



Hướng dẫn thực hành Scrum

Quản trị dự án phần mềm theo triết lý Agile

**Scrum in Action:
Agile Software Project Management and Development**

- ◆ Cung cấp hướng dẫn từ kinh nghiệm thực tiễn cho các nhóm dự án để áp dụng dễ dàng vào những tình huống thực tế.
- ◆ Cung cấp giải pháp chi tiết áp dụng được cho bất cứ loại hình dự án phần mềm nào.
- ◆ Được viết lời nói đầu bởi Sanjiv Augustine, tác giả cuốn *Managing Agile Projects* và lời tựa bởi Dan Pilone, tác giả cuốn *Head First Software Development* và cuốn *Head First iPhone Development*

Andrew Pham và Phuong-Van Pham



NHÀ XUẤT BẢN BÁCH KHOA - HÀ NỘI



FPT Fpt University
TRƯỜNG ĐẠI HỌC FPT



 CENGAGE
Learning

HƯỚNG DẪN THỰC HÀNH SCRUM

QUẢN TRỊ DỰ ÁN PHẦN MỀM THEO TRIẾT LÝ AGILE

**SCRUM® IN ACTION: AGILE SOFTWARE
PROJECT MANAGEMENT AND
DEVELOPMENT**

ANDREW PHAM
PHUONG-VAN PHAM

Dịch thuật
Trường Đại Học FPT



NHÀ XUẤT BẢN BÁCH KHOA HÀ NỘI

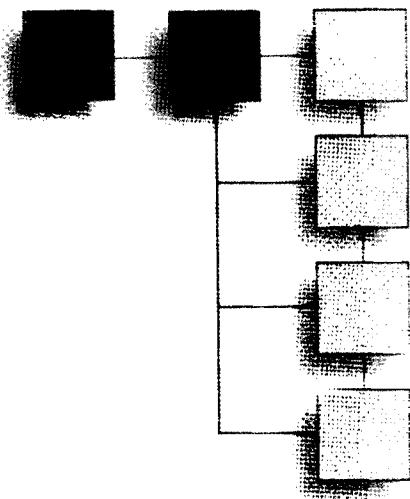
 FPT University

TRƯỜNG ĐẠI HỌC FPT

Gửi tới gia đình yêu thương của chúng tôi.

Cuốn sách này dành cho tất cả chuyên gia trên toàn thế giới - những người thông minh và chăm chỉ đã và đang làm cho thế giới này trở nên tốt đẹp hơn.

“Hành trình ngàn dặm bắt đầu từ một bước đi.”
—Lão Tử, triết gia người Trung Quốc



LỜI NÓI ĐẦU

Cuốn sách này là một tài liệu Agile rất có giá trị và mang tính thực tiễn cao. Hai tác giả Andrew và Phuong-Van đã nhanh chóng đi vào vấn đề chính bao gồm những mảng lớn về quản lý và phát triển phần mềm. Những nội dung chính bao gồm căn bản về Agile, tài chính, giành được sự hỗ trợ từ quản lý, thu thập yêu cầu trong Agile, tầm nhìn kiến trúc, vai trò của Product Owner, những kiểm thử cần chú trọng, làm việc nhóm, quản lý Agile, và cách để thích ứng với Scrum mà không phá hủy nó, và cuối cùng họ cung cấp sẵn một công cụ để đánh giá mức độ sẵn sàng.

Có hai nhóm người sẽ nhận thấy cuốn sách này rất hữu ích: một nhóm với tư duy hướng kế hoạch truyền thống và nhóm còn lại là những người tiếp cận tư duy Agile. Sức mạnh của cuốn sách bắt nguồn từ tính thực tế, không giáo điều, do đó cuốn sách đóng vai trò là cầu nối cho những suy nghĩ khác của cả hai nhóm người này.

Cuốn sách có một vài phần thông tin rất tuyệt vời dành cho những người xuất phát từ nền tảng hướng kế hoạch, truyền thống. Những nội dung về tầm nhìn kiến trúc; xây dựng những yêu cầu thông minh cho Product Backlog; trở thành một Product Owner hiệu quả trong Agile, lãnh đạo kiểu phục vụ (đầy tớ); tập trung vào kiểm thử tự động, kiểm thử hồi quy, và kiểm thử tích hợp; trở thành thành viên trong nhóm đam mê và hiệu quả; và thích ứng Scrum mà không triệt tiêu những giá trị Agile cốt lõi sẽ đáp ứng được cả hai mục đích. Một mặt, cuốn sách mang đến sự ấm áp và thoải mái nếu trái tim bạn đã bị ướp lạnh bởi những lời đồn về việc thiếu vắng các nhóm Agile trong những lĩnh vực phát triển phần mềm quan trọng. Mặt khác, cuốn sách sẽ trình bày một số công cụ “mềm” thực sự quan trọng đó là: con người, nhóm và các nhân tố lãnh đạo tuyệt vời của Scrum đã khiến cho nó trở nên phô biến về vị trí, thứ hạng và cũng như khai sáng các nhà quản lý.

Đối với những người tiếp cận tư duy Agile, đặc biệt là khi bạn không bị giới hạn nhiều bởi sự chặt chẽ mang tính truyền thống, tài liệu này đề cập về giá trị thu được nói riêng và về tài chính nói chung, về việc tạo ra tầm nhìn kiến trúc và áp dụng tầm nhìn này vào Product Backlog, cũng như việc tiến hành dự toán dựa trên các tiêu chí cơ

bản như một phương pháp bổ sung cho planning poker (bộ bài lập kế hoạch) sẽ giúp mở rộng suy nghĩ của bạn mà không khiến bạn nỗi cáu. Ở đây sự chật chẽ truyền thống được giải quyết khá hợp lý và phù hợp và tôi thực sự tin rằng bạn sẽ không gặp vấn đề khi tìm kiếm cách để áp dụng một vài trong số những kỹ thuật truyền thống đã được chứng minh ngay lập tức cho dự án của mình.

Nếu bạn là nhà quản lý dự án, nhà phát triển phần mềm, nhân viên kiểm thử, quản lý sản phẩm, chuyên gia phân tích nghiệp vụ, hay bất kỳ ai trong lĩnh vực phát triển phần mềm, bạn sẽ thấy cuốn sách rất hữu ích, giúp bạn hiểu mọi thứ về nhóm Scrum với tính thực tiễn cao. Đây là một tác phẩm dễ hiểu và tôi tin rằng bạn sẽ hứng thú đọc, bất kể bạn xuất phát ở đâu.

Tôi đã gặp Andrew trong một lớp học về ScrumMaster, và tôi đã nhanh chóng bị ấn tượng bởi chiều sâu kiến thức và sự chân thành trong quan điểm của anh ấy. Andrew khá thẳng thắn và khiêm tốn trong việc giảng dạy và chia sẻ những kiến thức của mình. Cùng với đồng tác giả Phuong-Van, Andrew mời bạn tham dự một chuyến hành trình thú vị về quản lý và phát triển linh hoạt. Tôi hy vọng bạn sẽ chấp nhận lời mời này.

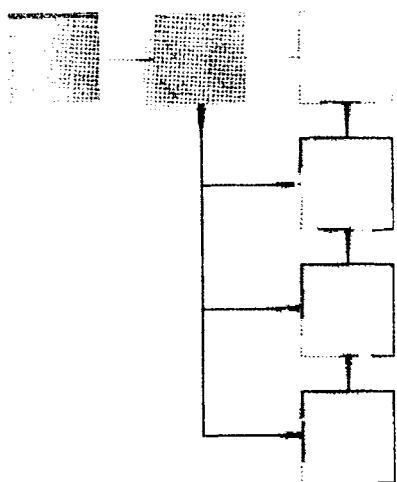
Sanjiv Augustine

Tác giả cuốn sách *Managing Agile Projects* (tạm dịch: Quản lý dự án linh hoạt)

Đã được cấp chứng chỉ đào tạo Scrum (Certified Scrum Trainer - CST)

Đồng sáng lập Agile Project Leadership Network (Mạng lưới Lãnh đạo Dự án theo Agile)

Là chủ tịch của LitheSpeed



LỜI TỰA

Agile là một quy trình phát triển phần mềm, thường bị mọi người hiểu nhầm rằng đây là một quy trình không có yêu cầu, thiết kế sẽ được tìm ra trong quá trình phát triển, và bắt cứ điều gì xa vời kiều như cuộc họp lập kế hoạch là đơn giản không cần bàn tới. Cuốn sách này rất hiệu quả để chấm dứt những quan niệm sai lầm đó.

Thông thường, những người mới thực hành phát triển theo Agile và Scrum không chỉ gặp khó khăn để nhận thấy những việc cần phải làm mà còn khó trong việc làm thế nào để biết việc họ đang làm có đúng hay không. Nếu không thực hiện các Sprint với những người có kinh nghiệm, sẽ rất khó để hiểu được sự ảnh hưởng của những thỏa hiệp xảy ra liên tục trong quá trình phát triển dự án thực tế. Liệu Sprint có quá dài? Quá ngắn? Bạn đã dành quá nhiều thời gian để thực hiện việc cải tiến chưa? Lý do đúng đắn để bạn bắt đầu làm việc này là gì? Cuốn sách của Andrew Pham và Phuong-Van Pham không những giúp giải thích lý do tại sao bạn nên thực hành Agile hoặc Scrum, mà còn cung cấp những trải nghiệm mà hai tác giả đã vất vả có được, qua đó sẽ giúp bạn biết được liệu mình có đang triển khai đúng cách hay không. Cuốn sách cung cấp một “người rơm” (straw man) rất cần thiết cho các hoạt động mà đa phần là mới mẻ đối với mọi người trong nhóm.

Một trong những phần yêu thích của tôi trong cuốn sách này đó là thời gian để nhìn nhận Agile từ quan điểm của người không phải là nhà phát triển. Với vai trò là một kiến trúc sư doanh nghiệp (enterprise architect), nếu tổ chức của bạn đã quyết định bắt đầu sử dụng Scrum cho quá trình phát triển thì điều đó có ý nghĩa gì với bạn? Với vai trò là một khách hàng, thì Scrum mang lại cho bạn điều gì? Sẽ như thế nào nếu như bạn đang chịu trách nhiệm về nỗ lực và cần phải bằng cách nào đó kết hợp quy trình Agile nhằm hỗ trợ các thông tin theo dõi về tài chính và tiến độ trong bộ phận của mình? Như phần nhiều các nhóm phát triển đều muốn nghĩ đến những thuật ngữ như ROI (tỷ lệ hoàn vốn đầu tư) và EVM (kỹ thuật quản lý giá trị thu được) để làm việc với nhà quản lý, cuốn sách này sẽ giúp liên kết một dự án lại với nhau mà không làm mất đi tính hiệu quả của Agile.

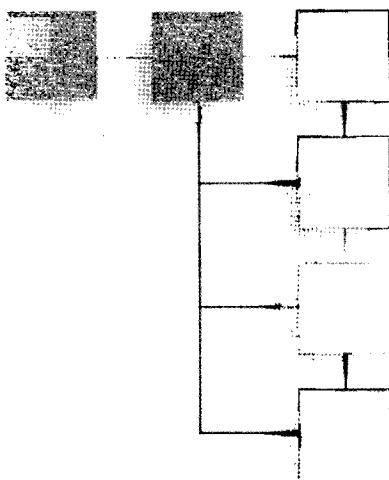
Không giống như những kiến thức đôi khi được giảng dạy trong các lớp học lý thuyết, đây là cuốn sách dành cho các chuyên gia - người muốn và cần biết cách làm thế nào để áp dụng Agile và Scrum cho tình huống thực tế. Nói điều này để thấy rằng đây không phải là một cuốn sách giáo điều về quy trình, mà là sách dành cho những chuyên gia thực dụng có sứ mệnh cung cấp phần mềm thực trong các công ty thực với những con người và tình huống thực tế.

Với tất cả những khó khăn mà chúng ta đã biết, việc triển khai các quy trình Agile và đặc biệt là Scrum trong một tổ chức là một chiến công không nhỏ. Cuốn sách này là một nguồn tài nguyên có giá trị để giúp thực hiện điều đó.

Dan Pilone

Tác giả cuốn *Head First Software Development* (tạm dịch: Nhập môn Phát triển Phần mềm) và *Head First iPhone Development* (tạm dịch: Nhập môn Phát triển trên iPhone) là người sáng lập đồng thời là Tổng Giám đốc của Element 84, LLC.

LỜI CẢM ƠN



Chúng tôi nợ rất nhiều người đã giúp chúng tôi hoàn thành cuốn sách này vô vàn lời cảm ơn và cảm tạ.

Đầu tiên là gia đình của chúng tôi, bởi tình yêu, sự hỗ trợ kiên định và vô điều kiện của họ.

Tiếp đến, chúng tôi muốn cảm ơn nhóm tại Course Technology PTR. Tôi xin cảm ơn Mitzi Koontz - biên tập thủ đắc¹ cấp cao, người luôn tin tưởng cuốn sách ngay từ ngày đầu tiên; kế đến là Sandy Doell - biên tập viên bản thảo, người đã thực hiện một công việc tuyệt vời, biến cuốn sách trở nên trôi chảy hơn rất nhiều so với bản thảo ban đầu của chúng tôi; sau cùng là Jenny Davidson - biên tập dự án, người đã bảo đảm sự cộng tác của tất cả chúng tôi để giúp mọi thứ diễn ra.

Tiếp theo, chúng tôi muốn chuyển lời cảm ơn tới:

- David K Pham, cựu Giám đốc công nghệ (CTO) của KTD Media Corporation đồng thời là người sáng lập Công ty 7billion, LLC vì đã kiểm tra và giúp cải thiện mọi ý tưởng trong cuốn sách này.
- Scott Booth, có chứng chỉ MCP, là Quản lý của Pariveda Solutions - một công ty tư vấn ở Texas, người mặc dù không tham dự buổi thuyết trình của Andrew về Ảnh hưởng của Tầm nhìn Kiến trúc tối tốc độ của nhóm và chất lượng Phần mềm ở DFW Scrum User Group, nhưng cũng đã dành khá nhiều thời gian để đánh giá cuốn sách.
- Sameer Bendre, có chứng chỉ CSM và PMP, là tư vấn cấp cao ở 3i Infotech Consulting, người đã rất nhiệt tình với cuốn sách và cũng giới thiệu một vài người bạn cho chúng tôi.

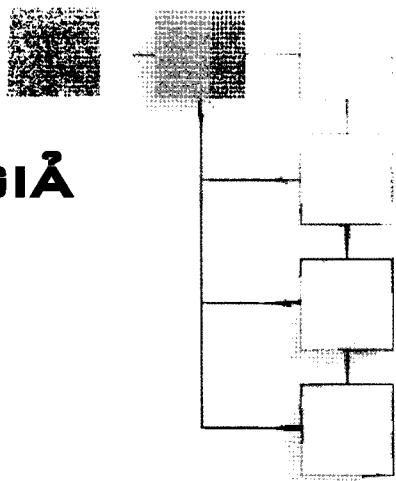
¹ Biên tập thủ đắc: Từ tiếng Anh là Acquisitions Editor - Người chịu trách nhiệm tìm kiếm và thu mua các bản thảo cho nhà xuất bản.

- Mike Vizdos, có chứng chỉ CST tại Richmond, Virginia, người đã dành thời gian để phản hồi về cuốn sách bất chấp lịch dạy dày đặc.
- Linda Rising, một tác giả với lịch làm việc bận rộn đã không ngần ngại dành thời gian để gửi một vài phản hồi nhanh về bản nháp ban đầu trên đường để bắt chuyến bay tới Đức.
- Henrik Kniberg, tác giả của cuốn *Scrum and XP in the Trenches* (tạm dịch: Scrum và XP từ những chiến hào), có chứng chỉ CST, ở Stockhold, Thụy Điển, người đầu tiên giúp chúng tôi nghĩ về một cái tên hay hơn cho cuốn sách.
- Anwar Bardai, chuyên gia máy tính cấp cao ở Công ty Compucom, người vài năm trước đã cộng tác thành công với Andrew Pham khi áp dụng Scrum và Agile để cài đặt một gói ERP trong thời gian kỷ lục cho một nhà bán lẻ máy tính lớn.
- Benjamin Oguntona, quản lý hệ thống cấp cao tại AT&T, người đã làm việc với Andrew Pham khi Andrew còn là Giám đốc kỹ thuật tại SBC, sau đó là kiến trúc sư trưởng tại Cingular Wireless, đã nhiệt tình phê bình và chỉnh sửa cuốn sách.
- Dennis Palmer, Product Owner ở Công ty Esquire Innovations, một công ty phần mềm có trụ sở ở California, người đã dành thời gian để phê bình về cuốn sách ở giai đoạn đầu.
- Dennis Simpson, chuyên gia cứu hộ công nghệ thông tin ở Dallas, Texas, người không chỉ phê bình về cuốn sách mà còn giúp chỉnh sửa để cuốn sách trở nên tinh tế hơn.
- Harold Thomas, có chứng chỉ CBAP, chuyên gia phân tích nghiệp vụ (Business Analyst - BA) ở Phòng Dịch vụ việc làm và Gia đình của bang Ohio, người không ngần ngại dành thời gian làm việc của mình để phê bình và giúp chỉnh sửa cuốn sách.
- Sanjiv Augustine, có chứng chỉ CST, đồng sáng lập Agile Project Leadership Network (APLN) và là tác giả của cuốn sách *Managing Agile Projects*, người đã không chỉ dành thời gian trong lịch làm việc bận rộn của mình để phê bình mà còn rất tốt bụng khi viết lời tựa cho cuốn sách của chúng tôi.
- Dan Pilone, là tác giả của *Head First Software Development* và *Head First iPhone Development*, đồng thời là Tổng giám đốc của Công ty TNHH Element 84, vì đã dành thời gian để phê bình và viết lời tựa cho cuốn sách.

Lời cảm ơn cuối cùng nhưng không kém phần quan trọng, chúng tôi xin gửi tới tất cả những tác giả có tác phẩm trích dẫn trong cuốn sách. Những người mà chúng tôi chưa kể đến, làm ơn hãy hiểu rằng chúng tôi đã làm tất cả để lưu lại mọi quyền sở hữu, nhưng nếu có bất cứ điều gì vô tình bị bỏ qua, xin hãy cho chúng tôi hoặc nhà xuất bản biết để thực hiện mọi chỉnh sửa cần thiết một cách sớm nhất có thể.

Andrew Pham
Phuong-Van Pham

GIỚI THIỆU TÁC GIẢ



Andrew Pham đã được cấp chứng chỉ ScrumMaster (CSM), Scrum Product Owner (CSPO) và Scrum Professional (CSP).

Ngoài ra, ông còn có chứng chỉ PMP (Project Management Professional), Kiến trúc sư Công nghệ Java (Architect for Java Technologies) của Sun Microsystems và OOAD UML (Thiết kế và phân tích hướng đối tượng UML) của IBM.

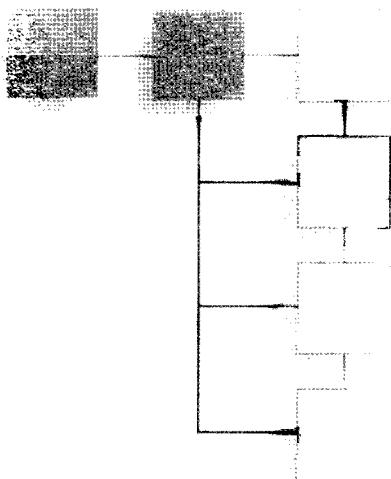
Với tư cách thành viên cấp cao của IEEE và thành viên của PMI, Andrew Pham đã giữ những vị trí cao nhất trong quản lý dự án, kiến trúc sư doanh nghiệp và phát triển phần mềm ở các công ty khởi nghiệp, tầm trung hoặc nhiều tỷ đô. Là một nhà huấn luyện Agile và Lean giàu kinh nghiệm, Andrew đã giúp nhiều công ty ứng dụng thành công Agile (Scrum) và Lean (kanban) vào các dự án thực tiễn cũng như dạy nhiều nhóm phát triển ở Hoa Kỳ và cả nước khác.

Với tố chất của một doanh nhân, Andrew Pham cũng là Chủ tịch của Công ty Agile Enterprise Consulting, LLC, chuyên cung cấp dịch vụ tư vấn, đào tạo và phát triển phần mềm. Bạn có thể liên hệ với ông qua địa chỉ thư điện tử andrew@agileenterpriseconsulting.com.

Phuong-Van Pham hiện là quản lý dự án trong một công ty nhiều tỷ đô. Phuong-Van cũng có chứng chỉ PMP, ScrumMaster (CSM), Scrum Practitioner (CSP) và Project Manager for Technology (Project+) của CompTIA.

Ngoài trách nhiệm công việc của mình, Phuong-Van cũng là thành viên tích cực của Association of University Women (AUA), Project Management Institute (PMI), PMI New York Chapter, New York Scrum User Group, Agile Project Leadership Network (APLN), và Women in Project Management Special Interest Group (SIG).

