cho cấp quản lý về quyết định đó, trong khi số khác

chỉ khuyến nghị để cấp quản lý ra quyết định. Một CCB mức cao sẽ có quyền chấp thuận các thay đổi với ảnh hưởng lớn hơn trên kế hoạch dự án so với một CCB mức thấp. Trong một dự án nhỏ, chỉ một hoặc hai người có thể ra quyết định về thay đổi mà thôi.

Đối với một số người, khái niệm “ban kiểm soát thay đổi” gợi lên hình ảnh một nơi làm mát thì giờ và chỉ hay hạch sách là giỏi. Tuy nhiên, bạn cần nghĩ khác, đó là cấu trúc giúp bạn quản lý dự án hiệu quả hơn. Cấu trúc này không phải là nơi làm tôn thì giờ của bạn.

THÀNH PHẦN CỦA CCB (CCB COMPOSITION)

Các thành viên của CCB cần đại diện tất cả các nhóm (groups) có thể bị ảnh hưởng bởi một thay đổi thuộc phạm vi mà CCB được cấp quyền xem xét. CCB có thể bao gồm đại diện từ các lĩnh vực sau:

- Cấp quản lý chương trình hoặc sản phẩm.
- Cấp quản lý dự án.
- Nhóm phát triển.
- Kiểm thử hoặc đảm bảo chất lượng.
- Marketing hoặc đại diện khách hàng.
- Người làm tài liệu người dùng.
- Người hỗ trợ kỹ thuật.
- Nhóm hỗ trợ sản phẩm (help desk).
- Nhóm quản lý cấu hình.

Một số ít cá nhân có thể đóng một số vai trò trong dự án nhỏ, các vai trò khác không cần thiết. CCB cho một dự án với hai components phần mềm và phần cứng cũng có thể bao hàm các đại diện từ công nghệ phần cứng, công nghệ hệ thống, sản xuất, và có lẽ đảm bảo chất lượng phần cứng và quản lý cấu hình. Khi bạn lập một CCB, hãy chỉ đưa vào đó một số người ít nhất mà vẫn có thể đại diện đầy đủ và có quyền ra quyết định. Các nhóm lớn có thể hay gặp khó khăn khi triệu tập họp và ra quyết định. Hãy đảm bảo rằng các thành viên CCB hiểu được trách nhiệm của họ và thực hiện chúng một cách nghiêm túc. Bạn có thể mời các cá nhân khác tới

tham dự các cuộc họp của CCB khi các đề xuất cụ thể được thảo luận để đảm bảo các thành viên của CCB có đầy đủ thông tin về kỹ thuật và nghiệp vụ.

QUY CHẾ HOẠT ĐỘNG CỦA CCB (CCB CHARTER)

Bước đầu tiên khi thiết lập CCB là viết quy chế hoạt động mô tả mục đích của CCB, phạm vi làm việc, cơ cấu thành viên, quy trình ra quyết định, các thủ tục hoạt động (Sorensen 1999). Quy chế này cần xác định mức độ thường xuyên của các cuộc họp của CCB và điều kiện để tổ chức một cuộc họp đặc biệt. Phạm vi làm việc của CCB sẽ xác định các quyết định mà họ có quyền ban hành, quyết định nào cần phải đưa lên cấp cao hơn để phân giải.

1. Ra quyết định

Mô tả quy trình ra quyết định cần xác định:

- Số lượng các thành viên CCB tối thiểu cần thiết để ra quyết định tại một cuộc họp hoặc các vai trò cần đại diện để cuộc họp có thể diễn ra.
- Biểu quyết, tìm kiếm sự đồng thuận hoặc một cơ chế khác được sử dụng để ra quyết định.
- Khi nào thì Trưởng CCB được phép bác bỏ quyết định đã được đa số các thành viên của CCB thông qua, khi nào không.