MỤC LỤC



Giới thiệu.....	xv
Lời khen tặng.....	xxix
Chương 1 Chuẩn bị cho Agile và Scrum.....	1
Nền tảng của Phát triển phần mềm và	
Quản lý dự án linh hoạt là gì?	2
Nguồn gốc của Scrum	4
Scrum hoạt động như thế nào	5
Tại sao Agile và Scrum mang lại hiệu quả trong	
quản lý dự án phần mềm?.....	10
Tóm tắt	13
Chương 2 Bàn luận về tài chính	15
Tính chi phí dự án	15
Chọn dự án đầu tư	16
Thời gian hoàn vốn	16
Mua so với Xây dựng	17
Giá trị Hiện tại Thuần (NPV)	18
Tỷ lệ Hoàn vốn nội bộ (Internal Rate of Return - IRR),	
hoặc Tỷ lệ Hoàn vốn đầu tư (Return on Investment - ROI)	19
Theo dõi hiệu suất dự án.....	20
Chi phí thực hiện.....	21
Tiến độ thực hiện	22
Dự toán ngân sách Dự án	23
Tóm tắt	25

Chương 3	Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung.....	27
	Làm việc với Quản lý Kinh doanh cấp cao	28
	Làm việc với Quản lý CNTT cấp cao.....	31
	Phòng Quản lý Chương trình.....	31
	Làm việc với Quản lý CNTT cấp trung	32
	Đảm bảo chất lượng	34
	Quản lý vận hành.....	34
	Kiến trúc doanh nghiệp (EA).....	35
	Biến người Quản lý trực tiếp trở thành Đồng minh.....	38
	Tóm tắt	39
Chương 4	Thu thập các yêu cầu trực quan cho Product Backlog.....	41
	Quy trình thu thập trực quan các yêu cầu cho Agile và Scrum	41
	Bước một: Xác định những bên liên quan và mục tiêu của họ	41
	Quy tắc SMART	42
	Bước hai: Thu thập yêu cầu cho Product Backlog.....	43
	Quy tắc CUTFIT	45
	Một ví dụ.....	48
	Tóm tắt	53
Chương 5	Tạo ra ước tính điểm story so sánh được để triển khai Scrum trong toàn doanh nghiệp.....	55
	Vấn đề với điểm story không so sánh được.....	55
	Các vấn đề về Văn hóa với Planning Poker.....	56
	Quy trình U'ớc tính dựa theo tiêu chí khách quan.....	56
	Ví dụ.....	63
	Tóm tắt	65
Chương 6	Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm.....	67
	Tầm quan trọng của tầm nhìn kiến trúc.....	69
	Làm thế nào để xác định tầm nhìn kiến trúc.....	70
	Lợi ích khác của việc có tầm nhìn kiến trúc	73
	Tóm tắt	83

Mục lục

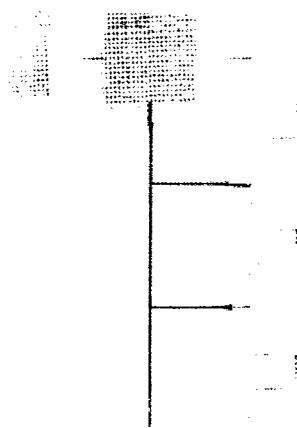
Chương 7	Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành, Lập kế hoạch Sprint và Phát triển phần mềm song song.....	85
	Từ Tầm nhìn kiến trúc đến lập kế hoạch phát hành và Lập kế hoạch Sprint.....	85
	Từ Phát triển phần mềm tăng trưởng đến Phát triển phần mềm song song	95
	Tóm tắt	98
Chương 8	Product Owner	99
	Quản lý kỳ vọng của các bên liên quan và độ ưu tiên	101
	Có tầm nhìn và hiểu biết rõ ràng về sản phẩm.....	101
	Biết cách thu thập yêu cầu cho Product Backlog	102
	Tự làm mình luôn sẵn sàng.....	102
	Biết cách trở thành một nhà tổ chức giỏi.....	103
	Biết cách để giao tiếp tốt hơn người bình thường	103
	Biết tất cả về lãnh đạo kiểu phục vụ.....	103
	Tóm tắt	104
Chương 9	Tầm quan trọng của kiểm thử tự động, kiểm thử Hồi quy và Kiểm thử tích hợp.....	105
	Tầm quan trọng của Định nghĩa hoàn thành	107
	Những kiểm thử quan trọng nhất	109
	Kiểm thử tự động	110
	Kiểm thử tích hợp liên tục	111
	Tổ chức cơ sở hạ tầng kiểm thử	111
	Tóm tắt	114
Chương 10	Tầm quan trọng của làm việc nhóm	115
	Cá nhân	116
	Tập hợp (Group).....	117
	Nhóm (Team).....	118
	Các loại tính cách Keirsey	118
	Năm giai đoạn của nhóm.....	120
	Các kỹ thuật để giải quyết xung đột trong nhóm	121
	Điều kiện tuyệt vời để làm việc nhóm.....	122
	Tóm tắt	124

Chương 11	Bản chất mới của quản lý và lãnh đạo trong dự án Scrum..	125
Huấn luyện để đạt hiệu quả cao nhất: Mô hình GROW	130	
Đặc điểm của một nhà Lãnh đạo và Quản lý chu đáo.....	132	
Tóm tắt	134	
Chương 12	Làm thế nào để thích ứng với Scrum mà không phá bỏ tính linh hoạt cơ bản hoặc triển khai ScrumBut tiêu cực...	135
Làm thế nào để thích ứng Scrum mà không triển khai "ScrumBut" tiêu cực với những ngụy biện.....	136	
Ví dụ về các tình huống thích ứng Scrum	137	
Khía cạnh Tổ chức.....	137	
Khía cạnh Hạ tầng	140	
Khía cạnh Nhóm	141	
Khía cạnh Công nghệ	142	
Khía cạnh Quy trình	142	
Khía cạnh Nghiệp vụ.....	143	
Tóm tắt	144	
Chương 13	Tự đánh giá độ sẵn sàng của Dự án Scrum	145
Một công cụ đơn giản để đánh giá độ sẵn sàng với Scrum	145	
Khía cạnh Tổ chức.....	147	
Khía cạnh Hạ tầng	148	
Khía cạnh Nhóm	148	
Khía cạnh Công nghệ	148	
Khía cạnh Quy trình	149	
Khía cạnh Nghiệp vụ.....	149	
Ví dụ	151	
Diễn đạt lại các đánh giá	156	
Tóm tắt	157	
Chương 14	Khi nào bạn cần một ScrumMaster?	159
Kiến thức chuyên sâu về lý thuyết và thực hành Scrum	160	
Tổ chất lãnh đạo kiểu phục vụ tuyệt vời.....	161	
Kỹ năng tốt về mặt Tổ chức	161	
Kỹ năng tốt về Giao tiếp	161	
Kỹ năng Thuyết trình tuyệt vời	161	
Kỹ năng Giải quyết xung đột	162	
Kỹ năng tốt về Phát triển Nhân lực.....	162	
Tóm tắt	162	

Mục lục

Chương 15	Nhắn gửi bạn đọc	163
Phụ lục A	Tìm hiểu hai case study về phát triển sản phẩm phần mềm...167	
	Giới thiệu	167
	Ruby và Ruby on Rails (RoR)	167
	Ngôn ngữ Ruby.....	167
	Ruby on Rails (RoR), khung làm việc Web.....	171
	Quản lý phiên bản và kiểm thử đối với việc phát triển Web bằng RoR	174
	Hệ thống Quản lý phiên bản Git.....	174
	Kiểm thử và Khung làm việc kiểm thử	174
	Case study 1 (Noshster).....	176
	Tầm nhìn và Mục tiêu của sản phẩm	176
	Thu thập yêu cầu bằng cách sử dụng kỹ thuật trực quan trong cuốn sách	177
	Tầm nhìn Kiến trúc và Lập kế hoạch phát hành/ kế hoạch Sprint	177
	Ước tính dự án sử dụng kỹ thuật dựa trên tiêu chí khách quan	182
	Phát triển Noshster	182
	Case study 2 (Conferous)	225
	Tầm nhìn và Mục tiêu của sản phẩm	225
	Thu thập yêu cầu bằng cách sử dụng kỹ thuật trực quan trong cuốn sách	225
	Tầm nhìn Kiến trúc và Lập kế hoạch phát hành/ kế hoạch Sprint	225
	Ước tính dự án sử dụng kỹ thuật dựa trên tiêu chí khách quan	227
	Phát triển Conferous	229
Phụ lục B	Bạn có thể hoặc có nên chấm dứt thường một Sprint.. 247	
	Lời giới thiệu.....	247
	Khi nào một Sprint có thể được kết thúc sớm hơn dự định?	247
	Làm thế nào để tránh dừng một Sprint sớm hơn dự kiến	248
	Làm thế nào để khởi động lại sau khi chấm dứt một Sprint sớm hơn dự định.....	249
Thuật ngữ.....	251	
Tham khảo	257	
Chỉ mục	261	

GIỚI THIỆU



NỘI DUNG CUỐN SÁCH

Có nhiều cuốn sách hay về Scrum, nhưng chúng tôi tin rằng không cuốn nào trong số đó cung cấp đầy đủ những kiến thức cần thiết mà một nhóm dự án phần mềm cần biết để có thể bắt đầu và hoàn thành một dự án phần mềm Scrum với những ràng buộc của tổ chức (ràng buộc ở đây có nghĩa là trong những công ty mà Scrum hay Agile vẫn chưa được áp dụng thành công rộng rãi ở quy mô toàn doanh nghiệp).

Để giúp đỡ các nhóm Scrum thành công, đôi khi là phải đi ngược lại với những ràng buộc của tổ chức, chúng tôi đã quyết định viết một cuốn sách thực hành về Scrum thông qua việc sử dụng những kiến thức mà chúng tôi thu được từ các “chiến hào”.

Với mục đích đó, 15 chương của cuốn sách sẽ cung cấp cho độc giả tất cả những kiến thức cần thiết mà một ScrumMaster có kinh nghiệm sẽ chỉ cho bạn. Ngoài ra, bạn cũng sẽ thấy trong phần Phụ lục A hai case study (tình huống nghiên cứu) về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công bằng cách sử dụng những kỹ thuật và lời khuyên trình bày trong cuốn sách này.

Chương 1: Chuẩn bị cho Agile và Scrum

Trọng tâm của Chương 1 là những kiến thức căn bản về Agile và đặc biệt nhấn mạnh vào Scrum - một phần của gia đình Agile. Chương 1 cũng đóng vai trò như là phần giới thiệu nhập môn về Scrum và chỉnh sửa một số thông tin không chính xác về Scrum, cũng gần giống như những gì mà Alan Shalloway và nhóm của ông đã viết ở trang 84-92 trong cuốn *Lean-Agile Software Development Achieving Enterprise Agility* (tạm dịch: Phát triển phần mềm linh hoạt-tinh gọn nhằm đạt được sự linh hoạt của doanh nghiệp). Đây là một cách để đưa bạn lên ngang tầm với chúng tôi trước khi chúng ta tiếp tục các nội dung khác.

Chương 2: Bàn luận về tài chính

Cho dù bạn có niềm đam mê với Agile và Scrum hay không thì bạn cũng nên nhớ rằng ngôn ngữ của quản lý kinh doanh là tài chính.

Trong Chương 2, bạn sẽ tìm hiểu các yếu tố cần thiết về tài chính, những kiến thức cần biết để phối hợp tốt hơn với nhà quản lý kinh doanh trong việc lựa chọn dự án, dự toán ngân sách dự án, và dự đoán lượng tiền và thời gian bạn sẽ cần để hoàn thành dự án.

Chương 3: Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung

Mặc dù việc giành được sự đồng tình của nhà quản lý kinh doanh cấp cao khi làm việc với họ là rất quan trọng, nhưng quan trọng hơn cả là việc trao đổi công việc hàng ngày với quản lý cấp trung, bởi vì quản lý cấp trung là điểm mấu chốt để thành công. Mục đích của Chương 3 là nhằm cung cấp cho bạn đủ kiến thức để giao tiếp thành công với cả hai cấp này.

Chương 4: Thu thập các yêu cầu trực quan cho Product Backlog

Một khi nhóm dự án đã được đồng ý để tiến hành dự án thì không có gì quan trọng hơn việc có được một tập các yêu cầu tốt. Chương 4 trình bày một quy trình rất đơn giản và trực quan để thu thập yêu cầu cho dự án Scrum mà bất kỳ ai cũng có thể sử dụng được.

Chương 5: Tạo ra ước tính điểm story so sánh được để triển khai Scrum trong toàn doanh nghiệp

Với việc Agile và Scrum ngày càng được phổ biến rộng rãi, một trong những vấn đề gặp phải với hệ thống điểm User Story hiện tại (dựa vào đó để xây dựng tốc độ nhóm) là nó không cho phép so sánh giữa các nhóm với nhau. Điều này gây cản trở cho nhiều phòng CNTT mong muốn triển khai Scrum trên diện rộng, đặc biệt là khi nói đến việc phân bổ nguồn lực và tái phân bổ. Cũng như ở Chương 4, chương này trình bày một cách tiếp cận đã mang lại thành công cho nhiều dự án. Cách tiếp cận này sẽ giúp bạn ước tính các User Story và hỗ trợ cho những điều bạn nói, không phải bằng những cảm giác cá nhân mà bằng dữ liệu hữu hình. Bên cạnh đó, cách tiếp cận này cũng tạo ra khả năng so sánh tốc độ giữa các nhóm khác nhau bên trong cùng một tổ chức.

Chương 6: Ảnh hưởng của tầm nhìn kiến trúc đối với tốc độ nhóm và chất lượng phần mềm

Đối với bất kỳ ai đã làm việc với Scrum, cho dù chỉ trong một dự án duy nhất, cũng có thể nhận thấy rằng khi tốc độ của nhóm thay đổi tăng và giảm, thường dẫn đến việc làm giảm năng suất và khả năng chuyển giao. Chương này sẽ chỉ ra cho bạn những nguyên nhân dẫn đến sự biến động của tốc độ nhóm; đồng thời cũng gợi ý cách làm thế nào để sử dụng tầm nhìn kiến trúc, hoặc như một vài người còn gọi là ý đồ kiến trúc, để khắc phục việc đó.

Chương 7: Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành, lập kế hoạch Sprint và đến phát triển phần mềm song song

Ngoài việc trợ giúp, thậm chí làm tăng tốc độ của nhóm, thì một tầm nhìn kiến trúc tốt còn có những lợi ích khác nữa. Điều này có thể bao gồm cả những ảnh hưởng tích cực đến kế hoạch phát hành và kế hoạch Sprint.

Chương 8: Product Owner

Trong một dự án Scrum, tất cả mọi người đều quan trọng, nhưng Product Owner là người có thể giúp nhóm chuyển giao giá trị kinh doanh cao nhất. Chương 8 sẽ đánh giá những phẩm chất cá nhân và chuyên môn mà một Product Owner cần phải có để thành công.

Chương 9: Tầm quan trọng của kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp

Không phải là tất cả các kiểm thử đều được tạo ra như nhau. Trong chương này, chúng tôi sẽ cung cấp một cách xử lý chuyên sâu của một vài kiểm thử mà theo chúng tôi là quan trọng nhất đối với các dự án Scrum, đồng thời giải thích tại sao chúng lại là chìa khóa thành công đối với nhóm dự án Scrum.

Chương 10: Tầm quan trọng của làm việc nhóm

Tất cả chúng ta đều đã từng nghe nói đến làm việc nhóm và tầm quan trọng của nó. Cho dù nghe có vẻ sáo rỗng, Chương 10 khẳng định rằng làm việc nhóm là điều cần thiết để các nhóm Scrum chuyển giao giá trị, đặc biệt khi nhóm Scrum là nhóm tự tổ chức. Ngoài việc nói về tầm quan trọng của làm việc nhóm, chương này cũng cung cấp một số hiểu biết về các loại tâm lý và tính khí con người, giúp cho các đồng nghiệp hiểu nhau hơn và cộng tác với nhau tốt hơn. Ngoài ra, chương này cũng cung cấp một số kỹ thuật để giải quyết xung đột, có tính đến giai đoạn trong một dự án mà tại đó xung đột xảy ra.

Chương 11: Bản chất mới của quản lý và lãnh đạo trong dự án Scrum

Mặc dù các nhóm Scrum là tự tổ chức, song việc quản lý dự án và lãnh đạo nhóm vẫn là cần thiết. Những cách thức khiến việc quản lý dự án có đôi chút biến đổi trong môi trường Scrum sẽ được tìm hiểu trong chương này. Điểm mấu chốt cần ghi nhớ ở đây là lãnh đạo kiểu phục vụ sẽ thay thế kiểu mệnh lệnh và kiểm soát trong quá khứ. Chương 11 điểm lại một vài kỹ thuật hữu ích cho ScrumMaster và Product Owner để huấn luyện và lãnh đạo khi họ thành lập nhóm, nhằm giúp hướng dẫn nhóm đi đến việc chuyển giao sản phẩm cuối cùng của dự án.

Chương 12: Làm thế nào để thích ứng với Scrum mà không phá bỏ tính linh hoạt cơ bản hoặc triển khai ScrumBut tiêu cực

Có tuyệt vời không nếu ai đó sáng tạo ra một phương pháp luận hoặc quy trình có thể phù hợp với mọi công ty và mọi vấn đề và chúng ta có thể dùng mà không cần thích nghi nó với môi trường của mình? Thực tế là chúng ta thường phải thích nghi các phương pháp với những ràng buộc của mình để chúng hoạt động. Điều này cũng đúng với Scrum.

Không giống như các “ScrumBut” tiêu cực (chính là việc ứng dụng sai Scrum), ở đây chúng ta sẽ xem xét một số ví dụ về “ScrumBut” tích cực (hoặc về việc thích nghi tốt của Scrum) giống như phương pháp mà Jurgen Appelo, CIO tại CSM eCompany ở Netherlands đã làm trong cuốn “ScrumButs are the best part of Scrum” (tạm dịch: ScrumBut là phần tốt nhất của Scrum).

Chương 13: Tự đánh giá độ sẵn sàng của dự án Scrum

Chương 13 cung cấp một ví dụ về việc tự đánh giá độ sẵn sàng của dự án Scrum, điều mà bạn cần phải tiến hành ở thời điểm ban đầu của dự án Scrum. Việc này sẽ cho phép bạn xác định những khía cạnh mà các trở ngại có thể gây tổn hại đến khả năng chuyển giao của nhóm dự án.

Tùy vào điểm số có được từ môi trường hiện tại, bạn biết được nhóm của mình sẽ gặp thuận lợi hay khó khăn khi làm việc trong môi trường đó và cố gắng cải thiện để có thể chuyển giao dễ dàng hơn.

Chương 14: Khi nào bạn cần một ScrumMaster

Trừ khi bạn hoặc nhóm của bạn đã có nhiều kinh nghiệm với Scrum, nếu không bạn sẽ cần đến một ScrumMaster để hướng dẫn bạn vượt qua một số khó khăn xuất hiện trong nỗ lực đầu tiên với Scrum của bạn.

Mặc dù cuốn sách này đóng vai trò như một ScrumMaster hướng dẫn bạn, Chương 14 sẽ nêu một số phẩm chất cá nhân và chuyên môn mà một ScrumMaster phải có để thành công.

Chương 15: Nhắn gửi bạn đọc

Thay vì để bạn bước đi một mình, Chương 15 cung cấp một số gợi ý về những ứng dụng chính có thể rút ra từ các chương và trật tự mà bạn cần phải tuân thủ khi áp dụng chúng vào trong các tình huống dự án.

Phụ lục A: Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Phụ lục A trình bày hai sản phẩm phần mềm đã được xây dựng và triển khai thành công bằng cách sử dụng những lời khuyên trong cuốn sách này, từ việc thu thập yêu cầu cho đến tầm nhìn kiến trúc đến việc lập kế hoạch phát hành, kế hoạch Sprint và kiểm thử.

Case study đầu tiên cung cấp một ví dụ về một ứng dụng được phát triển theo chiều dọc, trong khi case study thứ hai cung cấp ví dụ về một ứng dụng được phát triển theo chiều ngang.

Phụ lục B: Bạn có thể hoặc có nên chấm dứt bất thường một Sprint ?

Thông thường, dự án Scrum sẽ diễn ra suôn sẻ nếu như bạn đã làm theo những lời khuyên được trình bày trong cuốn sách này. Nhưng thật đáng buồn, có những lúc mà bạn phải kết thúc bất thường một Sprint. Điều này đôi khi xảy ra do việc đánh giá thấp tốc độ của nhóm hoặc do sự phức tạp của một vài User Story.

Nhằm cung cấp cho bạn những tư vấn cần thiết trong tình huống này, cuốn sách cũng kèm theo một chương nhỏ về sự kết thúc bất thường của Sprint, những nguyên nhân là gì, cách để né tránh hoặc đối phó với tình huống này, và làm thế nào để tái khởi động lại công việc của nhóm dự án sau khi tình huống xảy ra.

Thuật ngữ

Một bảng chú giải được cung cấp ở cuối cuốn sách nhằm giúp độc giả hiểu một số thuật ngữ quan trọng của Scrum.

NHỮNG ĐỐI TƯỢNG NÊN ĐỌC CUỐN SÁCH NÀY

Sau đây là gợi ý những chương bạn nên đọc và thứ tự đọc chúng tùy theo chức danh và vai trò của bạn trong tổ chức. Có hai nhóm người sẽ thấy cuốn sách này có ích: Những người mới với Scrum và (2) những người đã có chút kinh nghiệm với Scrum.

Giới thiệu

Nếu Scrum còn mới với bạn

Nếu bạn là thành viên của đội quản lý

Là một thành viên của bộ phận quản lý, đầu tiên bạn nên đọc Chương 1 để làm quen với sự phát triển của Agile, cách để bắt đầu và nguồn gốc cũng như những điều cần bản về Scrum. Kế tiếp, bạn nên đọc Chương 3 để xem những lời khuyên dành cho nhóm Scrum khi làm việc với các nhà quản lý, cả quản lý kinh doanh cấp cao cũng như quản lý cấp trung. Sau đó, bạn nên đọc Chương 8 và Chương 14 để tìm hiểu vai trò và phẩm chất của Product Owner và ScrumMaster.

Dù có dùng hay không dùng Scrum, chúng tôi đoán rằng bạn sẽ thắc mắc về cách bạn hoặc nhóm của bạn sẽ thu thập yêu cầu người dùng để tạo ra Product Backlog - danh sách mà bạn đã nghe tới rất nhiều. Trong trường hợp này, bạn nên đọc Chương 4 để biết cách xác định mục tiêu của người dùng và sử dụng kỹ thuật trực quan để thu thập yêu cầu cho dự án Scrum. Có thể bạn cũng đã nghe thấy rằng khi dùng planning poker (bộ bài lập kế hoạch) thì tốc độ của nhóm khác nhau là khác nhau. Tốc độ nhóm là số lượng User Story mà nhóm Scrum có thể chuyển giao trong mỗi Sprint. Nếu đó là những gì mà bạn đã nghe thấy và nếu bạn thấy băn khoăn về cách so sánh tốc độ của các nhóm khác nhau khi triển khai Scrum ở diện rộng, thi bạn có thể muốn đọc Chương 5. Chương này giới thiệu một cách đơn giản để ước tính các yêu cầu, cách làm này hỗ trợ mọi thành viên nhóm trong việc ước tính và làm cho tốc độ nhóm trở nên có thể so sánh được. Sau này chắc chắn đây sẽ là một điều kiện then chốt để triển khai Scrum thành công với nhiều nhóm trong doanh nghiệp.

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này. Bạn sẽ thấy bất ngờ về lượng kiến thức mình đã học được về Agile và Scrum.

Nếu bạn là thành viên của đội quản lý kỹ thuật

Là một thành viên của bộ phận quản lý, đầu tiên bạn nên đọc Chương 1 để làm quen với sự phát triển của Agile, cách để bắt đầu và nguồn gốc cũng như những điều cần bản của Scrum. Kế tiếp, bạn nên đọc Chương 3 để xem những lời khuyên dành cho Nhóm Scrum khi làm việc với các nhà quản lý, cả quản lý kinh doanh cấp cao cũng như quản lý cấp trung. Sau đó, bạn nên đọc Chương 8 và Chương 14 để tìm hiểu vai trò và phẩm chất của Product Owner và ScrumMaster.

Dù có hay không việc dùng Scrum, chúng tôi đoán rằng bạn sẽ thắc mắc về cách bạn hoặc nhóm của bạn sẽ thu thập yêu cầu người dùng để tạo ra Product Backlog - danh sách mà bạn đã nghe tới rất nhiều. Trong trường hợp này, bạn nên đọc Chương 4 để biết cách xác định mục tiêu của người dùng và sử dụng kỹ thuật trực quan để thu thập yêu cầu cho dự án Scrum. Có thể bạn cũng đã nghe thấy rằng khi dùng planning poker (bộ bài lập kế hoạch) thì tốc độ của nhóm khác nhau là khác nhau. Tốc độ nhóm là số lượng User Story mà nhóm Scrum có thể chuyển giao trong mỗi Sprint. Nếu đó là những gì mà bạn đã nghe thấy và nếu bạn thấy băn khoăn về cách

so sánh tốc độ của các nhóm khác nhau khi triển khai Scrum ở diện rộng, thì bạn có thể muốn đọc Chương 5. Chương này giới thiệu một cách đơn giản để ước tính các yêu cầu, cách làm này hỗ trợ mọi thành viên nhóm trong việc ước tính và làm cho tốc độ nhóm trở nên có thể so sánh được. Sau này chắc chắn đây sẽ là một điều kiện then chốt để triển khai Scrum thành công với nhiều nhóm trong doanh nghiệp.

Bởi bạn xuất phát từ nền tảng kỹ thuật, nên tiếp theo bạn nên đọc Chương 6, chương này viết về tầm nhìn kiến trúc, về sự hữu ích của tầm nhìn kiến trúc trong việc giúp nhóm duy trì một tốc độ tốt và thậm chí cả việc lập kế hoạch Sprint và lập kế hoạch phát hành cũng tốt hơn (Chương 7).

Nếu bạn muốn biết ai là người sẽ dẫn dắt các yêu cầu và nhu cầu kinh doanh trong Scrum thì Chương 8 viết về Product Owner sẽ cung cấp thông tin cho bạn. Tương tự, nếu bạn muốn biết ai sẽ là người chịu trách nhiệm giúp nhóm và Product Owner hiểu và áp dụng đúng đắn Scrum, bạn nên đọc Chương 14.

Dù bạn thuộc bộ phận quản lý và không cần bận tâm thêm nữa, nhưng có thể bạn vẫn muốn đọc Chương 9 để biết ba loại kiểm thử quan trọng nhất và cách tổ chức chúng để giúp nhóm hoặc tổ chức của bạn chuyển giao. Nếu bạn là một thành viên của đội quản lý, chúng tôi đoán rằng bạn sẽ tò mò về cách quản lý nhóm bởi bạn đã nghe thấy rằng trong Scrum nhóm là tự quản. Trong trường hợp này, bạn nên đọc Chương 10. Bởi bạn là thành viên của đội quản lý, nên rất tự nhiên chúng tôi khuyên bạn đọc Chương 11 với nội dung về quản lý dự án và lãnh đạo nhóm. Như bất kỳ một quản lý chuyên nghiệp dạn dày nào, bạn biết một thực tế là không hề dễ dàng trong việc áp dụng một quy trình hoặc khung quy trình vào tổ chức mà không có một sự thích ứng. Trong trường hợp này, Chương 12 là phần tiếp theo bạn nên đọc. Sau đó, nếu bạn băn khoăn về vị trí của mình với Scrum thì Chương 13 có những khía cạnh mà bạn cần xem xét và bắt đầu bằng việc sử dụng các câu hỏi.

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này. Bạn sẽ thấy bất ngờ về lượng kiến thức mình đã học được về Agile và Scrum.

Nếu bạn là nhà quản lý dự án

Do bạn sẽ có thể được yêu cầu đảm nhận vai trò ScrumMaster trong dự án và đồng thời giữ một vai trò nào đó về quản lý, khi đó chúng tôi có thể coi bạn là nhà quản lý dự án, người có cùng những mối quan tâm giống như các nhà quản lý khác, cho nên chúng tôi đề nghị bạn đọc toàn bộ cuốn sách này.

Bạn sẽ ngạc nhiên về lượng kiến thức đã học được về Agile và Scrum để hiểu những gì mà các chuyên gia nói và tự tin tiến về phía trước.

Nếu bạn là nhà phát triển

Bởi bạn là thành viên của đội kỹ thuật, chúng tôi cho rằng bạn chủ yếu quan tâm tới những chương hướng dẫn cách thực hiện công việc phát triển. Trong trường hợp này, bạn hãy đọc:

Giới thiệu

- **Chương 1 (Toàn bộ nội dung về Agile và Scrum)**
- **Chương 5 (Tạo ra ước tính điểm story so sánh được để triển khai Scrum trong toàn doanh nghiệp)**
- **Chương 6 (Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm)**
- **Chương 7 (Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành và lập kế hoạch Sprint)**
- **Chương 8 (Product Owner)**
- **Chương 9 (Tầm quan trọng của kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp)**
- **Chương 10 (Tầm quan trọng của làm việc nhóm)**
- **Chương 14 (Khi nào bạn cần một ScrumMaster?)**

Bây giờ nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này. Bạn sẽ ngạc nhiên về lượng kiến thức đã học được về Agile và Scrum để hiểu những gì mà các chuyên gia nói và tự tin tiến về phía trước.

Nếu bạn là chuyên viên phân tích nghiệp vụ (Business Analyst - BA)

Giống như nhà phát triển, chúng tôi giả định rằng bạn chủ yếu quan tâm tới những chương phi kỹ thuật, giúp bạn hiểu Scrum là gì và cách để thực hiện công việc. Nếu như thế bạn nên đọc:

- **Chương 1 (Toàn bộ nội dung về Agile và Scrum)**
- **Chương 2 (Bàn luận về tài chính)**
- **Chương 4 (Thu thập yêu cầu cho Product Backlog)**
- **Chương 5 (Tạo ra ước tính điểm story so sánh được để triển khai Scrum trong toàn doanh nghiệp)**
- **Chương 8 (Product Owner)**
- **Chương 10 (Tầm quan trọng của làm việc nhóm)**
- **Chương 11 (Bản chất mới của quản lý và lãnh đạo trong dự án Scrum)**
- **Chương 14 (Khi nào bạn cần một ScrumMaster?)**

Nếu bạn có nền tảng kỹ thuật hoặc thời gian thì nên đọc Chương 6, 7 để biết về tầm nhìn kiến trúc. Chúng tôi đã viết những chương này theo cách dễ hiểu cho cả những thành viên phi kỹ thuật. Bạn và những thành viên kỹ thuật trong nhóm có thể hiểu nhau về những chương này, do đó bạn có thể cộng tác với họ một cách tốt nhất.

Bây giờ nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này.

Nếu bạn là kiểm thử viên

Giống như nhà phát triển và nhân viên phân tích nghiệp vụ, chúng tôi giả định rằng bạn chủ yếu quan tâm tới những chương có thể giúp hiểu Scrum là gì và cách để thực hiện công việc.

Nếu như thế bạn nên đọc:

- Chương 1 (Toàn bộ nội dung về Agile và Scrum)
- Chương 4 (Thu thập yêu cầu cho Product Backlog)
- Chương 9 (Tầm quan trọng của kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp)
- Chương 10 (Tầm quan trọng của làm việc nhóm)
- Chương 11 (Bản chất mới của quản lý và lãnh đạo trong dự án Scrum)

Nếu bạn quan tâm tới cách những đồng nghiệp của mình thu thập yêu cầu người dùng và ước tính trong dự án Scrum thì hãy đọc Chương 4 và 5. Nếu bạn có một chút nền tảng kỹ thuật thì chúng tôi đề nghị bạn cũng nên đọc Chương 6 và 7 có chủ đề về tầm nhìn kiến trúc. Chúng tôi đã viết những chương này theo cách dễ hiểu cho cả những thành viên phi kỹ thuật. Hãy thử đọc những chương này và bạn sẽ không phải hối tiếc bởi bạn và những thành viên kỹ thuật trong nhóm có thể hiểu những chương này như nhau, từ đó bạn có thể cộng tác với họ một cách tốt nhất.

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này. Bạn sẽ ngạc nhiên về lượng kiến thức đã học được về Agile và Scrum.

Nếu bạn là chuyên gia kiến trúc ứng dụng

Giống như nhà phát triển và chuyên viên phân tích nghiệp vụ, chúng tôi giả định rằng bạn chủ yếu quan tâm tới những chương có thể giúp hiểu Scrum là gì và cách để thực hiện công việc.

Nếu như thế bạn nên đọc:

- Chương 1 (Toàn bộ nội dung về Agile và Scrum)
- Chương 4 (Thu thập yêu cầu cho Product Backlog)
- Chương 6 (Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm)
- Chương 7 (Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành và lập kế hoạch Sprint)

Giới thiệu

- **Chương 10 (Tầm quan trọng của làm việc nhóm)**
- **Chương 11 (Bản chất mới của quản lý và lãnh đạo trong dự án Scrum)**

Tiếp theo hãy đọc Chương 3, đặc biệt là phần 3.3.

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này. Bạn sẽ thấy bất ngờ về lượng kiến thức mà mình đã học được về Agile và Scrum.

Nếu bạn là kiến trúc sư doanh nghiệp

Không giống như nhà phát triển, kiểm thử viên và chuyên viên phân tích nghiệp vụ, kiến trúc sư doanh nghiệp chủ yếu quan tâm tới việc đảm bảo kiến trúc của dự án Scrum phù hợp với kiến trúc tổng thể của doanh nghiệp. Do đó, bạn hãy đọc:

- **Chương 1 (Toàn bộ nội dung về Agile và Scrum)**
- **Chương 4 (Thu thập yêu cầu cho Product Backlog)**
- **Chương 6 (Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm)**
- **Chương 7 (Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành và lập kế hoạch Sprint)**
- **Chương 8 (Product Owner)**
- **Chương 10 (Tầm quan trọng của làm việc nhóm)**
- **Chương 11 (Bản chất mới của quản lý và lãnh đạo trong dự án Scrum)**
- **Chương 14 (Khi nào bạn cần một ScrumMaster?)**

Tiếp theo hãy đọc Chương 3, đặc biệt là phần 3.3.

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này. Bạn sẽ thấy bất ngờ về lượng kiến thức mà mình đã học được về Agile và Scrum.

Nếu bạn là thành viên của PMO

Bởi nhóm của bạn quan tâm tới sự phù hợp của Công Nghệ Thông Tin (CNTT) tới nhu cầu kinh doanh và hiệu quả CNTT, nên bạn sẽ muốn đọc:

- **Chương 1 (Toàn bộ nội dung về Agile và Scrum)**
- **Chương 2 (Bàn luận về tài chính)**
- **Chương 3 (Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung)**

- Chương 4 (Thu thập yêu cầu cho Product Backlog)
- Chương 5 (Tạo ra ước tính điểm story so sánh được để triển khai Scrum cho toàn doanh nghiệp)
- Chương 6 (Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm)
- Chương 7 (Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành và lập kế hoạch Sprint)
- Chương 8 (Product Owner)
- Chương 11 (Bản chất mới của quản lý và lãnh đạo trong dự án Scrum)
- Chương 12 (Cách thích ứng với Scrum)
- Chương 13 (Tự đánh giá độ sẵn sàng của dự án Scrum)
- Chương 14 (Khi nào bạn cần một ScrumMaster?)

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này. Bạn sẽ thấy bất ngờ về lượng kiến thức mà mình đã học được về Agile và Scrum.

Nếu bạn là thành viên của nhóm vận hành

Bởi nhóm của bạn chịu trách nhiệm hàng ngày vận hành ứng dụng phần mềm cho tổ chức, nên bạn sẽ muốn đọc:

- Chương 1 (Toàn bộ nội dung về Agile và Scrum)
- Chương 3 (Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung)
- Chương 6 (Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm)
- Chương 7 (Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành và lập kế hoạch Sprint)
- Chương 8 (Product Owner)
- Chương 9 (Tầm quan trọng của kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp)
- Chương 14 (Khi nào bạn cần một ScrumMaster?)

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này. Bạn sẽ thấy bất ngờ về lượng kiến thức mà mình đã học được về Agile và Scrum.

Giới thiệu

Nếu bạn là một Product Owner

Bạn chịu trách nhiệm về vấn đề kinh doanh của dự án, do đó hãy đọc:

- Chương 8 (Product Owner)
- Chương 2 (Bàn luận về tài chính)
- Chương 3 (Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung)
- Chương 4 (Thu thập yêu cầu cho Product Backlog)
- Chương 5 (Tạo ra ước tính điểm story so sánh được để triển khai Scrum cho toàn doanh nghiệp)
- Chương 9 (Tầm quan trọng của kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp)
- Chương 11 (Bản chất mới của quản lý và lãnh đạo trong dự án Scrum)
- Chương 14 (Khi nào bạn cần một ScrumMaster?)

Nếu bạn chịu trách nhiệm đảm bảo nhóm phát triển tạo ra một ứng dụng mạnh mẽ, không bị sụp đổ sau khi đưa vào sử dụng, chúng tôi gợi ý bạn đọc:

- Chương 6 (Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm)
- Chương 7 (Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành và lập kế hoạch Sprint)

Nếu có thêm thời gian, chúng tôi gợi ý bạn đọc toàn bộ cuốn sách từ đầu tới cuối. Bạn sẽ thấy bất ngờ về lượng kiến thức mà mình đã học được về Agile và Scrum.

Nếu bạn là người tài trợ kinh doanh

Là người đứng sau nhu cầu kinh doanh của dự án và là người cấp vốn chính cho dự án, bạn nên quan tâm tới việc đọc những chương sau:

- Chương 1 (Toàn bộ nội dung về Agile và Scrum)
- Chương 2 (Bàn luận về tài chính)
- Chương 3 (Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung)
- Chương 8 (Product Owner)
- Chương 11 (Bản chất mới của quản lý và lãnh đạo trong dự án Scrum)
- Chương 14 (Khi nào bạn cần một ScrumMaster?)

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này. Bạn sẽ thấy bất ngờ về lượng kiến thức mà mình đã học được về Agile và Scrum.

Nếu bạn đã có kinh nghiệm với Scrum

Nếu bạn là ScrumMaster

Là một ScrumMaster có kinh nghiệm, bạn nên đọc:

- Chương 14 (Khi nào bạn cần một ScrumMaster?)
- Chương 2 (Bàn luận về tài chính)
- Chương 3 (Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung)
- Chương 4 (Thu thập yêu cầu cho Product Backlog)
- Chương 5 (Tạo ra ước tính điểm story so sánh được để triển khai Scrum cho toàn doanh nghiệp)
- Chương 6 (Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm)
- Chương 7 (Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành và lập kế hoạch Sprint)
- Chương 11 (Bản chất mới của quản lý và lãnh đạo trong dự án Scrum)
- Chương 12 (Cách thích ứng với Scrum)
- Chương 13 (Tự đánh giá độ sẵn sàng của dự án Scrum)

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này.

Nếu bạn là Product Owner

Là một Product Owner có kinh nghiệm, bạn nên đọc:

- Chương 8 (Product Owner)
- Chương 2 (Bàn luận về tài chính)
- Chương 4 (Thu thập yêu cầu cho Product Backlog)
- Chương 5 (Tạo ra ước tính điểm story so sánh được để triển khai Scrum cho toàn doanh nghiệp)
- Chương 6 (Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm)
- Chương 7 (Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành và lập kế hoạch Sprint)
- Chương 11 (Bản chất mới của quản lý và lãnh đạo trong dự án Scrum)

Giới thiệu

- Chương 12 (Cách thích ứng với Scrum)
- Chương 14 (Khi nào bạn cần một ScrumMaster?)

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này.

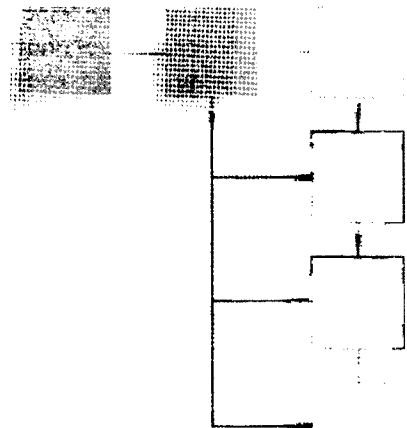
Nếu bạn là thành viên của nhóm (phát triển)

Mặc dù bạn đã có chút kinh nghiệm với Scrum, nhưng có thể bạn thấy một vài thứ thú vị khi đọc các chương sau:

- Chương 3 (Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung)
- Chương 4 (Thu thập yêu cầu cho Product Backlog)
- Chương 5 (Tạo ra ước tính điểm story so sánh được để triển khai Scrum cho toàn doanh nghiệp)
- Chương 6 (Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm)
- Chương 7 (Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành và lập kế hoạch Sprint)
- Chương 8 (Product Owner)
- Chương 9 (Tầm quan trọng của kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp)
- Chương 10 (Tầm quan trọng của làm việc nhóm)
- Chương 13 (Tự đánh giá độ sẵn sàng của dự án Scrum)
- Chương 14 (Khi nào bạn cần một ScrumMaster?)

Nếu có thêm thời gian, chúng tôi khuyến nghị bạn đọc toàn bộ cuốn sách từ đầu tới cuối, bao gồm cả Phụ lục A với hai ví dụ về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công dùng những lời khuyên trong cuốn sách này.

LỜI KHEN TẶNG



Cuốn sách của Andrew Pham là sự đền đáp cho lời nguyên cầu của những người mới đến với Agile/Scrum. Cuốn sách cung cấp từ những kỹ thuật đánh giá tính linh hoạt cho dự án/doanh nghiệp, tới hướng dẫn triển khai. Ông đã không chỉ đưa vào khía cạnh lý thuyết, mà còn cả khía cạnh con người và thực hành. Andrew Pham chỉ đơn giản đưa ra tất cả những trớ ngại mà ông và nhóm của mình đã gặp phải trong nhiều năm, và cung cấp những hướng dẫn học tập cùng với câu trả lời cho hầu hết vấn đề có thể gặp phải khi bắt đầu quá trình chuyển đổi với Agile. Cuốn sách là một công cụ tuyệt vời cho bất cứ nhóm nào muốn thử khám phá con đường Agile!

Sameer Bendre, CSM, PMP

3i Infotech Consulting

Tôi biết Andrew Pham trong nhiều năm, từ khi tôi còn làm việc dưới sự điều hành của ông ở AT&T (trước đây là SBC) và luôn cảm thấy thích thú vì những điều học được từ ông.

Andrew Phạm đã giải thích đặc biệt rõ ràng cho chúng tôi về Agile và Tầm nhìn kiến trúc, điều đó được ông truyền đạt lại trong cuốn sách này.

Xét về khía cạnh kỹ thuật thì cuốn sách này rất thú vị bởi nó có ý tưởng mới về quản lý dự án hoặc quy trình phát triển phần mềm nói chung. Cuốn sách được viết rất tốt, đáng là cuốn phải đọc cho những ai phải cải tiến cách làm việc hiện thời, đồng thời là một cuốn sách tham khảo tốt trong ngành CNTT.