CCB phải ra quyết định căn cứ trên sự cân bằng giữa lợi ích dự đoán (anticipated benefit) và ảnh hưởng ước lượng (estimated impact) của việc chấp nhận thay đổi được đề xuất. Các lợi ích đến từ việc cải tiến sản phẩm gồm tiết kiệm tiền bạc hoặc thu nhập tăng thêm, sự hài lòng của khách hàng được đáp ứng tốt hơn, lợi thế cạnh tranh, rút ngắn thời gian ra thị trường của sản phẩm. Ảnh hưởng chỉ ra các hiệu ứng ngược nếu chấp nhận thay đổi được đề xuất có đối với sản phẩm hoặc dự án. Các ảnh hưởng đó có thể gồm tăng chi phí hoặc thời gian phát triển, làm chậm tiến độ giao hàng, hạ thấp chất lượng sản phẩm, giảm bớt các chức năng, làm mất đi sự hài lòng của khách hàng. Nếu ảnh hưởng của các chi phí hoặc lịch biểu được ước lượng vượt quá một ngưỡng ước lượng mà CCB có quyền ra quyết định thì hãy tham khảo cấp quản lý cao hơn. Ngược lại, hãy sử dụng quy trình ra quyết định của CCB để chấp nhận hoặc từ chối thay đổi được đề xuất và phân bổ mỗi thay đổi được chấp thuận tới một phiên bản cụ thể của sản phẩm.

2. Truyền thông trạng thái (Communicating Status)

Trước khi CCB ra quyết định, một cá nhân đã được chỉ định cập nhật trạng thái của đề xuất trong cơ sở dữ liệu thay đổi. Một số công cụ sẽ tự động sinh các messages bằng email để gửi trạng thái mới của đề xuất tới người đề xuất thay đổi và những người bị ảnh hưởng bởi thay đổi. Nếu email không được tự động sinh ra thì một thông báo sẽ được gửi bằng tay.

3. Tái đàm phán các cam kết (Renegotiating Commitments)

Thay đổi luôn có cái giá nào đó. Thậm chí một thay đổi bị từ chối cũng đã tiêu tốn một nguồn lực nào đó để đệ trình, đánh giá, và ra quyết định từ chối. Các tính năng mới thì có ảnh hưởng lớn hơn. Sẽ không thực tế để giả định rằng bạn có thể làm nhiều chức năng hơn với lịch biểu, nhân viên, ngân sách, các ràng buộc chất lượng không còn đủ nữa. Khi dự án chấp nhận các thay đổi yêu cầu quan trọng, thì bạn cần lập kế hoạch tái đàm phán các cam kết với cấp quản lý và khách hàng cho phù hợp với các thay đổi (Humphrey 1997). Bạn có thể đàm phán để thêm thời gian hoặc chi phí, để trì hoãn các yêu cầu chưa thực hiện về mức ưu tiên thấp hơn, hoặc để dàn xếp về chất lượng. Nếu bạn không đạt được một số điều chỉnh cam kết thì hãy ghi chép vào kế hoạch quản lý rủi ro các nguy cơ đối với thành công của dự án, để không ai ngạc nhiên nếu dự án không đạt được kết quả mong muốn.