Ben Oguntona, MBA

Quản lý Hệ thống cấp cao, AT&T

Trong cuốn sách này, Andrew Pham đã cung cấp cái nhìn rất khác về Scrum và giúp làm sáng tỏ nhiều vấn đề mà nhóm dự án phải đối mặt khi bắt đầu chấp nhận Scrum. Andrew mô tả nhiều kỹ thuật quản lý dự án tốt để giúp nhóm dự án đạt hiệu quả nhất.

Lời khen tặng

Andrew cũng kết nối khoảng trống giữa nhiều sách Scrum và quản lý dự án thông qua việc chỉ ra cách giao tiếp với những nhà điều hành bằng cách sử dụng thuật ngữ tài chính, cách sử dụng kỹ thuật ước tính khách quan thay cho cách thông thường “poker estimation” (ước tính với bộ bài) và nơi mà kiến trúc phần mềm phù hợp trong Scrum.

Đây là một cuốn sách tốt, có thể đọc nhanh nhầm chuẩn bị cho các nhà quản lý dự án những kiến thức để triển khai Scrum thành công.

Scott Booth, MCP, Nhà quản lý

Pariveda Solutions, Dallas, Texas

Cuốn sách là bài thuyết trình súc tích về các bước và những khái niệm mà một tổ chức cần phải nhận thức khi áp dụng Scrum. Công ty tôi đang ở Sprint thứ ba trong dự án Scrum đầu tiên. Tôi không thể đợi tới khi phát hành bản cuối của cuốn sách để có thể chia sẻ với những thành viên còn lại trong nhóm. Theo nhiều khía cạnh khác nhau, cuốn sách như một lời khẳng định rằng chúng tôi đang đi đúng hướng và cũng là một nhắc nhở tuyệt vời về những thứ chúng tôi vẫn cần phải cải tiến. Chương bàn về việc làm gì để trở thành Product Owner tốt đã mô tả gần như chính xác về người đang đảm nhận vai trò đó trong công ty chúng tôi. Tôi đã không có cơ hội để hoàn thành khóa huấn luyện chứng chỉ ScrumMaster, nên thông qua việc đọc về cách ScrumMaster và Product Owner làm việc cùng nhau để quản lý nhóm đã giúp tôi có một số hiểu biết quan trọng.

Dennis Palmer

Quản lý sản phẩm/ScrumMaster

Esquire Innovations, Inc.

Cuốn sách của Andrew Pham là cuốn tham khảo rất súc tích về Agile và Scrum, cuốn sách sử dụng nhiều thuật ngữ thực hành hơn là hàn lâm. Cuốn sách dễ đọc cho cả quản lý dự án non trẻ và cấp cao. Ngôn ngữ của cuốn sách đã mô tả một sự thực rằng Agile và Scrum là cách nhìn quản lý dự án tuyệt đối tự nhiên.

Bởi Andrew Pham và tôi đã dẫn dắt cài đặt thành công gói PeopleSoft đầu tiên tại Computer City nhiều năm trước, tôi có thể nhận ra rằng trong cuốn sách có nhiều yếu tố Scrum mà ông đã sử dụng. Về cơ bản, những yếu tố này nhằm xây dựng sự cộng tác tốt và thường xuyên với người dùng, trao quyền cho nhóm trong khi đảm bảo rằng họ chuyển giao những phần sản phẩm tăng trưởng nhỏ chạy được.

Cuốn sách đã đưa được tiến trình tự nhiên này vào trong một khung làm việc chính thức hơn trong khi nhấn mạnh việc xây dựng tính linh hoạt trong quy trình. Là một Chuyên gia phần mềm cấp cao đã làm việc trong những dự án lớn và nhỏ, tôi thấy rằng đây chính là cuốn sách có giá trị với những nhà quản lý dự án, không phân biệt quy mô và phạm vi của dự án.

Anwar Bardai, Chuyên gia Phần mềm cấp cao

CompuCom, Inc.

Tôi thấy rằng cuốn sách của Andrew Pham và Phuong-Van Pham vô cùng thiết thực. Cuốn sách đưa ra những câu trả lời ngắn gọn về những việc tôi thiếu mà tôi cần làm. Tất nhiên bạn có thể đọc toàn bộ cuốn sách và điều đó rất đáng làm. Nhưng ngày nay, nhiều người thấy rằng họ không có thời gian để đọc cả một cuốn sách mà họ chỉ muốn câu trả lời. Và thường một cuốn sách thường quá bao quát, chứa những thứ không phù hợp với bạn, trong khi tình huống của mỗi nhóm lại khác nhau. Cuốn sách này là câu trả lời. Tùy vào tình huống, sách hướng dẫn cho bạn biết nên đọc chương nào. Hãy chọn lấy tình huống của mình trong danh sách, viết lại những chương cần đọc và cuốn sách đã được tùy biến cho riêng bạn.

Những cuốn sách khác về Agile/Scrum mà tôi đã đọc thường che đậm những phàn nán hoặc thiếu sót của Agile/Scrum thường có ở những công ty bắt đầu xem xét sử dụng Agile/Scrum. Andrew nêu ra những vấn đề hoặc khó khăn và đưa ra lời khuyên thiết thực tùy thuộc vào lựa chọn của bạn và cách triển khai để giúp đảm bảo một dự án thành công.

Dennis Simpson, Systems Crisis Resolution, MBA, MS, CCP

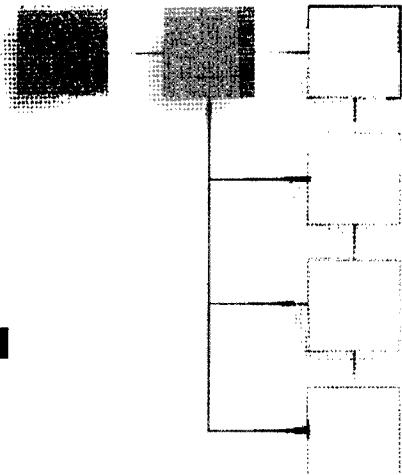
Cuốn sách của Andrew và Phuong-Van Pham là một bài trình bày ngắn gọn, rõ ràng về các khái niệm của Agile và Scrum cho các chuyên gia, nhà điều hành đang tò mò liệu những dự án CNTT của họ có thể được lợi từ chúng. Cuốn sách không chỉ có những lời khuyên kỹ thuật xuất sắc cho những vấn đề của quản lý dự án và chuyên gia phân tích nghiệp vụ, mà còn đề cập tới những vấn đề tài chính và nguồn nhân lực có ảnh hưởng tới cấp quản lý kinh doanh. Nếu có một cuốn sách có thể làm cho Agile hoạt động trong một tổ chức thì đây chính là cuốn sách đó.

Harold Thomas, CBAP

Phòng Dịch vụ Việc làm và Gia đình của Bang Ohio

CHƯƠNG 1

CHUẨN BỊ CHO AGILE VÀ SCRUM



Giống như bao người khác, chúng tôi rất phấn khích với việc áp dụng quản lý và phát triển dự án linh hoạt - Agile (thích ứng) nói chung và áp dụng Scrum nói riêng.

Khác với những người rất giáo điều về Agile và Scrum, chúng tôi không tin rằng thế giới doanh nghiệp hiện nay đã hoàn toàn được tái tổ chức với Scrum. Có thể một vài công ty phần mềm thương mại đã làm việc đó, nhưng phần lớn những công ty khác đều vẫn chưa sa thải các chuyên gia của họ, hoặc điều chuyển những chuyên gia này sang chức danh khác như những gì mà các nhà giáo điều về Scrum tin rằng họ nên làm. Hầu hết những công ty này vẫn có các tổ chức theo cách hiện tại họ đang làm, với nhiều chức năng riêng biệt trong các bộ phận CNTT. CapitalOne, một trong những công ty đã ứng dụng Agile mà hay được các nhà huấn luyện Scrum trích dẫn, hiện vẫn đang thuê các chuyên gia phân tích nghiệp vụ, chuyên gia phân tích hệ thống nghiệp vụ và quản trị viên dự án. Điều này cũng chính là thực tế ở nhiều công ty lớn khác được biết là đã chuyển đổi sang Scrum, chẳng hạn như Sabre, Verizon, NBC Universal, General Dynamics, Texas Instruments và American Airlines.com, ...

Chúng ta sẽ bàn thêm về nguồn gốc và những nguyên lý cơ bản của Scrum ở phần sau của chương. Còn bây giờ, hãy nói về Scrum, như trong môn bóng bầu dục, là một cách để tái khởi động trận đấu khi xảy ra sự cố hoặc khi bóng đã ra ngoài cuộc chơi. Ý tưởng ở đây là để giữ cho trận đấu (phát triển phần mềm) được tiếp tục.

Mặc dù Scrum có thể được sử dụng bên ngoài lĩnh vực phát triển phần mềm, nhưng trong cuốn sách này chúng ta sẽ tập trung vào Scrum như là một khung quy trình dành cho việc quản lý và phát triển dự án phần mềm đơn lẻ.

Với việc quản lý các dự án theo kiểu truyền thống là mệnh lệnh và kiểm soát trong một thời gian dài, chúng tôi thường xuyên chứng kiến việc Agile và Scrum có vẻ như đã giúp cho các nhóm tạo ra những phần mềm hiệu quả hơn kiểu ngược lại là mệnh lệnh và kiểm soát.

Điều này không nhằm để nói rằng kiểu hướng kế hoạch truyền thống không còn hoạt động tốt trong mọi tình huống. Có nhiều khía cạnh ở trong cách tiếp cận truyền thống

Chương 1 ▪ Chuẩn bị cho Agile và Scrum

vẫn còn có ích trong Agile và Scrum. Những gì chúng tôi đang nói đơn giản chỉ là chúng ta có thể làm tốt hơn nếu như ta biết dừng lại, cân nhắc những việc đã làm và học cách cải tiến các kỹ thuật, quy trình của mình bằng những ý tưởng và khái niệm mới.

Bởi vì mọi người và các công ty đều có xu hướng nắm bắt sự thay đổi một cách chậm chạp, nên việc phong trào Agile và đặc biệt là Scrum, được đón nhận và tạo ra nhiều hứng thú là một tín hiệu tốt. Cùng nhau, chúng tôi hy vọng góp phần vào việc mở rộng ứng dụng Agile và Scrum thông qua cuốn sách này, chúng đã được hình thành từ nhiều năm kinh nghiệm thực tế trong quản lý và phát triển dự án phần mềm.

Giống như nhiều người khác, nếu đã gặp phải những trở ngại trong quá khứ, bạn sẽ thấy rằng Agile và Scrum mới mẻ một cách đáng kinh ngạc và bạn sẽ học cách để triển khai thành công chúng trong một dự án. Ngược lại, con đường của bạn có thể gặp một số khó khăn dẫn đến việc dự án bị thất bại.

Có rất nhiều lý do dẫn đến việc đôi khi các nhóm gặp thất bại với Scrum. Một trong những lý do thường gặp nhất là nhiều chuyên gia vẫn chưa quen với Scrum, kể cả sau khi đã tham gia một lớp học Scrum hoặc đọc một số bài viết hay về nó. Lý do thứ hai có thể là do dự án của họ quá phức tạp và cần dùng những kỹ thuật cao cấp hơn nhằm giảm độ phức tạp và kiểm soát được dự án. Lý do thứ ba có thể là do tổ chức của họ vẫn chưa được thiết lập cho Scrum, hoặc là các nhóm vẫn chưa biết làm cách nào để sử dụng Scrum trong những ràng buộc hiện tại của công ty. Điều này có thể là do sự thiếu kinh nghiệm hoặc là nhóm được hướng dẫn không đúng bởi những ScrumMaster hoặc nhà huấn luyện giáo điều. Chúng ta sẽ giải quyết một số lý do đó ở trong cuốn sách hướng dẫn thực tế này.

Bất kể nguyên nhân là gì, Agile và Scrum đi kèm với sự thay đổi và thay đổi thì luôn khó khăn, trừ phi chúng được quản lý đúng cách. Nhưng với việc quản lý tốt thì những khó khăn sẽ trở thành cơ hội.

Một trong những mục tiêu của cuốn sách là nhằm chuẩn bị cho bạn trước những thay đổi mà bạn và nhóm của mình sẽ gặp phải với Agile và Scrum. Cuốn sách có thể giúp bạn thực hiện quy trình “trơn tru” nhất có thể trong khi mang lại cho nhóm của bạn cơ hội để thích ứng, nắm bắt và thành công trong dự án Scrum.

NỀN TẢNG CỦA PHÁT TRIỂN PHẦN MỀM VÀ QUẢN LÝ DỰ ÁN LINH HOẠT LÀ GÌ?

Không nghi ngờ gì cả, nền tảng của Phát triển phần mềm và Quản lý Dự án linh hoạt là Tuyên ngôn Agile (Agile Manifesto) và Tuyên ngôn Tương hỗ (Declaration of Inter-Dependence).

Năm 2001, một nhóm các chuyên gia phần mềm tụ họp tại Snowbird Resort ở Utah đã công bố một bản dự thảo được biết đến như là Tuyên ngôn Agile (www.agilemanifesto.org):

Bản tuyên ngôn được tạm dịch như sau:

Nền tảng của Phát triển Phần mềm và Quản lý Dự án Linh Hoạt là gì?

“Chúng tôi đã phát hiện ra những phương pháp phát triển phần mềm tốt hơn bằng cách thực hiện và giúp đỡ những người khác thực hiện nó. Qua công việc này, chúng tôi đã đi đến việc đánh giá cao:

- Cá nhân và sự tương tác hơn là quy trình và công cụ;
- Phần mềm chạy tốt hơn là tài liệu đầy đủ;
- Cộng tác với khách hàng hơn là đàm phán hợp đồng;
- Đáp ứng với sự thay đổi hơn là theo sát kế hoạch.

Mặc dù những điều bên phải vẫn còn giá trị, nhưng chúng tôi đánh giá cao những mục ở bên trái hơn.”

[© 2001 Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas.]

Cùng với bốn giá trị trên, Tuyên ngôn Agile còn có mười hai nguyên lý:

1. Ưu tiên cao nhất của chúng tôi là thỏa mãn khách hàng thông qua việc chuyển giao sớm và liên tục các phần mềm có giá trị.
2. Chào đón những thay đổi về yêu cầu, kể cả rất muộn trong quá trình phát triển. Các quy trình Agile tận dụng sự thay đổi cho các lợi thế cạnh tranh của khách hàng.
3. Thường xuyên chuyển giao phần mềm chạy tốt tới khách hàng, từ vài tuần đến vài tháng, ưu tiên những khoảng thời gian ngắn hơn.
4. Nhà kinh doanh và nhà phát triển phải làm việc cùng nhau hằng ngày trong suốt dự án.
5. Đội ngũ xây dựng dự án là những cá nhân có động lực. Cung cấp cho họ môi trường, sự hỗ trợ cần thiết và tin tưởng họ để hoàn thành công việc.
6. Phương pháp hiệu quả nhất để truyền đạt thông tin tới nhóm phát triển và trong nội bộ nhóm phát triển là hội thoại trực tiếp.
7. Phần mềm chạy tốt là thước đo chính của tiến độ.
8. Các quy trình Agile thúc đẩy sự phát triển bền vững. Các nhà tài trợ, nhà phát triển và người dùng có thể duy trì một nhịp độ liên tục không giới hạn.
9. Liên tục quan tâm đến thiết kế tốt và hoàn hảo về kỹ thuật để gia tăng sự linh hoạt.
10. Sự đơn giản - nghệ thuật tối đa hóa lượng công việc chưa hoàn thành - là căn bản.
11. Các kiến trúc, yêu cầu và thiết kế tốt nhất sẽ được tạo ra bởi các nhóm tự tổ chức.
12. Nhóm phát triển sẽ thường xuyên suy ngẫm về cách thức để trở nên hiệu quả hơn, sau đó họ sẽ điều chỉnh và thay đổi các hành vi của mình cho phù hợp.

Mặc dù Tuyên ngôn Agile được soạn thảo năm 2001 vài năm sau khi Scrum được công bố tại hội nghị OOPSLA (Object Oriented Programming, Systems, Languages,

Chương 1 • Chuẩn bị cho Agile và Scrum

and Applications - Lập trình hướng đối tượng, Hệ thống, Ngôn ngữ và Ứng dụng) năm 1996, nhưng có một điều được công nhận rõ ràng giữa các chuyên gia đó là bản tuyên ngôn đã có ảnh hưởng lớn đến Scrum. Sự ảnh hưởng này thể hiện rõ rệt trong cuốn sách thứ hai của Ken Schwaber: *Agile Project Management with Scrum* (tạm dịch: Quản lý Dự án linh hoạt với Scrum), ông đã viết rằng Scrum là một trong những quy trình Agile có những giá trị và nguyên lý như mô tả trong Tuyên ngôn Agile.

Trong khi Tuyên ngôn Agile giải quyết việc phát triển phần mềm, bản Tuyên ngôn Tương hỗ về Quản lý Dự án linh hoạt mà một nhóm các chuyên gia cùng nhau đưa ra vào năm 2005 lại tập trung nhiều hơn về khía cạnh quản lý dự án (<http://pmdoi.org>):

Bản tuyên ngôn được tạm dịch như sau:

"Chúng tôi là một cộng đồng gồm các lãnh đạo dự án đã đạt được thành công trong việc chuyển giao kết quả. Để đạt được những kết quả này:

- Chúng tôi **tăng tỷ lệ hoàn vốn đầu tư** bằng cách biến những luồng giá trị liên tục thành trọng tâm của mình.
- Chúng tôi **chuyển giao những kết quả đáng tin cậy** bằng cách lôi kéo khách hàng thường xuyên tương tác và chia sẻ quyền sở hữu.
- Chúng tôi **chờ đợi các bất ổn** và quản lý chúng qua các phân đoạn, dự đoán và thích ứng.
- Chúng tôi **giải phóng sức sáng tạo và đổi mới** bằng cách công nhận rằng các cá nhân là nguồn gốc cao nhất của giá trị và tạo ra một môi trường nơi họ có thể làm nên sự khác biệt.
- Chúng tôi **tăng hiệu suất** thông qua phân nhóm trách nhiệm đối với các kết quả và chia sẻ trách nhiệm về hiệu quả của nhóm.
- Chúng tôi **nâng cao hiệu quả và độ tin cậy** thông qua các chiến lược, quy trình và thực tiễn cụ thể trong từng tình huống."

[©2005 David Anderson, Sanjiv Augustine, Christopher Avery, Alistair Cockburn, Mike Cohn, Doug DeCarlo, Donna Fitzgerald, Jim Highsmith, Ole Jepsen, Lowell Lindstrom, Todd Little, Kent McDonald, Pollyanna Pixton, Preston Smith and Robert Wysocki.]

Cho dù cả Tuyên ngôn Agile và Tuyên ngôn Tương hỗ (DOI) xuất phát từ tâm trí của các chuyên gia trước hay sau khi họ đã chịu một vài ảnh hưởng từ Scrum hoặc bất cứ quy trình Agile nào đã tồn tại, thì điều đó cũng không thực sự quan trọng.

Điều quan trọng là nếu bạn thực sự hiểu được ý nghĩa của bản Tuyên ngôn Agile và DOI thì bạn đã "đặt một chân" vào việc thích ứng với Scrum khi cần thiết mà không đi ngược lại nền tảng cơ bản của Agile.

NGUỒN GỐC CỦA SCRUM

Trong lịch sử, thuật ngữ Scrum xuất hiện từ một bài viết trên tạp chí Harvard Business Review của Hirotaka Takeuchi và Ikujiro Nonaka năm 1986. Trong bài viết có tên "The New New Product Development Game" (tạm dịch: Trò chơi phát triển sản phẩm mới)

đó, Takeuchi và Nonaka đã mô tả một cách tiếp cận toàn diện. Trong cách tiếp cận này các nhóm dự án được tạo thành từ những nhóm nhỏ liên chức năng làm việc hướng tới một mục tiêu chung. Các tác giả đã so sánh những nhóm này với sự hình thành Scrum¹ trong môn bóng bầu dục.

Khi xây dựng một công cụ Phân tích và Thiết kế hướng đối tượng (OOAD - Object Oriented Analysis and Design) tại Easel và sau đó với vai trò Phó Chủ tịch kỹ thuật tại Công ty Easel, Jeff Sutherland nhận ra rằng nhóm phần mềm của mình cần một phiên bản phát triển ứng dụng nhanh cài tiến. Điều mà ông muốn là một quy trình tương tự như Scrum. Với quy trình này thì ở cuối mỗi phân đoạn ngắn, CEO của Easel sẽ nhìn thấy mã chạy được trình diễn thay vì biểu đồ Gantt.

Trước hoặc sau khoảng thời gian đó, Ken Schwaber cũng đang tích cực tìm kiếm cách giúp công ty Advanced Development Methods (ADM) của ông nâng cao quy trình phần mềm để cài tiến năng suất làm việc của nhóm.

Sau khi tiếp tục phân tích cách thức giúp những nhà cung cấp phần mềm độc lập (ISV - Independent Software Vendor) xây dựng phần mềm thành công, Ken đã đi tới kết luận là tất cả những quy trình phát triển của họ đều tương tự nhau và đều sử dụng quy trình thực nghiệm, quy trình này liên tục yêu cầu sự thanh tra và thích nghi.

Theo yêu cầu của Object Management Group (OMG) vào năm 1995, Jeff và Ken đã cùng nhau nhau tóm tắt lại những gì đã học được qua nhiều năm; họ đã tạo ra một phương pháp luận mới và đặt tên là Scrum. Scrum được mô tả trong bài viết "Scrum and the Perfect Storm" (tạm dịch: Scrum và Cơn bão Hoàn hảo) của Schwaber đăng tại www.controlchaos.com/my-articles.