IV. ĐO LƯỜNG HOẠT ĐỘNG THAY ĐỔI (MEASURING CHANGE ACTIVITY)

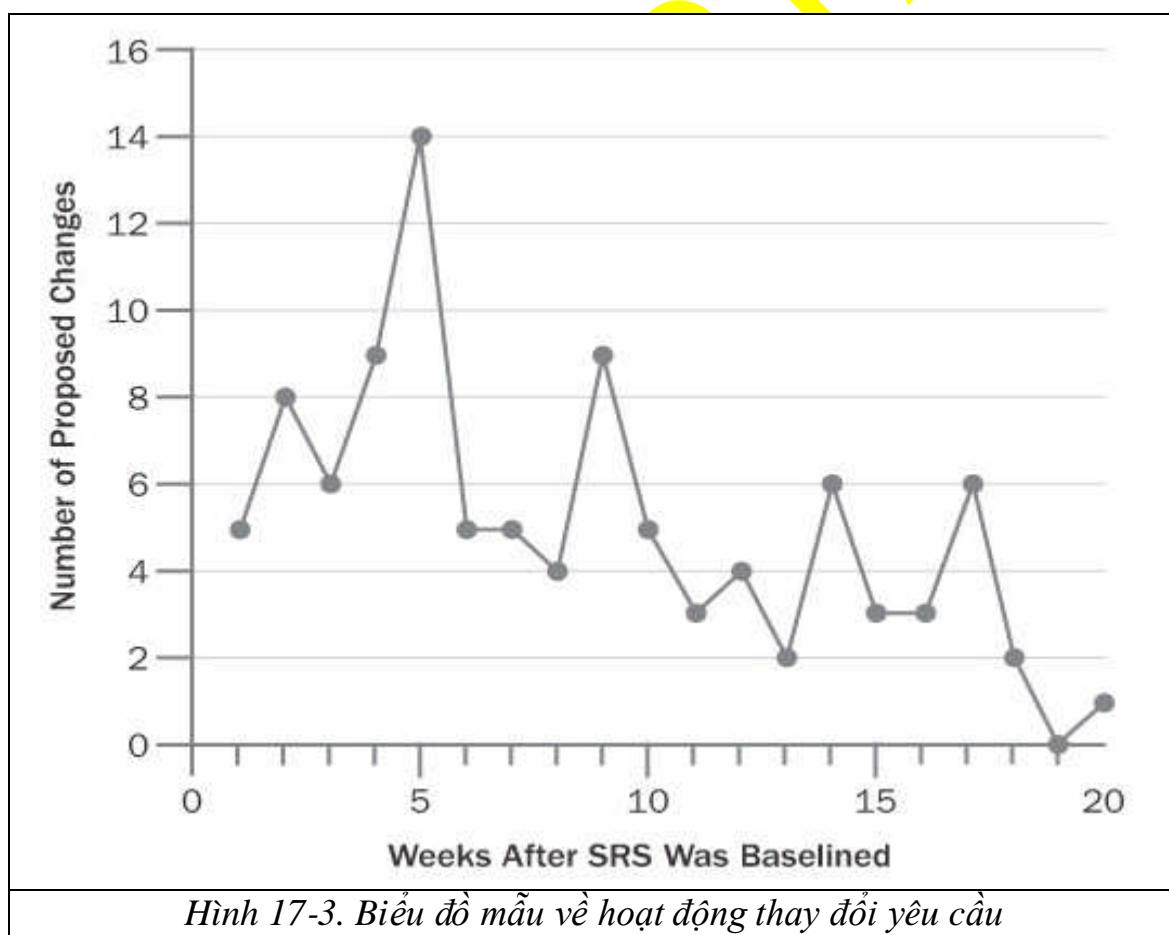
Các phép đo lường giúp bạn nhìn sâu vào bên trong dự án, sản phẩm, quy trình với cái nhìn chính xác hơn so với các cảm giác chủ quan hoặc các thông tin thu thập mơ ảo (vague recollections) về cái gì đã xảy ra trong quá khứ. Các phép đo bạn chọn cần phải thỏa mãn các câu hỏi mà bạn đặt ra hoặc các mục tiêu mà bạn cố gắng đạt được. Đo lường hoạt động thay đổi là một cách đánh giá độ ổn định của các yêu cầu và định danh cơ hội cải tiến quy trình nhằm có thể làm giảm bớt số lượng các thay đổi được đề xuất trong tương lai. Hãy cân nhắc các khía cạnh sau của hoạt động thay đổi yêu cầu của bạn (CMU/SEI 1995):

- Số lượng các đề xuất thay đổi nhận được, đề xuất thay đổi mở và đề xuất thay đổi đóng (received, open, closed).

- Số lượng lũy tiến các thay đổi cần thực hiện như thêm, xóa, chỉnh sửa các yêu cầu (bạn có thể diễn dịch nhanh ý này như lượng phần trăm của tổng số các yêu cầu trong baseline).
- Số lượng các đề xuất thay đổi xuất phát từ mỗi nguồn.
- Số lượng các thay đổi được đề xuất và thực hiện trong mỗi yêu cầu kể từ khi nó được vạch ranh giới.
- Nỗ lực tổng thể để xử lý các thay đổi.

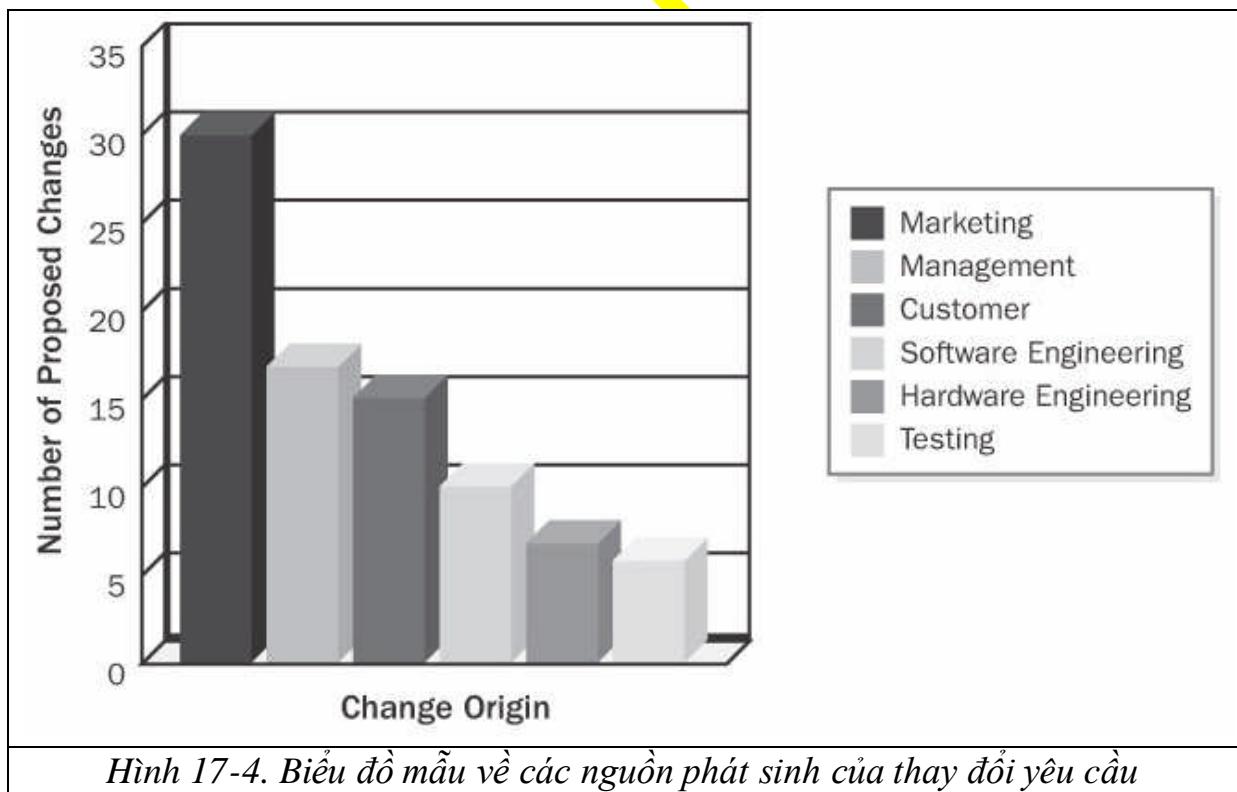
Bắt đầu với các phép đo đơn giản để khởi động văn hóa đo lường trong tổ chức của bạn và tập hợp dữ liệu chính mà bạn cần để quản lý dự án một cách hiệu quả. Khi bạn đã có kinh nghiệm, bạn có thể thực hiện các phép đo ở mức phức tạp hơn.