Scrum hoạt động như thế nào

Lưu ý trong Hình 1.1, nhóm Scrum nên là liên chức năng, bao gồm một ScrumMaster, một Product Owner và Nhóm phát triển (gọi tắt là "Nhóm"), với tất cả kỹ năng cần thiết (như thu thập yêu cầu, thiết kế, lập trình và kiểm thử) để xây dựng sản phẩm phần mềm.

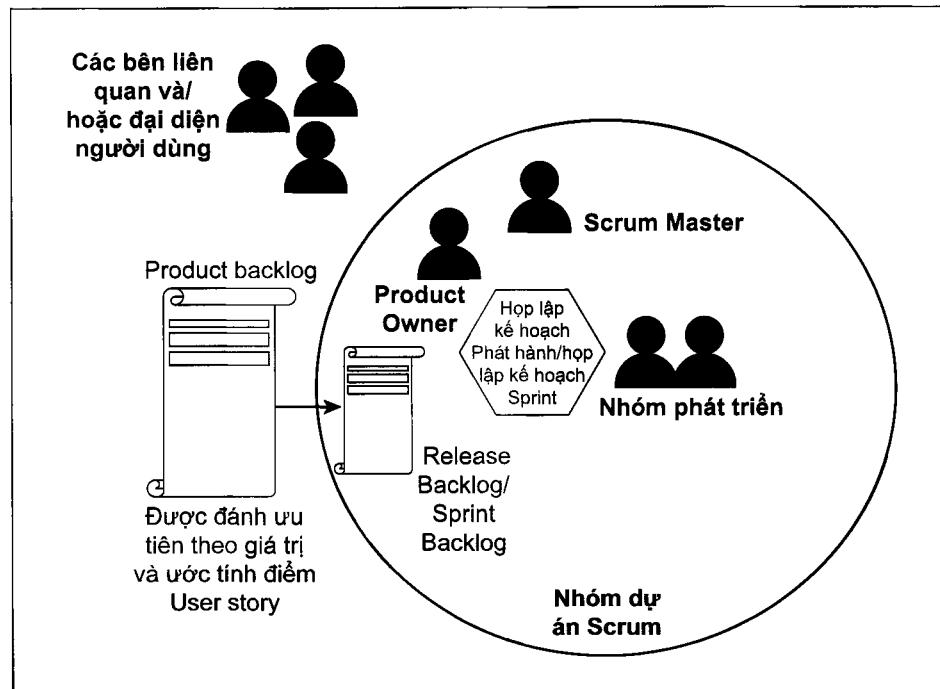
Dù trong trường hợp tốt nhất, nhóm Scrum là liên chức năng hoàn toàn, thì cũng không nhất thiết phải tuân thủ rằng các thành viên của dự án Scrum phải thuộc một cấu trúc nhóm dự án ổn định về mặt tổ chức và phân cấp. Chỉ khi công ty của bạn kiên quyết triển khai Scrum cho toàn doanh nghiệp và đã tự tổ chức lại toàn bộ, thì hãy tuân thủ những khuyến nghị về cấu trúc tổ chức Scrum (Scrum organizational structure recommendations), còn không nhóm Scrum vẫn được mượn từ những tổ chức khác như Đảm bảo chất lượng (QA), Kiến trúc doanh nghiệp (Enterprise Architecture), Tiền sản xuất (Pre-production) hoặc DBA (Quản trị cơ sở dữ liệu).

Ngay cả những công ty thường được nhắc đến là đã áp dụng Scrum thì vẫn tồn tại nhóm Quản lý dự án (Project Management), Đảm bảo chất lượng, Tiền sản xuất, Phân tích Hệ thống nghiệp vụ và Kiến trúc doanh nghiệp hoạt động với chức năng

¹Scrum: Là một cơ chế trong trò chơi bóng bầu dục để đưa "bóng chết" vào cuộc chơi. Từ "scrum" có nghĩa là chen lấn, hàm ý sự hỗn độn, liên chức năng và tự quản cao độ trong các tình huống thực tiễn.

Chương 1 • Chuẩn bị cho Agile và Scrum

riêng rẽ và nhóm dự án phải thương lượng với những nhóm này để dự án Scrum chạy được. Rất hiếm khi một công ty sa thải toàn bộ quản lý dự án (Scrum không có quản lý dự án) hoặc đổi chức danh của họ sang Scrum Master hoặc tổ chức lại những nhóm CNTT riêng biệt vào những nhóm dự án Scrum đa ngành. Điều đó có thể là ước mơ, nhưng hiện thời chưa có.



Hình 1.1

Product backlog, Release backlog và Sprint backlog.

Trong Hình 1.1, bạn có thể thấy cách một dự án Scrum khởi động. Như bạn thấy trong sơ đồ, tất cả đều bắt đầu với Product Owner, người chịu trách nhiệm lấy đầu vào từ những bên liên quan hoặc đại diện của họ, để xây dựng một danh sách các yêu cầu nhằm tạo ra một Product backlog.

Chỉ đơn giản là đưa vào Product backlog mọi yêu cầu về tính năng nghiệp vụ, công nghệ, vấn đề kỹ thuật, việc sửa lỗi. Đây là một danh sách có thứ tự ưu tiên.

Một số người thực hành và tác giả, như Henrik Kniberg đã viết trong cuốn “*Scrum and XP from the Trenches*” (tạm dịch: Scrum và XP từ những chiến hào), muốn giữ Product backlog ở mức độ nghiệp vụ để chỉ chứa những yêu cầu nghiệp vụ.

Trong Chương 4 “Thu thập các yêu cầu trực quan cho Product Backlog”, bạn sẽ biết được rằng các yêu cầu người dùng cho Product backlog trong Scrum thường được

thu thập dưới dạng những User story ngắn trong một buổi hội thảo về yêu cầu người dùng một hoặc hai ngày trước buổi họp lập kế hoạch phát hành và lập kế hoạch Sprint.

Mặc dù trong thời gian đầu, việc lập kế hoạch phát hành là không bắt buộc trong Scrum, nhưng thời gian đã chứng minh là nó rất hữu ích trong việc giúp nhóm Scrum trở nên hiệu quả hơn. Bởi thế chúng tôi rất khuyến khích Product Owner hãy cùng làm việc với nhóm trong buổi lập kế hoạch phát hành mặc dù đó là việc khó khăn, bởi điều đó yêu cầu Product Owner phải nghiên cứu sản phẩm trước buổi họp lập kế hoạch.

Product Owner càng biết nhiều về sản phẩm, thì càng có thể giúp nhóm nhiều hơn. Mục tiêu then chốt của việc lập kế hoạch phát hành là để nhóm Scrum xác định tất cả các phát hành mà sản phẩm phần mềm nên có, có thể kèm theo lịch trình chuyển giao. Thông thường, việc lập kế hoạch phát hành kéo dài bốn giờ với một nhóm Scrum có Sprint bốn tuần.

Bên cạnh việc lập kế hoạch phát hành, nhóm Scrum cũng nên thực hiện việc lập kế hoạch Sprint, đó có thể là một phần của quá trình lập kế hoạch phát hành hoặc là một phiên làm việc độc lập sau khi hoàn thành việc lập kế hoạch phát hành.

Thông thường, buổi họp lập kế hoạch Sprint kéo dài khoảng tám giờ cho mỗi Sprint bốn tuần và nên được điều chỉnh xuống còn bốn giờ cho Sprint hai tuần.

Theo cách làm thông thường, buổi họp lập kế hoạch Sprint nên được chia thành hai cuộc họp bốn giờ.

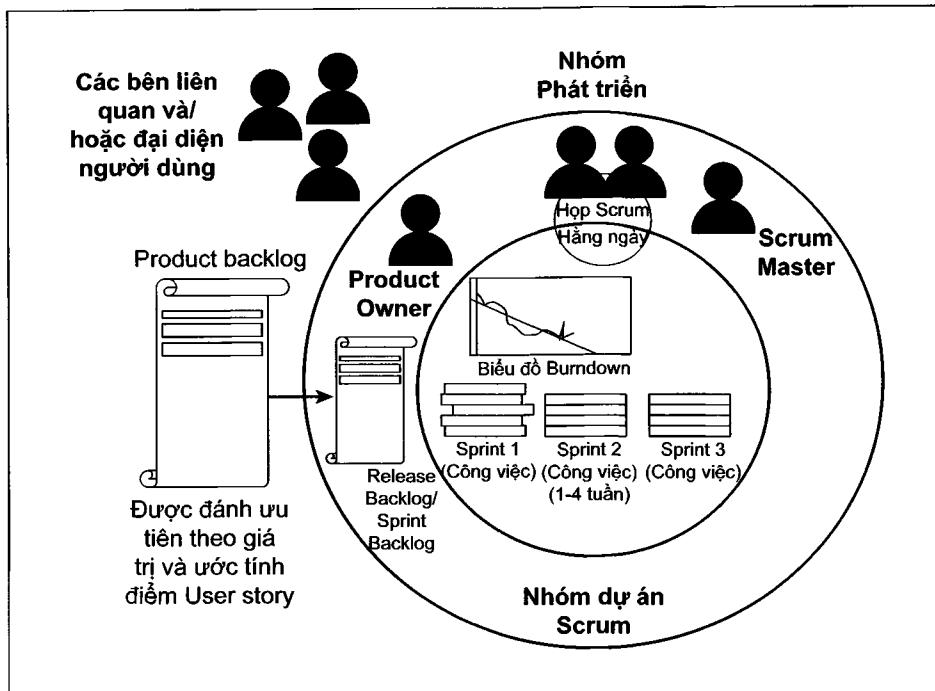
Trong phần đầu tiên, Product Owner sẽ xem xét yêu cầu, như các User story, cùng với phản hồi của nhóm để quyết định User story nào nên được phát triển ở Sprint nào và mục tiêu của chúng là gì. Phần đầu của buổi họp chủ yếu để giải đáp câu hỏi "Là gì"

Phần thứ hai của buổi họp lập kế hoạch Sprint tập trung vào "Cách làm", nhóm Phát triển sẽ cố gắng xác định những đầu việc từ những User story đã chọn và lượng thời gian (theo giờ) cần thiết để chuyển hóa những công việc này thành phần tăng trưởng sản phẩm có thể chuyển giao được. Trừ khi nhóm sử dụng phần mềm lập kế hoạch, còn không mọi công việc phát triển của Sprint sẽ được dán vào Task Board (Bảng công việc), một vài loại bảng trắng treo tường, để giúp nhóm dễ dàng phân bổ công việc và theo dõi.

Ngay khi cuộc họp lập kế hoạch phát hành và Sprint kết thúc, nhóm bắt đầu công việc của sprint hiện thời (Hình 1.2) cùng với những cuộc họp Scrum hằng ngày 15 phút (hay họp đứng hằng ngày).

Ban đầu Họp Đứng Hằng ngày có thể kéo dài tới 30 phút, nhưng như một phần của sự tiến hóa Scrum, hoặc sự điều chỉnh, mà chúng ta sẽ nói thêm trong Chương 12, "Làm thế nào để thích ứng với Scrum mà không phá bỏ tính linh hoạt cơ bản hoặc triển khai Scrum but tiêu cực," thì trong thực tế, độ dài của buổi họp càng ngày càng được rút ngắn xuống, đến ngày nay nó kéo dài khoảng 15 phút.

Chương 1 ▪ Chuẩn bị cho Agile và Scrum



Hình 1.2

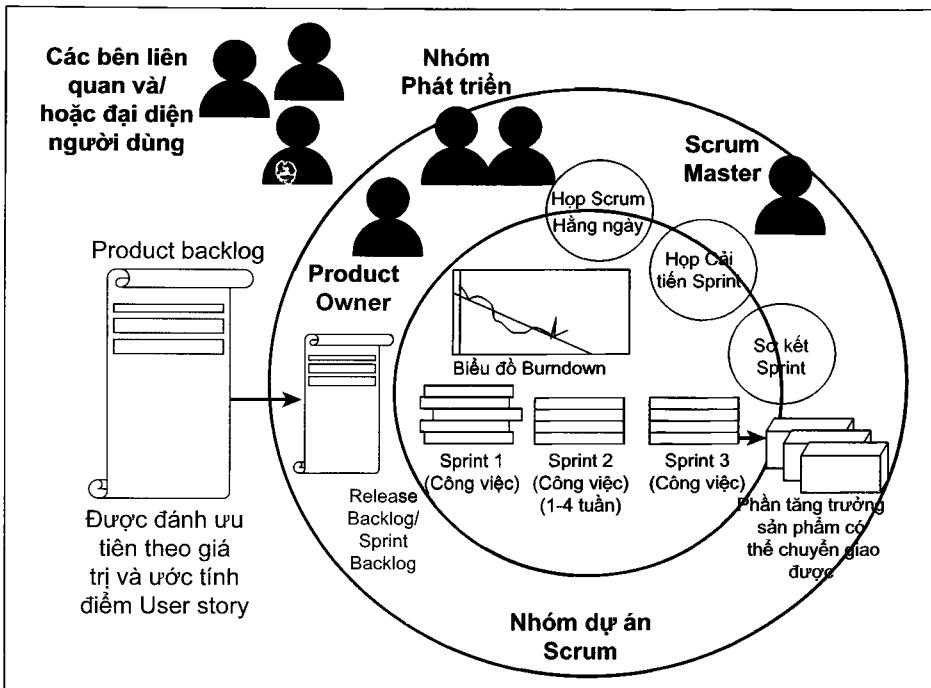
Các Sprint, Biểu đồ burndown và Họp Scrum hằng ngày.

Thông thường, một Sprint kéo dài từ một tới bốn tuần. Ngoại trừ một số tình huống rất đặc biệt, khi nhóm và Product Owner đồng ý, còn không Sprint Backlog sẽ không nhận thêm hoặc xóa bỏ gì cả trong toàn bộ Sprint, nhưng điều này cũng là một ngoại lệ so với bình thường.

Không giống trong quy trình truyền thống, nơi mà quản lý dự án chịu trách nhiệm tổ chức các cuộc họp trạng thái hàng tuần để theo dõi trạng thái dự án, trong Scrum, hàng ngày nhóm cùng nhau thanh tra, không phải trạng thái dự án, mà là tiến độ của nhóm hướng tới mục tiêu Sprint. Đó chính là lý do mà bạn thường nghe mọi người nói rằng Họp Scrum Hàng ngày không phải là họp trạng thái.

Nhóm tạo biểu đồ burndown để hiển thị lượng công việc còn lại tới khi nhóm hoàn thành Sprint, đây là công cụ để theo dõi tiến độ của nhóm hướng tới mục tiêu Sprint. Mặc dù việc tạo biểu đồ burndown là trách nhiệm của nhóm, nhưng khi nhóm không có thời gian thì ScrumMaster có thể cập nhật biểu đồ.

Ngay trước khi kết thúc mỗi Sprint, nhóm sẽ gặp Product Owner để thảo luận trong một buổi họp do Scrum Master tổ chức (buổi họp Sơ kết Sprint), đây là một phần của cơ chế Thanh tra và Thích nghi (Hình 1.3). Đây là một buổi họp nữa được đóng khung thời gian, thông thường buổi họp này diễn ra trong bốn giờ cho một Sprint bốn tuần hoặc hai giờ cho một Sprint hai tuần.

**Hình 1.3**

Sơ kết Sprint và Hợp cải tiến Sprint.

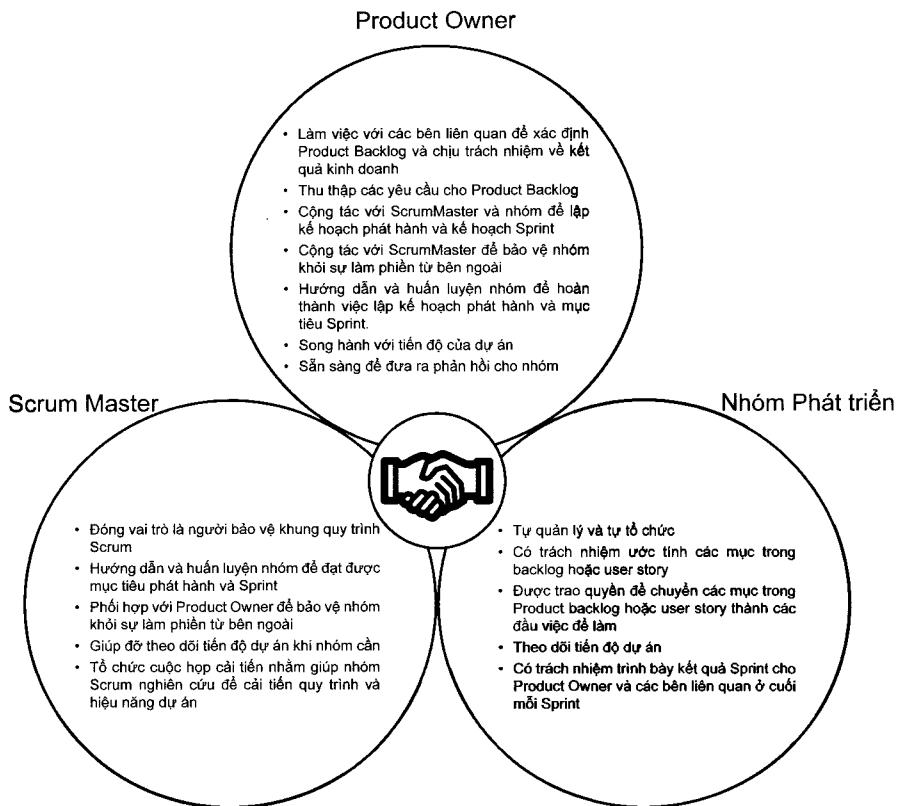
Buổi họp có nhiều mục đích: Đầu tiên là để nhóm Scrum và Product Owner thảo luận về những công việc đã hoàn thành và những công việc chưa hoàn thành. Mục đích thứ hai là để nhóm trình diễn kết quả đã xây dựng được cho Product Owner để nhận phản hồi. Cuối cùng, mục đích thứ ba là để có thể cập nhật thông tin từ Product Owner liên quan đến bất cứ thay đổi nào về sản phẩm hoặc định hướng của thị trường.

Ngay sau buổi họp Sơ kết Sprint và trước khi bắt đầu Sprint tiếp theo, nhóm Scrum sẽ có một buổi Hợp Cải tiến Sprint để xác định những gì tốt và những gì không tốt trong Sprint hiện thời. Mục đích là xem cách họ cộng tác như thế nào để đạt hiệu quả hơn nữa trong Sprint tiếp theo.

Theo cách làm thông thường thì buổi họp cải tiến thường kéo dài ba giờ cho một Sprint một tháng, nhưng độ dài này có thể được điều chỉnh giống như những cuộc họp đóng khung thời gian khác, tỷ lệ với độ dài của Sprint, ví dụ như hai giờ cho một Sprint hai tuần.

Hình 1.4 cung cấp một bức tranh tổng thể về trách nhiệm cộng tác của những thành viên trong nhóm Scrum.

Chương 1 ▪ Chuẩn bị cho Agile và Scrum



Hình 1.4

Sự cộng tác giữa nhóm, Scrum Master và Product Owner.

Nhìn chung thì Scrum không có vẻ quá phức tạp như một khung làm việc dự án nói chung? Điều đó là đúng, nhưng mặc dù Scrum có vẻ khá đơn giản về mặt lý thuyết, thì lại có thể rất phức tạp để triển khai, đặc biệt là khi phương pháp Scrum còn mới với công ty của bạn hoặc khi bạn muốn tăng tốc dự án Scrum với một số phương pháp thực hành hiện thời.

TAI SAO AGILE VÀ SCRUM MANG LẠI HIỆU QUẢ TRONG QUẢN LÝ DỰ ÁN PHẦN MỀM?

Mặc dù Agile và đặc biệt là Scrum có thể khó triển khai, nhưng thực tế đã chứng minh nó thực sự mang lại hiệu quả nếu triển khai đúng cách.

Khi đọc cuốn sách này, bạn sẽ thấy nhiều lý do khiến cho Agile và Scrum mang lại hiệu quả cao hơn trong việc quản lý và phát triển dự án. Tuy nhiên, chúng tôi muốn nhấn mạnh bốn lợi thế đầu tiên:

- Một cơ chế giảm thiểu rủi ro mang tính hệ thống: Tất cả những người chịu trách nhiệm lập kế hoạch và thực hiện dự án đều biết tầm quan trọng của

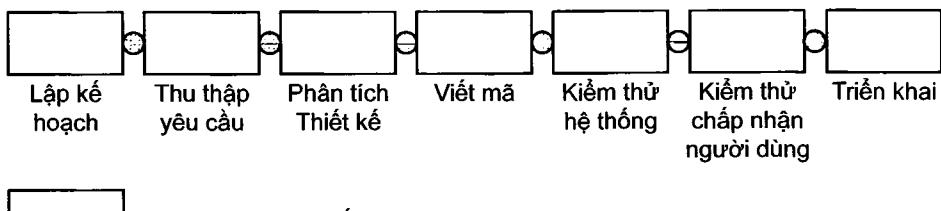
Tại sao Agile và Scrum mang lại hiệu quả trong quản lý dự án phần mềm?

việc giảm thiểu mức độ rủi ro hoặc sự bấp bênh tới mức 0 hoặc ít nhất có thể.

Trong khi chúng ta có tới bốn cách để đối phó với rủi ro (né tránh, chuyển nhượng, chấp nhận và giảm thiểu), thông thường người quản lý dự án phải chọn cách cuối cùng là giảm thiểu rủi ro. Đây chính là chỗ mà Scrum tỏ ra vượt trội với chu kỳ Thanh tra và Thích nghi thường xuyên.

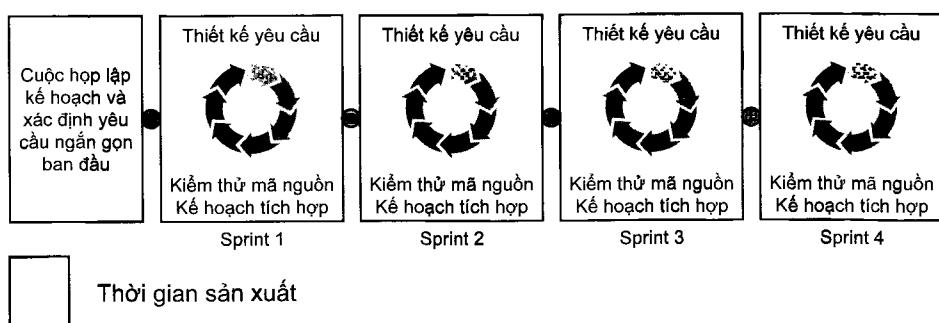
■ Một vòng đời phát triển phần mềm tinh gọn hơn:

Trong Hình 1.5 và 1.6, chúng ta thấy sự khác biệt lớn về mặt thời gian, trong đó một nhóm sử dụng vòng đời dài hơn (Hình 1.5), nhóm còn lại sử dụng vòng đời tinh gọn hơn (Hình 1.6) giúp tận dụng thời gian sản xuất hiệu quả hơn.



Hình 1.5

Dòng giá trị truyền thống.



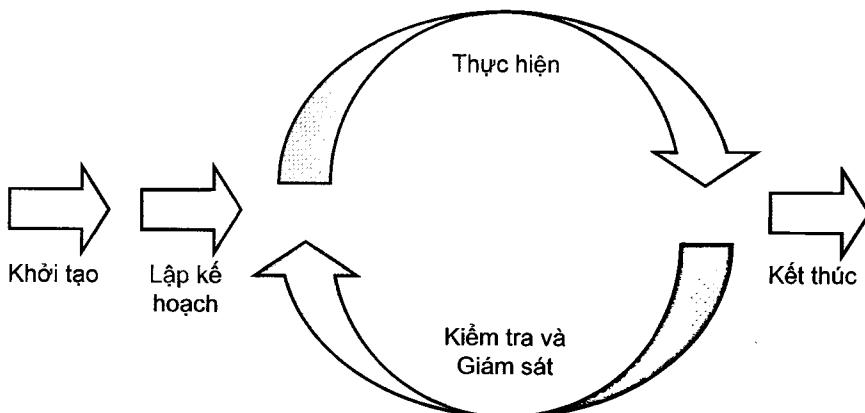
Hình 1.6

Dòng giá trị Scrum.

Chương 1 ▪ Chuẩn bị cho Agile và Scrum

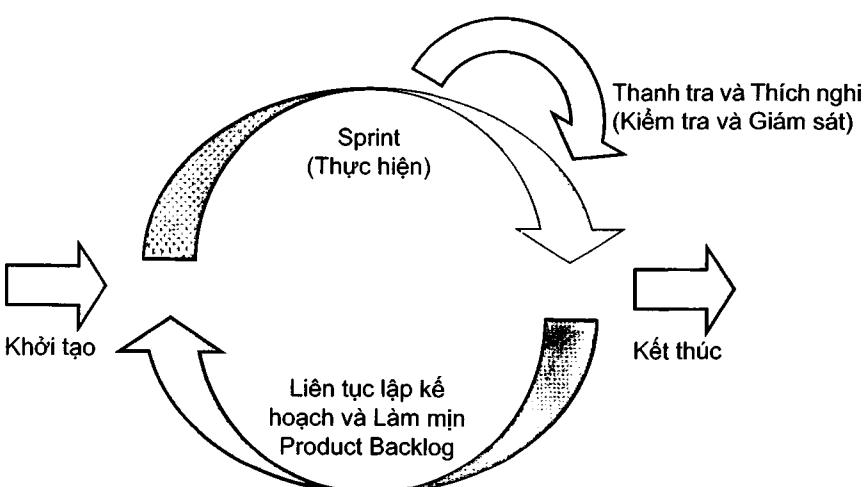
- Quy trình quản lý dự án có khả năng thích ứng hơn. Không giống như quy trình tuần tự trong môi trường thác nước (waterfall) coi tính ổn định dự án là nền tảng (Hình 1.7), Scrum sẽ trông giống như Hình 1.8, trong đó chỉ sự thay đổi được coi là bất biến.
- Một khung quy trình phát triển và quản lý dự án dựa trên động lực và niềm tự hào cá nhân. Hơn bất cứ điều gì khác, điều này có thể là một trong những nguyên lý mạnh mẽ nhất của Agile và Scrum. Tâm điểm mới không phải là việc có người quản lý ra lệnh cho các thành viên những gì họ cần phải làm, mà là để cho các nhóm tự quyết định họ sẽ làm thế nào để tự mình hoàn thành công việc của nhóm.

Scrum đề xuất một khung quản lý phần mềm mới, dựa trên sự tự tổ chức của nhóm, động lực, quyền sở hữu và tự hào về thành tích mà họ đạt được.



Hình 1.7

Quy trình quản lý dự án truyền thống.



Hình 1.8

Thích ứng khung quản lý dự án với Scrum.

Mặc dù, phần giới thiệu về Agile và đặc biệt là Scrum khá ngắn gọn nhưng nó cung cấp cho bạn đủ kiến thức để có thể tiếp tục với phần còn lại của cuốn sách.

Nhưng trước khi làm điều này, hãy lưu ý rằng kể từ giờ chúng ta chỉ tập trung vào Scrum, do đó chúng tôi sẽ chỉ đề cập đến Scrum trong phần còn lại của cuốn sách.

TÓM TẮT

Vào năm 2001, Agile đã chào đời khi một nhóm các chuyên gia đã tụ họp tại Utah để soạn thảo ra Tuyên ngôn Agile với trọng tâm nghiêng về phía phát triển phần mềm.

Để bổ sung cho Tuyên ngôn Agile, năm 2005 một nhóm chuyên gia khác đã ngồi lại với nhau để soạn thảo ra một tài liệu khác được gọi là Tuyên ngôn Tương hỗ trong Quản lý dự án (PM Declaration of Interdependence - DOI), trong đó nhấn mạnh về phía quản lý dự án.

Tuyên ngôn Agile cùng với DOI đã hình thành nền tảng của phong trào Agile và của tất cả các quy trình phát triển và quản lý dự án linh hoạt, bao gồm cả Scrum.

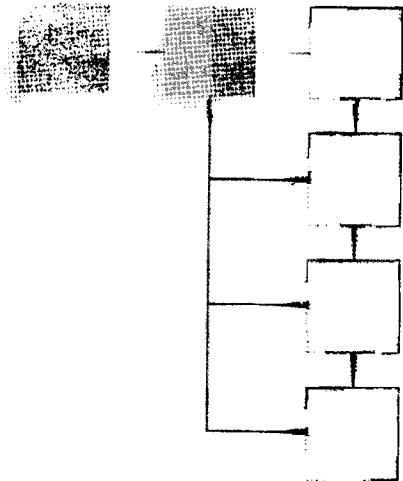
Vì Scrum không thể sử dụng được trong những hoàn cảnh mà không có sự thích ứng, nên việc có kiến thức nền tảng tốt về phong trào và quy trình Agile sẽ giúp bạn biết cách để thích ứng Scrum với môi trường của mình.

Xuyên suốt cuốn sách này, bạn sẽ thấy tất cả những lợi thế của Scrum, bốn trong số đó mang lại giá trị được chỉ ra ngay lập tức: (1) một cơ chế giảm thiểu rủi ro, (2) một quy trình làm phần mềm tinh gọn hơn, (3) quy trình quản lý dự án phần mềm có khả năng thích ứng hơn và (4) một khung làm việc dựa trên một nhóm tự tổ chức, động lực, quyền sở hữu và niềm tự hào.

Vì trọng tâm của cuốn sách là Scrum, chúng tôi sẽ chỉ đề cập đến Scrum từ đây và xuyên suốt cuốn sách.

CHƯƠNG 2

BÀN LUẬN VỀ TÀI CHÍNH



Dù vấn đề tài chính không được nhắc tới nhiều trong các tài liệu Scrum, nhưng sự hiểu biết về tài chính sẽ giúp bạn và nhóm của mình đổi thoại thành công hơn với những người làm kinh doanh.

Sẽ là không đủ nếu bạn chỉ đơn thuần thuyết phục đội kinh doanh và đội quản lý tin tưởng rằng dự án của mình đáng đầu tư và sản phẩm phần mềm sẽ tốt hơn nếu dùng Scrum, mà hơn thế bạn cần một số hiểu biết về kinh doanh.

Để tránh lãng phí thời gian nhiều hơn mức cần thiết cho chủ đề tài chính, chúng tôi đã chọn một số ít những khái niệm và công thức tài chính quan trọng nhằm giúp bạn giành được sự chấp thuận với dự án của mình; bạn cũng có thể sử dụng những thông tin trong chương này để theo dõi tình trạng tài chính của dự án đó.

TÍNH CHI PHÍ DỰ ÁN

Tốc độ của nhóm (team velocity) là một khái niệm cần thiết trong Scrum mặc dù nó không được coi là có giá trị hoặc là một tạo tác. *Tốc độ của nhóm* là tổng số điểm (point) của các User story hoặc hạng mục trong Product Backlog mà nhóm có thể chuyển giao trong một Sprint.

Chúng ta sẽ thảo luận chi tiết hơn về tốc độ của nhóm trong Chương 5, “Tạo ra ước tính điểm story so sánh được để triển khai Scrum trong toàn doanh nghiệp,” nhưng chủ đề này thích hợp khi nhắc tới ở đây bởi vai trò của tốc độ nhóm trong việc tính toán chi phí nguồn nhân lực của dự án.

Để minh họa chúng ta sẽ giả sử tốc độ của nhóm dự án của bạn là 20 điểm trong một Sprint 4 tuần. Do đó, nếu ước lượng dự án của bạn là 160 điểm, thì bạn có thể dự tính rằng nhóm sẽ hoàn thành dự án trong khoảng 8 Sprint hoặc 32 tuần.

Giả sử chi phí cho nhóm dự án là 150.000 USD trong một năm bao gồm cả lương và đãi ngộ. Điều này có nghĩa chi phí cho nhóm với dự án này sẽ là 92.308 USD ($[150.000 \text{ USD} * 32 \text{ tuần}] / 52 \text{ tuần}$).

Chương 2 ▪ Bàn luận về tài chính

Bây giờ nếu thêm các chi phí dự án khác như thiết bị máy tính và viễn thông vào chi phí nhân lực, bạn sẽ có chi phí dự án tổng thể.

Từ đó ta thấy, nếu bạn có thể ước đoán càng sớm tốc độ của nhóm, thì càng dễ ước lượng chính xác hơn chi phí cho nhóm dự án khi dùng Scrum. Nếu không, bạn vẫn sẽ cần sử dụng những kỹ thuật hoặc phương pháp tiếp cận hiện thời ở công ty của mình.

CHỌN DỰ ÁN ĐẦU TƯ

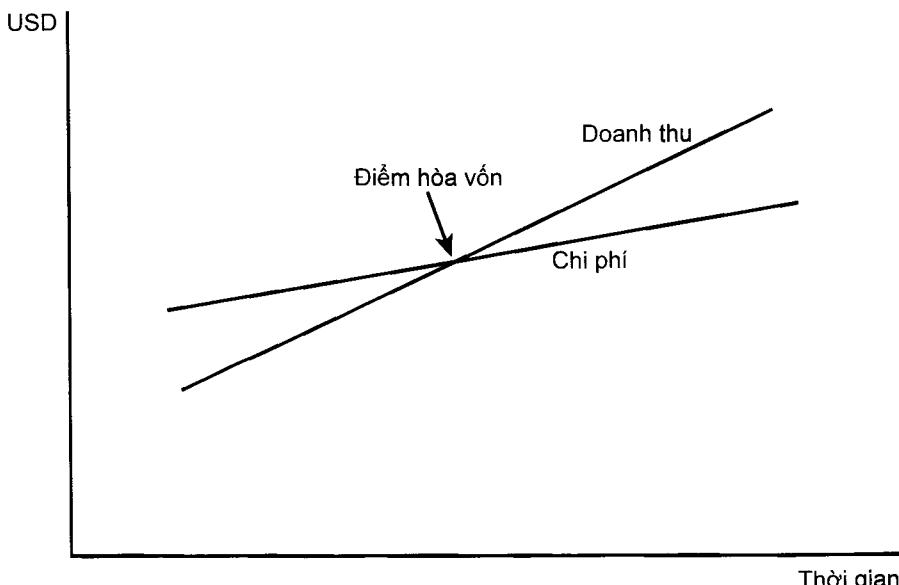
Chúng ta đã thấy cách tính chi phí nhân lực dự án bằng cách dùng tốc độ nhóm, giờ hãy xem cách tính lợi nhuận của dự án trước khi một công ty quyết định có thể đầu tư vào dự án hay không.

Có nhiều cách tính lợi nhuận, nhưng cách thông dụng nhất là Thời gian hoàn vốn (Payback Period), Mua so với Xây dựng (Buy Versus Build), Giá trị hiện tại thuần (Net Present Value - NPV), và Tỷ lệ hoàn vốn đầu tư (Return on Investment - ROI).

Thời gian hoàn vốn

Để thuyết phục những nhà kinh doanh rằng dự án của bạn đáng để họ đầu tư, thì bạn sẽ phải biết thời gian hoàn vốn. Nói một cách đơn giản, *thời gian hoàn vốn* là khoảng thời gian mà một công ty thu hồi lại vốn đầu tư ban đầu; hay còn gọi là *điểm hòa vốn* (break even point).

Kỹ thuật này so sánh chi phí đầu tư (chi phí thiết bị và nhân lực, ...) với dòng tiền hoặc doanh thu dự kiến trên vòng đời của sản phẩm phần mềm mới, như ở Hình 2.1



Hình 2.1

Điểm hòa vốn.

Để minh họa, hãy tưởng tượng rằng dự án của bạn sẽ cần một chi phí đầu tư ban đầu là 200.000 USD. Theo dự báo của bạn, luồng tiền hoặc doanh thu sẽ khoảng 28.572 USD mỗi quý.

Theo những con số trên thì sẽ cần 7 quý ($200.000 \text{ USD} / 28.572 \text{ USD} = 7$) hoặc 21 tháng để dự án hòa vốn.

Mặc dù kỹ thuật này không tính đến giá trị của tiền theo thời gian, nhưng nhiều công ty vẫn dùng khá thường xuyên, đặc biệt cho việc phát triển phần mềm nội bộ.

Mua so với Xây dựng

Nếu chúng ta thực hiện theo đúng trình tự dưới đây thì kỹ thuật này khá đơn giản.

Đầu tiên, chúng ta phải tính toán chênh lệch của giá cố định để xây dựng và giá cố định để mua, giá trị này gọi là *phần chênh lệch giá* (price difference).

Tiếp theo, chúng ta phải tính toán chênh lệch phí hằng tháng giữa phí mua hằng tháng và phí xây dựng hằng tháng.

Cuối cùng, chúng ta sẽ tính lượng thời gian bằng cách chia phần chênh lệch giá cho phần chênh lệch phí hằng tháng.

Phần chênh lệch giá / Phần chênh lệch phí hằng tháng = Số tháng.

Cụ thể hơn chúng ta có ba kịch bản:

1. Cả chi phí xây dựng cố định và phí xây dựng hằng tháng đều lớn hơn chi phí mua cố định và phí mua hằng tháng. Trong trường hợp này, lựa chọn hợp lý của công ty là mua.
 - a. Chi phí xây dựng cố định cao hơn chi phí mua cố định và phí mua hằng tháng cao hơn phí xây dựng hằng tháng. Trong trường hợp này, công ty nên mua nếu định giữ sản phẩm phần mềm ít hơn số tháng tính được.
 - b. Chi phí mua cố định cao hơn chi phí xây dựng cố định và phí xây dựng hằng tháng cao hơn phí mua hằng tháng. Trong trường hợp này, công ty nên xây dựng nếu định giữ sản phẩm phần mềm ít hơn số tháng tính được.
2. Cả chi phí mua cố định và phí mua hằng tháng đều lớn hơn chi phí xây dựng cố định và phí xây dựng hằng tháng. Trong trường hợp này, lựa chọn hợp lý cho công ty là xây dựng.
3. Kịch bản thứ ba phức tạp hơn và sẽ cần thêm vài tính toán:
 - a. Chi phí xây dựng cố định cao hơn chi phí mua cố định và phí mua hằng tháng cao hơn phí xây dựng hằng tháng. Trong trường hợp này, công ty nên mua nếu định giữ sản phẩm phần mềm ít hơn số tháng tính được.
 - b. Chi phí mua cố định cao hơn chi phí xây dựng cố định và phí xây dựng hằng tháng cao hơn phí mua hằng tháng. Trong trường hợp này, công ty nên xây dựng nếu định giữ sản phẩm phần mềm ít hơn số tháng tính được.

Để minh họa cho kịch bản thứ ba, chúng ta giả định có hai dự án A và B.

Dự án A có chi phí xây dựng cố định là 1.000.000 USD và phí xây dựng hằng tháng là 50.000 USD.

Dự án B có chi phí mua cố định là 900.000 USD và phí mua hằng tháng là 100.000 USD.

Chương 2 ▪ Bàn luận về tài chính

Nếu lắp những con số này vào các công thức ở trên, chúng ta sẽ có:

$$\text{Phản chênh lệch giá} = \text{Chi phí xây dựng cố định} - \text{Chi phí mua cố định} = 1.000.000 \text{ USD} - 900.000 \text{ USD} = 100.000 \text{ USD}$$

$$\text{Phản chênh lệch phí hằng tháng} = \text{Phí mua hằng tháng} - \text{Phí xây dựng hằng tháng} = 10.000 \text{ USD} - 50.000 \text{ USD} = 50.000 \text{ USD}$$

Bây giờ, nếu chia 100.000 USD cho 50.000 USD, chúng ta sẽ được 2 tháng, đây là điểm mà cả lựa chọn xây dựng và mua sẽ có cùng chi phí (chi phí cố định + tổng phí hằng tháng).

Do đó kết luận là, nếu định giữ phản mềm ít hơn 2 tháng thì chúng ta nên mua bởi tổng chi phí mua (900.000 USD + 100.000 USD = 1.000.000 USD) sẽ nhỏ hơn tổng chi phí xây dựng (1.000.000 USD + 50.000 USD = 1.050.00 USD). Trong trường hợp ngược lại, nếu dự định giữ phản mềm nhiều hơn 2 tháng thì chúng ta nên xây dựng bởi tổng chi phí xây dựng (1.000.000 USD + 50.000 USD + 50.000 USD + 50.000 USD = 1.150.000 USD) sẽ ít hơn tổng chi phí mua (900.000 USD + 100.000 USD + 100.000 USD + 100.000 USD = 1.200.000 USD). Thông thường, các công ty giữ phản mềm lâu hơn hai tháng, nhưng đây chỉ là một ví dụ minh họa.

Giá trị hiện tại thuần (NPV)

Chúng ta hãy nói về Giá trị Hiện tại (Present Value - PV) trước khi thảo luận về NPV và ROI (Tỷ lệ hoàn vốn đầu tư) như một cách để lựa chọn dự án đầu tư.

Giá trị hiện tại (PV) là một kỹ thuật phức tạp hơn kỹ thuật hoàn vốn trước đó bởi kỹ thuật này tính đến cả giá trị của đồng tiền theo thời gian. Điều này là quan trọng bởi tiền mà bạn nhận được ở tương lai lại thường ít giá trị hơn so với cùng số tiền đó mà bạn nhận được hôm nay.

Công thức là $PV = FV / (1 + i)^n$

FV là Giá trị tương lai (Future Value – giá trị của một số tiền trong tương lai), PV là Giá trị hiện tại (Present Value), i là lãi suất hoặc tỷ lệ lạm phát, và n là lượng thời gian mà lãi suất được trả.

Để minh họa, chúng ta hãy giả sử rằng bạn có 4.000 USD và muốn biết lượng tiền này đáng giá bao nhiêu trong ba năm nữa khi lãi suất hoặc tỷ lệ lạm phát là 5 %.

Sử dụng công thức này, bạn sẽ có FV bằng 4.630 USD bởi nó là tích của 4.000 USD với $(1,05)^3$ hoặc tích của 4.000 USD với 1,157625).

Nói cách khác, PV (Giá trị hiện tại) của 4.630,50 USD trong tương lai tương đương với 4.000 USD hôm nay.

Do đó, nếu bạn phải chọn giữa hai dự án có hai giá trị trong tương lai, thì hãy chọn dự án có PV cao hơn.

Để áp dụng kỹ thuật này vào quy trình chọn dự án, giả sử bạn phải chọn giữa dự án A và B. Dự án A có PV bằng 70.000 USD và dự án B có PV bằng 80.000 USD. Bằng

cách áp dụng quy tắc PV và thuần túy dựa trên góc nhìn tài chính, bạn nên chọn dự án B bởi nó có PV cao hơn dự án A.

Tương tự như PV (giá trị hiện tại của dòng tiền hoặc doanh thu trong tương lai), Giá trị hiện tại thuần (NPV) là giá trị hiện tại của tổng doanh thu trừ đi giá trị hiện tại của chi phí đầu tư trong một khoảng thời gian.