Hình 17-3 minh họa một cách giám sát số lượng các thay đổi yêu cầu trong một dự án.



Biểu đồ này giám sát tốc độ các đề xuất thay đổi yêu cầu mới xuất hiện. Bạn có thể giám sát số lượng các đề xuất thay đổi được chấp thuận một cách tương tự. Bạn không cần đếm các thay đổi yêu cầu được thực hiện trước khi vạch ranh giới do các yêu cầu vẫn đang tiến hóa. Tuy nhiên, một khi bạn đã vạch ranh giới cho yêu cầu, thì tất cả các thay đổi đã đề xuất cần phải tuân theo quy trình kiểm soát thay đổi, và bạn phải bắt đầu giám sát thường kỳ các thay đổi (nói cách khác, là giám sát độ bền vững của yêu cầu). Biểu đồ này có khuynh hướng tiến về zero khi càng đến gần ngày giao hàng. Nhịp độ các thay đổi thường xuyên cao thì có khả năng bạn sẽ không đáp ứng được các cam kết lịch biểu. Điều đó cũng cho thấy các yêu cầu đã được vạch ranh giới từ đầu là không đầy đủ, thực tế này gợi ý bạn có thể cải tiến các thực hành suy luận yêu cầu của mình.

Một quản lý dự án bận tâm thường xuyên đến các thay đổi yêu cầu thì có thể sẽ ảnh hưởng đến lịch giao hàng, để tránh điều đó anh ta có thể giám sát các thay đổi yêu cầu ngay từ gốc. Hình 17-4 minh họa một cách biểu diễn số lượng các đề xuất thay đổi từ các nguồn khác nhau.



Các thay đổi yêu cầu là một thực tế không thể bỏ qua trong hầu hết các dự án phần mềm. Các thực hành quản lý thay đổi yêu cầu được quy tắc hóa có thể làm giảm những hậu quả không tốt của thay đổi. Các kỹ thuật phát triển yêu cầu được cải tiến có thể làm giảm số lượng các thay đổi yêu cầu của các dự án mà bạn phải đối đầu. Các thực hành quản lý và suy luận yêu cầu hiệu quả sẽ cải tiến khả năng chuyển giao sản phẩm đúng cam kết.

Các bước tiếp theo

- Xác định những người ra quyết định trong dự án của bạn và tổ chức họ như một ban kiểm soát thay đổi. Yêu cầu CCB viết một quy chế hoạt động để chắc chắn mỗi người đều hiểu mục đích của ban, thành phần và quy trình ra quyết định.
- Định nghĩa một sơ đồ state-transition đối với chu trình sống của các thay đổi yêu cầu được đề xuất trong dự án của bạn, bắt đầu với sơ đồ trong Hình 17-2. Viết một thủ tục mô tả nhóm của bạn sẽ xử lý các thay đổi yêu cầu được đề xuất như thế nào. Sử dụng thủ tục bằng tay cho đến khi bạn tự nhận thấy thủ tục đã mang tính thực tế, hiệu quả, đơn giản hết mức có thể.
- Lựa chọn một công cụ giám sát thích hợp với môi trường làm việc của bạn và tùy biến nó để hỗ trợ thủ tục kiểm soát thay đổi mà bạn đã phát triển trước đó.

Chú thích

Cuốn sách này đã được dịch hết chương 18, chương 19 và các phụ lục sẽ được dịch và đưa lên trong phiên bản sau của bản dịch này. Xin trân trọng thông báo cùng bạn đọc.

Hà Nội, 2009