Khi kết thúc việc tính toán, bạn sẽ có ba tình huống để lựa chọn:

- Nếu NPV của một dự án lớn hơn không, dự án nên được chấp nhận.
- Nếu NPV nhỏ hơn không, dự án nên bị bác bỏ.
- Nếu có hai hoặc nhiều hơn một dự án, thì nên chọn dự án có NPV cao nhất.

Để minh họa, hãy giả sử chúng ta cần tính NPV của một dự án có lãi suất 10% trong khoảng thời gian ba năm. Với doanh thu ước tính tương ứng là 25.000 USD, 100.000 USD, và 200.000 USD cho các năm thứ nhất, thứ hai, thứ ba và chi phí dự án hằng năm tính được là 100.000 USD cho ba năm tiếp theo, như Hình 2.2, chúng ta sẽ có:

Chênh lệch NPV = NPV doanh thu – NPV chi phí dự án

Chênh lệch NPV = 252.046 USD – 281.654 USD = -29.608 USD

Bởi NPV là một số âm, nhà quản lý kinh doanh không nên đầu tư vào dự án này.

Khoảng thời gian	Doanh thu	NPV Doanh thu (Lãi suất 10%) PV = FV/(1+i) ⁿ	Chi phí dự án	NPV Chi phí dự án (Lãi suất 10%) PV = FV/(1+i) ⁿ	Chênh lệch NPV (Lãi suất 10%)
0	0	0	100.00	100.00	-100.00
1	25.000	18.783	100.00	99.009	-82.226
2	100.00	83.000	100.00	82.645	+335
3	200.00	150.263	0	0	+150.263
		252.046		281.265	-29.608

Hình 2.2

Giá trị hiện tại thuần (NPV).

Tỷ lệ hoàn vốn nội bộ (Internal Rate of Return - IRR), hoặc Tỷ lệ hoàn vốn đầu tư (Return on Investment - ROI)

Việc tính toán IRR hoặc ROI khá phức tạp và sẽ cần một máy vi tính hoặc một máy tính mạnh. Tuy nhiên, có một cách thiết thực mà những CEO thường sử dụng để tính ROI chỉ cần sử dụng hai khái niệm: Tốc độ và tỷ suất lợi nhuận biên (profit margin).

Khái niệm đầu tiên, tốc độ, gần giống với khái niệm tốc độ trong Scrum. Trong kinh doanh, tốc độ xác định mức độ mà một công ty nhanh chóng tạo ra đủ doanh thu (hoặc giảm chi phí) để trả chi phí đầu tư (hoặc tài sản hoặc hàng tồn kho), cộng với lợi nhuận mà công ty hy vọng đạt được.

Chương 2 • Bàn luận về tài chính

Khái niệm thứ hai, tỷ suất lợi nhuận biên, là số tiền, hoặc lợi nhuận, mà công ty tạo ra sau khi trả hết mọi phí tổn, chi phí liên quan tới việc tạo và bán sản phẩm hoặc dịch vụ cũng như thuế và lãi suất vay.

Để tính ROI trong dự án, bạn sẽ chỉ cần biết tốc độ của dự án và tỷ suất lợi nhuận biên của công ty. Phòng tài chính phải tính toán giá trị ROI cho công ty và sẽ cung cấp nếu bạn yêu cầu.

ROI = Tốc độ x Tỷ suất lợi nhuận biên

Giả sử rằng dự án của bạn cần mang về doanh thu 4 triệu USD với chi phí đầu tư 2 triệu USD, và nếu tỷ suất lợi nhuận biên của công ty bạn là 10%, thì sử dụng công thức trên cho ROI của dự án này là:

$$\text{ROI} = (4.000.000/2.000.000) \times 10\%$$

$$\text{ROI} = 20\%$$

Khi phải so sánh những dự án khác nhau thì hãy nhớ chọn dự án có ROI cao nhất.

THEO DÕI HIỆU SUẤT DỰ ÁN

Người ta dùng Giá trị thu được (Earned Value - EV) để đo hiệu suất công việc của một nhóm dự án so với kế hoạch, nhằm xác định những chênh lệch, nguy cơ lạc hướng về cả hai khía cạnh tiến độ và chi phí.

EV là giá trị được tính toán dựa trên khối lượng công việc đã hoàn thành tính tới thời điểm hiện tại và ngân sách được phê duyệt để thực hiện những công việc đó. Do vậy, nếu một dự án có ngân sách là 100.000 USD và đã có 30% công việc được hoàn thành thì EV của nó là 30.000 USD.

Nhằm phục vụ cho những mục đích thực tế của cuốn sách, chúng ta sẽ xem xét ba công thức sau đây, chúng đều là những thứ được nhóm dự án quan tâm.

1. Công thức dành cho Chi phí thực hiện (Cost Performance):

Chênh lệch chi phí (CV) = Giá trị thu được (EV) – Chi phí thực tế (AC)

Chỉ số chi phí thực hiện (CPI) = Giá trị thu được (EV)/Chi phí thực tế (AC)

2. Công thức dành cho Tiết kiệm Thực hiện:

Chênh lệch tiết kiệm (SV, Chênh lệch chi phí do thay đổi tiến độ) = Giá trị thu được (EV) – Giá trị dự kiến (PV)

Chỉ số tiết kiệm thực hiện (SPI)= Giá trị thu được (EV)/Giá trị Dự kiến (PV)

3. Dự toán Ngân sách Dự án:

Dự toán lúc hoàn thành (EAC) = Ngân sách lúc hoàn thành (BAC)/Chỉ số Chi phí thực hiện (CPI)

Dự toán để hoàn thành (ETC) = Dự toán lúc hoàn thành (EAC) – Chi phí thực tế (AC)

Chênh lệch lúc Hoàn thành (VAC) = Ngân sách lúc hoàn thành (BAC) – Dự toán lúc hoàn thành (EAC)

Chi phí Thực hiện

Mục đích của chi phí thực hiện (cost performance) chủ yếu là để tính hiệu suất dự án về mặt hiệu quả sử dụng chi phí. Giá trị này còn có thể được sử dụng để tiên đoán các xu hướng về chi phí thực hiện trong tương lai.

Chênh lệch Chi phí (Cost Variance - CV)

Chênh lệch chi phí là phần chênh lệch giữa giá trị của công việc đã hoàn thành (EV) với chi phí thực tế (Actual Cost - AC) của dự án.

Nếu bạn đang thắc mắc AC là gì, thì nó chính là chi phí thực tế, hoặc là lượng tiền thực tế đã chi cho dự án tính đến nay. Ví dụ, nếu một dự án có ngân sách là 100.000 USD, trong đó 40.000 USD đã được chi trả cho dự án đến thời điểm hiện tại thì AC của dự án đó là 40.000 USD.

Biết EV và AC sẽ tính được các chênh lệch, tiến độ, và các chỉ số thực hiện của dự án, từ đó cho phép quản lý dự án xác định những chương trình hành động mới hoặc là xét xem có nên tiếp tục dự án.

Giá trị CV dương nói lên rằng dự án không vượt ngân sách, còn với một giá trị chênh lệch chi phí âm thì có nghĩa là dự án đang vượt ngân sách.

Chênh lệch chi phí (CV) = Giá trị thu được (EV) – Chi phí thực tế (AC)

Chỉ số chi phí thực hiện (CPI)

Trong khi mức chênh lệch là một con số thực tế (âm hoặc dương) chẳng hạn là 25.000 USD, thì một chỉ số, như Chỉ số chi phí thực hiện (Cost Performance Index - CPI) chẳng hạn, lại là một tỷ lệ phải nằm trong khoảng từ 0 đến 2.

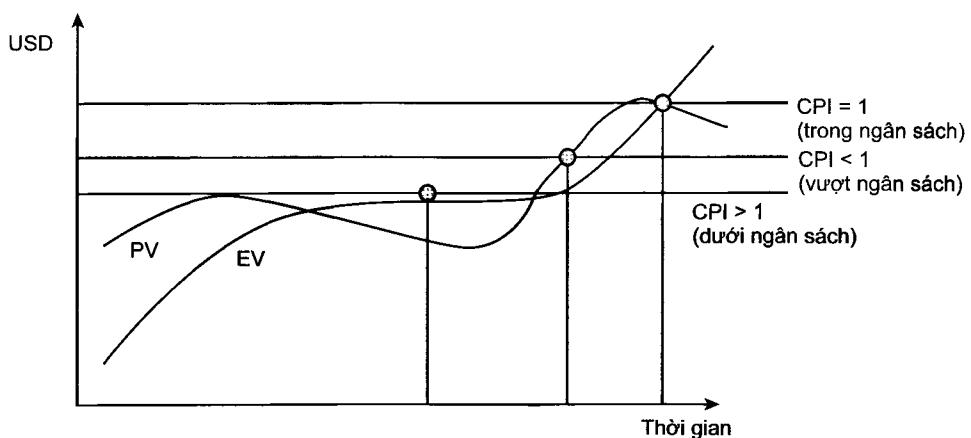
Công thức để tính Chỉ số chi phí thực hiện (CPI) như sau.

Chỉ số chi phí thực hiện (CPI) = Giá trị thu được (EV)/Chi phí thực tế (AC)

CPI lớn hơn 1 chỉ ra rằng chúng ta thu được nhiều giá trị hơn từ công việc đã làm so với chi phí phát sinh. Chỉ số CPI dưới 1 nghĩa là giá trị công việc của dự án đã làm là ít hơn chi phí phát sinh. Nói cách khác, nhóm dự án đang đốt tiền nhanh hơn là tạo ra giá trị, và do đó có thể bị vượt quá ngân sách. Biểu đồ ở Hình 2.3 chỉ ra sự biến đổi của CPI khi thay đổi EV và AC.

Vậy là, CPI bằng 0,90 có nghĩa là dự án đang vượt quá ngân sách.

Chương 2 • Bàn luận về tài chính



Hình 2.3

Chỉ số Chi phí Thực hiện (CPI).

Tiến độ thực hiện

Tiến độ thực hiện (Schedule Performance) chủ yếu là để tính hiệu suất của dự án về mặt tiến độ. Giá trị này cũng có thể được dùng để dự đoán các xu hướng về tiến độ thực hiện.

Chênh lệch Tiến độ (SV)

Trong khi AC là nói đến chi phí, Chênh lệch tiến độ (Schedule Variance - SV) lại chỉ cho chúng ta biết dự án đang bị chậm hay nhanh hơn tiến độ.

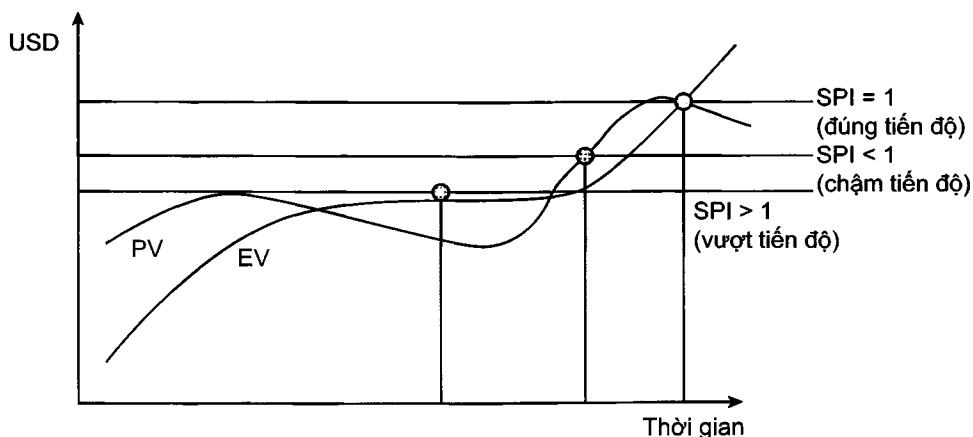
Chênh lệch tiến độ (SV) = Giá trị thu được (EV) – Giá trị dự kiến (PV)

Giá trị Dự kiến (PV) là ngân sách phê duyệt để thực hiện những công việc đã được lập kế hoạch hoàn thành vào thời điểm nào đó. Ví dụ, nếu một dự án có ngân sách là 500.000 USD, và được dự trù rằng 50% dự án sẽ hoàn thành trong một thời gian xác định, vậy thì vào ngày đó PV sẽ là 250.000 USD.

Chênh lệch tiến độ âm có nghĩa là dự án đang bị chậm tiến độ, trong khi đó, Chênh lệch tiến độ dương biểu thị rằng dự án đang vượt tiến độ.

Xét một ví dụ minh họa, hãy giả định rằng thay vì hoàn thành 50% như đã đề ra, nhóm chỉ đạt được 40% công việc, trong trường hợp này EV sẽ bằng 200.000 USD.

Kết quả SV sẽ bằng -50.000 USD (tức là bằng 200.000 USD – 250.000 USD), điều này có nghĩa là dự án đang chậm tiến độ.

**Hình 2.4**

Chỉ số Tiến độ thực hiện.

Chỉ số tiến độ thực hiện (SPI)

Chỉ số tiến độ thực hiện (Schedule Performance Index - SPI) giống với CPI về mặt nguyên tắc, nhưng CPI theo dõi hiệu suất chi phí, còn SPI theo dõi tiến độ dự án.

Công thức để tính Chỉ số tiến độ thực hiện (SPI) như sau.

Chỉ số Tiến độ thực hiện (SPI) = Giá trị thu được (EV)/Giá trị Dự kiến(PV)

Cũng giống như CPI, SPI là tỷ lệ, nó phải nằm trong khoảng từ 0 đến 2 như được thể hiện trong Hình 2.4.

SPI lớn hơn 1 có nghĩa là hiệu suất của nhóm tốt hơn dự toán, còn SPI dưới 1 chỉ ra rằng nhóm dự án đang chậm tiến độ.

Do vậy, SPI bằng 0,70 nói lên rằng nhóm dự án đang đi lệch mục tiêu.

Dự toán Ngân sách dự án

Ý tưởng đằng sau việc dự toán ngân sách dự án là sử dụng Giá trị thu được (EV) để dự toán chi phí thực tế của dự án khi nó được hoàn thành.

Nhằm mang lại những trợ giúp quý giá cho nhóm dự án, chúng tôi sẽ đánh giá ba trong số những giá trị đó ở những trang tiếp theo, chúng gồm EAC (Estimation at Completion – Dự toán tại lúc hoàn thành dự án, gọi tắt là Dự toán lúc hoàn thành), ETC (Estimate to Complete – Dự toán để hoàn thành nốt dự án, gọi tắt là Dự toán để hoàn thành), và VAC (Variance at Completion – Chênh lệch lúc hoàn thành).

Dự toán lúc hoàn thành (EAC)

Nếu biết được Chỉ số chi phí thực hiện (CPI), thì bạn cũng có khả năng dự toán tổng chi phí của dự án là bao nhiêu khi hoàn thành. Công thức là

Chương 2 • Bàn luận về tài chính

EAC = BAC/CPI (trong đó BAC là ngân sách ban đầu)

Do vậy, nếu BAC = 200.000 USD và CPI = 0,85, thì EAC = 200.000 USD/0,8 = 235.294 USD.

Điều này có nghĩa là ban đầu bạn yêu cầu 200.000 USD, nhưng dựa vào trạng thái hiện tại của dự án thì có vẻ như chi phí của dự án sẽ là 235.294 USD, hay là cần thêm 35.294 USD nữa so với dự toán ban đầu.

Dự toán để hoàn thành (ETC)

Dự toán để Hoàn thành (ETC) là tổng số tiền cần để hoàn thành những công việc còn lại của dự án, điều này cũng được thể hiện bởi biểu đồ Burndown của Scrum. Công thức tính là:

$$ETC = EAC - AC$$

Giả định rằng với EAC = 500.000 USD, đây chính là tổng số tiền bạn chi cho dự án này. Bây giờ, nếu AC (chi phí thực tế mà bạn đã bỏ ra) bằng 450.000 USD, thì ETC sẽ là 50.000 USD (ETC = 500.000 USD – 450.000 USD), đây là chi phí để hoàn thành những công việc còn lại.

Chênh lệch lúc Hoàn thành (VAC)

Chênh lệch lúc Hoàn thành (Variance at Completion - VAC) tính toán sự chênh lệch giữa ngân sách dự toán ban đầu (được thể hiện bởi BAC) với dự toán lúc hoàn thành (EAC). Công thức là:

$$VAC = BAC - EAC$$

Do vậy, nếu BAC = 25.000 USD và EAC = 13.400 USD thì VAC = 25.000 – 13.400 = 11.600 USD

Điều này có nghĩa là một ai đó phải bỏ ra 11.600 USD để trả cho khoản chênh lệch này.

TÓM TẮT

Không giống như nhiều cuốn sách về Agile hoặc Scrum mà hầu như không nhắc đến tài chính, chúng tôi tin rằng việc biết những kiến thức cơ bản về tài chính có thể tăng thêm cơ hội đổi thoại thành công với cấp quản lý kinh doanh.

Để không biến cuốn sách này trở thành một chuyên luận về tài chính, chúng tôi đã lựa chọn đề cập trong chương này chỉ một số khái niệm và công thức hữu ích có liên quan đến tài chính của dự án. Một số để chứng minh cho việc đầu tư vào dự án, một số nhằm tính chi phí của dự án, và một số được dùng trong quá trình thực hiện dự án để ước tính lượng tiền cần phải đầu tư thêm để hoàn thành công việc.

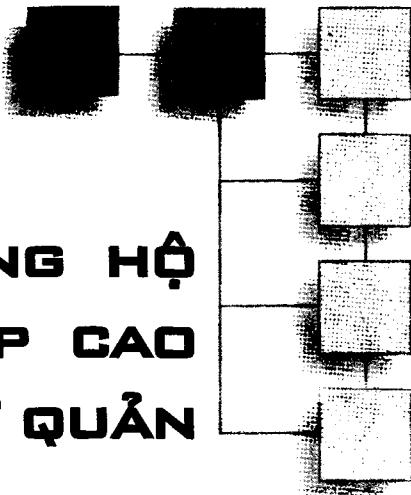
Với tư cách là những chuyên gia phần mềm, phần lớn chúng ta không cần phải trở thành chuyên gia tài chính, nhưng chúng tôi tin rằng ít nhất bạn sẽ cần đến những kiến thức cơ bản được đề cập ở đây. Việc nắm vững một số trong những khái niệm và công thức cơ bản này sẽ giúp bạn giao tiếp tốt hơn với cấp quản lý kinh doanh.

Một vài điểm mấu chốt phải nhớ về EV là:

- Khi nói đến chênh lệch, chính là EV trừ đi AC (chi phí) hoặc PV (tiền độ).
- Khi nói đến chỉ số, nó là EV chia cho AC (chi phí) hoặc PV (tiền độ).
- Chênh lệch âm là không tốt, nhưng nếu chênh lệch dương lại là điều tốt.
- Chỉ số nhỏ hơn 1 là không tốt, nhưng khi chỉ số lớn hơn 1 lại là điều tốt.

CHƯƠNG 3

ĐẢM BẢO SỰ ỦNG HỘ TỪ QUẢN LÝ CẤP CAO VÀ SỰ HỖ TRỢ TỪ QUẢN LÝ CẤP TRUNG

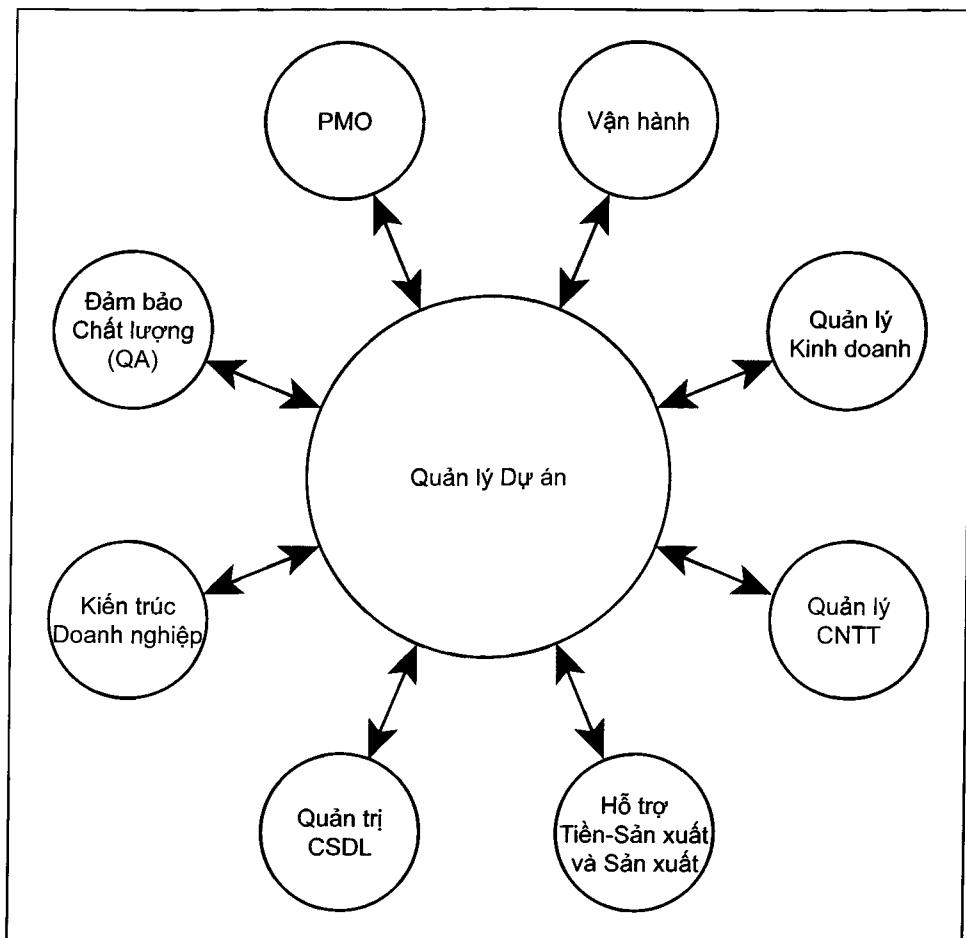


Trừ khi bạn đang làm việc độc lập hoặc trong một công ty nhỏ mới khởi nghiệp, nếu không người quản lý dự án sẽ cần phải quan tâm tới nhiều người bên trong một tổ chức, để đưa dự án về đích (xem Hình 3.1).

Mặc dù trong Scrum không có quản lý dự án, điều đó không có nghĩa là việc quản lý dự án đã bị loại bỏ. Đối với những người đã có kinh nghiệm với Scrum thì họ đều biết rằng nhiệm vụ quản lý dự án trong Scrum chỉ đơn giản bị thay đổi và chia sẻ giữa Product Owner, Nhóm (Phát triển) và ScrumMaster. Bạn có thể tìm hiểu thêm về điều này ở trang 15 trong cuốn sách chuyên đề *Agile Project Management with Scrum* (tạm dịch: Quản lý dự án linh hoạt với Scrum) của Ken Schwaber.

Trong cuốn sách của mình, Ken đã nói rằng thay vì chỉ có một mình người quản lý dự án (hoặc ScrumMaster) chịu trách nhiệm về toàn bộ tương tác với bên ngoài giống như trong môi trường mệnh lệnh và kiểm soát kiểu cũ, giờ đây trách nhiệm quản lý dự án được chia cho Nhóm Phát triển, Product Owner và Scrum Master như được thể hiện trong Hình 3.2.

Chúng ta sẽ bàn về trách nhiệm cụ thể của Scrum Master và Product Owner trong việc lãnh đạo và quản lý dự án Scrum ở chương sau. Chương này chủ yếu giúp bạn nâng cao nhận thức về những việc cần làm khi trao đổi với những cá nhân/bộ phận còn lại của công ty, đặc biệt là khi Scrum vẫn còn là điều mới mẻ, hoặc chưa được lựa chọn để áp dụng cho toàn doanh nghiệp trên quy mô lớn.



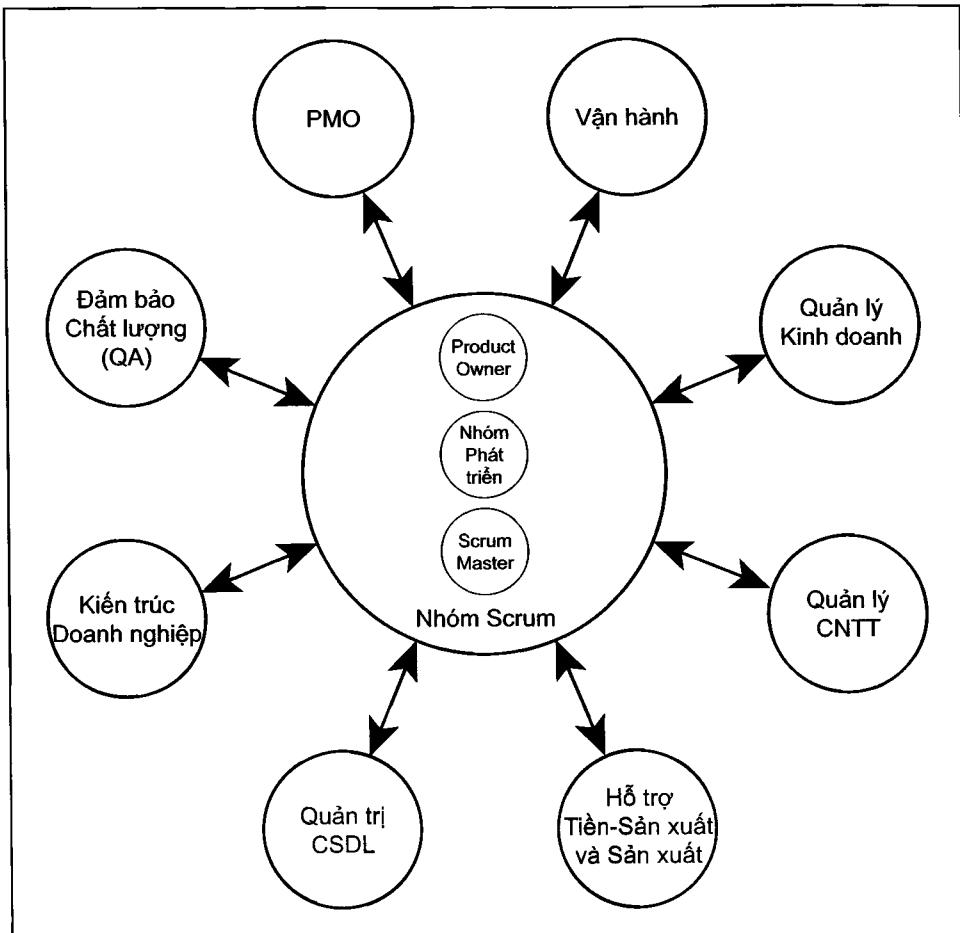
Hình 3.1

Vai trò truyền thống của người quản lý dự án trong công ty.

Như đã trao đổi ở Chương 2, chúng ta cần phải nói chuyện với quản lý kinh doanh hoặc quản lý CNTT cấp cao bằng những nội dung mà họ có thể hiểu được (ngôn ngữ tài chính), bởi vì họ phải bật đèn xanh trước khi bạn có thể làm bất cứ thứ gì khác.

LÀM VIỆC VỚI QUẢN LÝ KINH DOANH CẤP CAO

Khi tiếp xúc với quản lý kinh doanh, chẳng hạn như CEO, Chủ tịch, CFO, hoặc Phó Chủ tịch cấp cao của một đơn vị kinh doanh, chúng tôi khuyên bạn tránh nói về việc Scrum sẽ giúp bạn xây dựng phần mềm tốt hơn và nhanh hơn như thế nào.

**Hình 3.2**

Sự thay đổi về vai trò quản lý dự án trong Scrum.

Là những người điều hành kinh doanh, họ sẽ tỏ ra lịch sự nhưng cuối cùng cũng khước từ bạn và họ tự hỏi tại sao ai đó lại thuê bạn vào đây - một người lãng phí thời gian để nói về những thứ chung chung, thay vì trình bày ý tưởng bằng những thuật ngữ tài chính dễ hiểu hơn.

Cho dù việc xây dựng phần mềm tốt hơn hay việc cải tiến năng suất của lập trình viên đều quan trọng đối với các nhà lãnh đạo, nhưng họ vẫn quan tâm nhiều hơn đến thị trường chung của công ty, tiết kiệm ở mức vĩ mô, hoặc là tỷ suất lợi nhuận biên tổng thể.

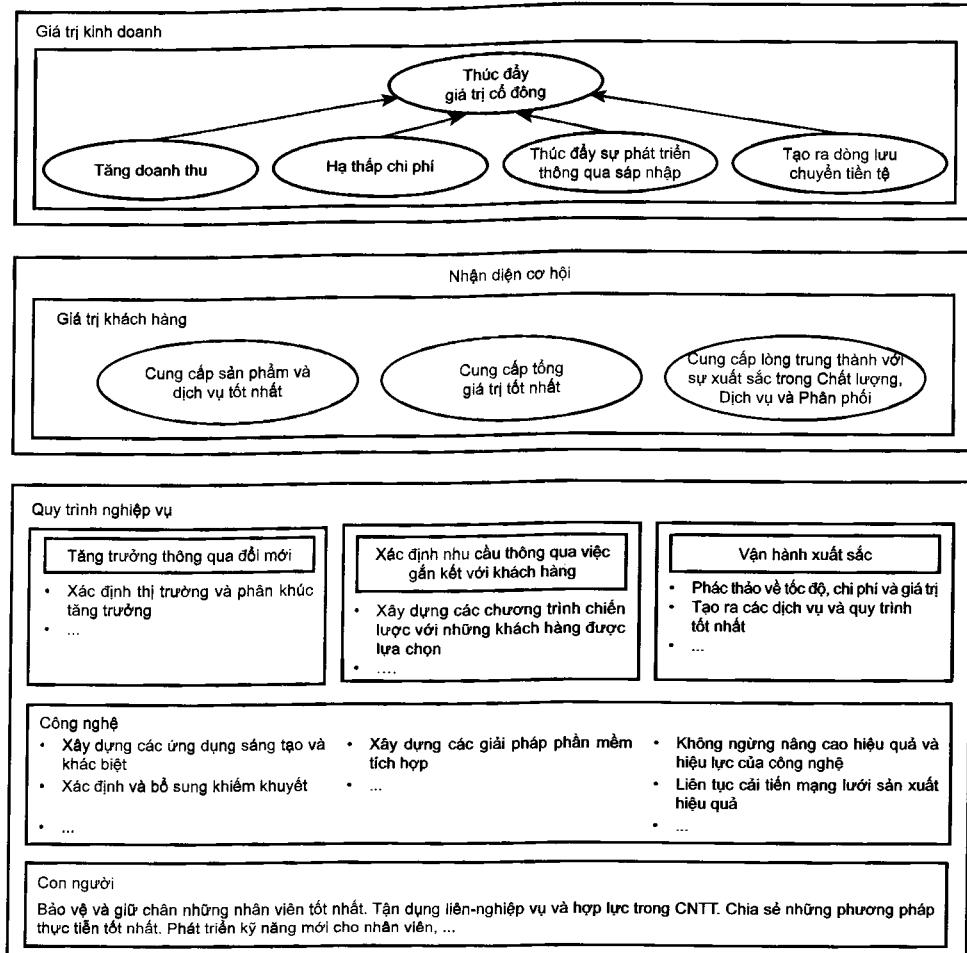
Bởi vậy, cho dù bạn rất phấn khích với Scrum nhưng đừng nên lấy đó làm chủ đề để nói chuyện với quản lý kinh doanh cấp cao về việc phần mềm của bạn sẽ tuyệt vời như thế nào, hay là việc xây dựng nhanh ra sao chỉ khi họ cho phép bạn sử dụng Scrum.

Thay vào đó, hãy luôn nêu mối liên hệ giữa năng lực kỹ thuật và niềm đam mê của bạn với những vấn đề tài chính bằng những con số, hay ít nhất là với chiến lược kinh doanh tổng thể của họ.

Chương 3 • Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung

Cho dù bạn đang nói chuyện với bất kỳ ai trong tổ chức, hãy suy xét dưới góc nhìn của người nghe và nói về những vấn đề đó.

Nhằm giúp đỡ bạn giao tiếp tốt hơn với những nhà điều hành kinh doanh cấp cao, chúng tôi đã tái lập một ví dụ về thẻ điểm cân bằng (Hình 3.3) cho một công ty giả định, gần giống với một công ty nào đó bạn có thể gặp trong những tình huống thật. Thẻ điểm cân bằng (balance scorecard) là một khái niệm phổ biến trong giới điều hành kinh doanh.



Hình 3.3

Thẻ điểm cân bằng trong quản lý kinh doanh cấp cao.

Thẻ điểm cân bằng của công ty giả định này thể hiện những nội dung mà các nhà quản lý kinh doanh cấp cao xem là mục tiêu kinh doanh và chiến lược trọng tâm của họ.

Trong ví dụ này, mục tiêu của họ là để:

1. Tăng doanh thu
2. Hạ thấp chi phí
3. Thúc đẩy sự phát triển thông qua sáp nhập
4. Tạo ra dòng lưu chuyển tiền tệ

Theo sau đó là chiến lược về giá trị khách hàng và những gì mà họ cho rằng công ty cần phải thực hiện về mặt quy trình nghiệp vụ, dự án CNTT và nhân sự.

Việc biết các dự án CNTT phù hợp như thế nào với tầm nhìn tổng thể và chiến lược của các nhà lãnh đạo doanh nghiệp sẽ giúp bạn giao tiếp hiệu quả hơn với họ về việc dự án sẽ đóng góp như thế nào vào chiến lược tổng thể của công ty. Bạn hãy luôn ghi nhớ một điều vô cùng quan trọng là phải liên hệ sự đóng góp của dự án cho công ty về mặt tài chính, đây chính là ngôn ngữ kinh doanh. Các nhà lãnh đạo sẽ cảm ơn bạn về điều này.

LÀM VIỆC VỚI QUẢN LÝ CNTT CẤP CAO

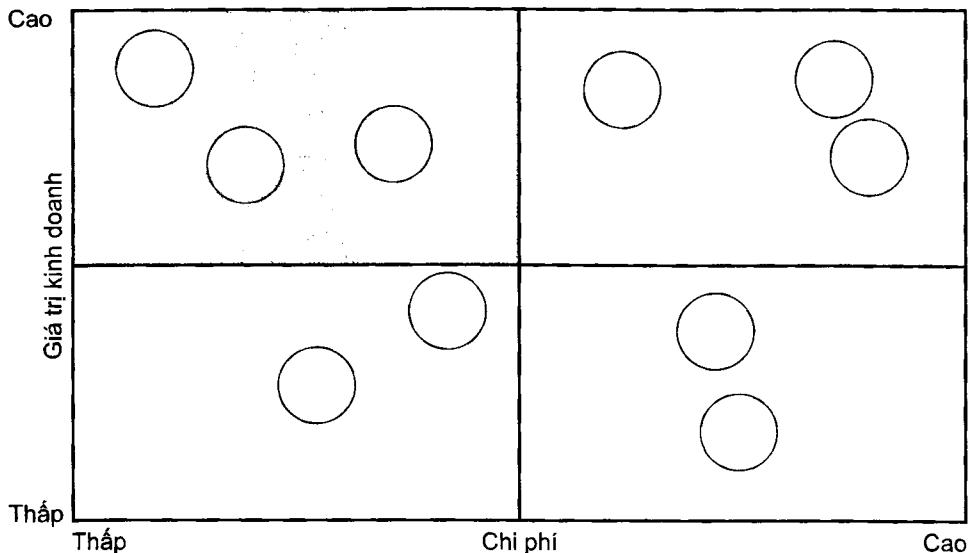
Trong một số công ty, phòng quản lý chương trình (PMO - Program Management Office) không được coi là một phần của bộ phận quản lý CNTT cấp cao. Chúng tôi kể tên phòng này ở đây bởi nó là đại diện chính cho CIO (Chief Infomation Officer - Giám đốc Thông tin), chịu trách nhiệm trong việc giúp phòng CNTT triển khai các dự án CNTT phù hợp với chiến lược kinh doanh, liên tục cải tiến và theo dõi tính hiệu quả của CNTT tại công ty.

Phòng Quản lý Chương trình

Bởi Phòng Quản lý Chương trình (Program Management Office - PMO) là bộ phận chịu trách nhiệm về hiệu năng và sự phù hợp của CNTT với nhu cầu kinh doanh và chiến lược của công ty, thế nên đây là bộ phận đầu tiên bạn nên cố gắng gặp và làm việc với họ.

Nếu hiểu rõ quan tâm của PMO, bạn sẽ không gặp khó khăn để biến PMO thành đồng minh, bởi sứ mệnh của phòng này là đảm bảo các dự án CNTT mang lại giá trị kinh doanh lớn nhất cho doanh nghiệp. Cuối cùng, có phải đó là tất cả những gì chúng ta đã được học về sứ mệnh của Product Owner trong dự án Scrum?

Hình 3.4 là một trong số nhiều ma trận mà PMO phải duy trì. Góc hấp dẫn nhất ở phía trên, bên trái bởi nó có giá trị kinh doanh cao nhất và chi phí thấp nhất. Hãy sử dụng ma trận này để lượng hóa dự án của bạn từ đó có được một cái nhìn thực tế về giá trị kinh doanh thật sự cũng như chi phí tương ứng. Nếu dự án của bạn rơi vào góc phần tư bên trái phía trên với giá trị kinh doanh cao và chi phí thấp, thì dự án có khả năng lớn được chấp thuận.



Hình 3.4

Ma trận ưu tiên hóa kinh doanh CNTT.

Nhiệm vụ khác của PMO là quản trị dự án CNTT. Hình 3.5 cho bạn một ý tưởng về quy trình tổng thể từ khi dự án được thông qua tới cuộc họp đánh giá sau khi triển khai, ở cuộc họp này dự án được đối chiếu với các mục tiêu ban đầu để đảm bảo đạt được mục tiêu đề ra. Do đó, hãy xác minh rằng Product Owner có những tiêu chí và mục tiêu vững chắc đối với công việc bạn làm, để đảm bảo bạn có thể đáp ứng được mong đợi của họ.

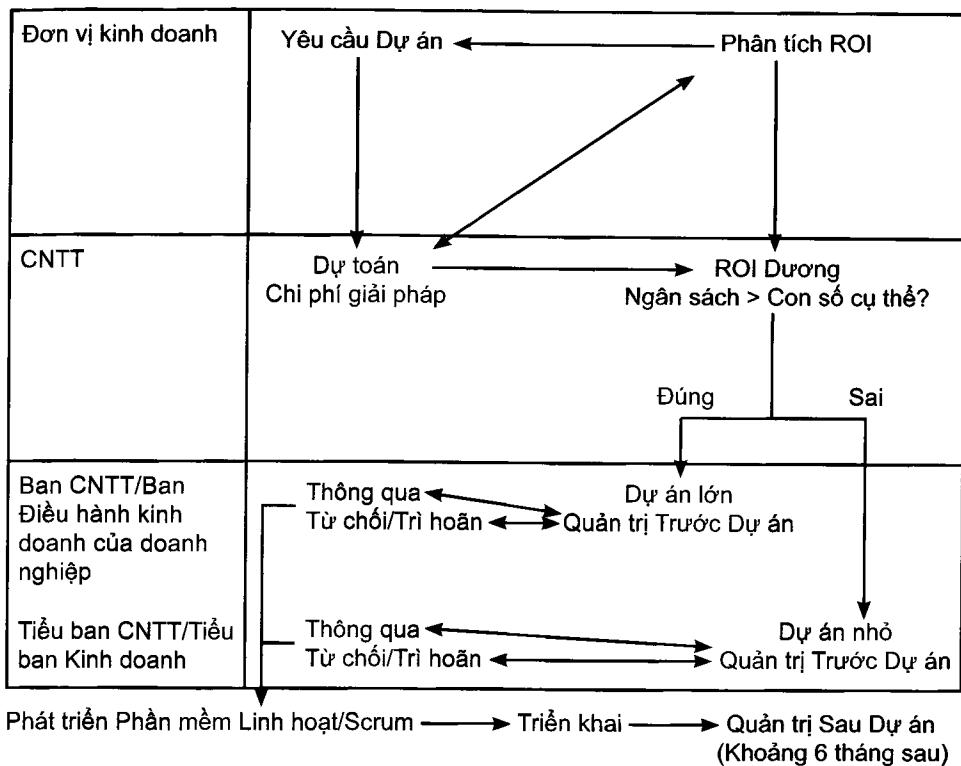
LÀM VIỆC VỚI QUẢN LÝ CNTT CẤP TRUNG

Trước khi đi sâu vào việc thảo luận về chức năng của quản lý cấp trung hoặc quản lý vận hành, chúng ta nên làm rõ rằng dự án của bạn rất ảnh hưởng tới trách nhiệm trong công việc hàng ngày của họ. Do đó, bạn cần sự hỗ trợ của họ để dự án về đích.

Nên nhớ rằng vai trò của quản lý cấp trung không phải là làm xáo trộn dự án mà là giữ cho dự án chạy.

Mỗi quan tâm của họ là giữ cho các vấn đề ở mức tối thiểu và làm mọi thứ đúng thời điểm, giống như một đội chiến thắng.

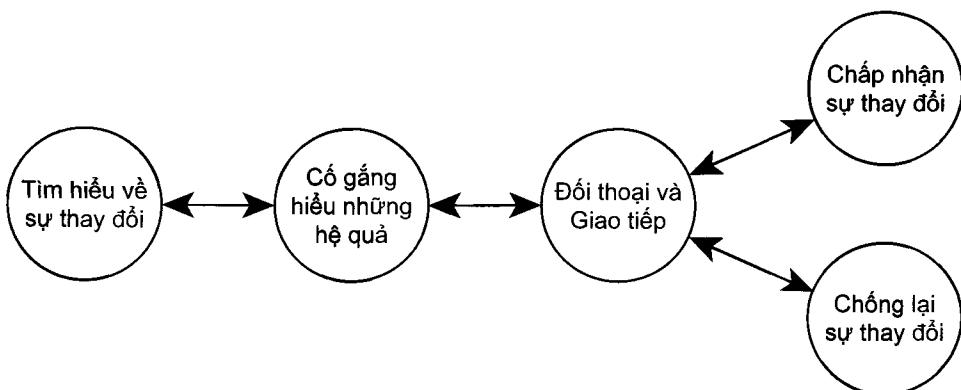
Sự thay đổi bao giờ cũng khó khăn và cần được quản lý hợp lý. Là người mang tới sự thay đổi cùng với dự án của mình, bạn nên biết quá trình mọi người thường trải nghiệm sự thay đổi như thế nào (Hình 3.6). Bạn phải biết cách giao tiếp với quản lý cấp trung và cung cấp đầy đủ lý lẽ để họ nghĩ rằng những thay đổi này đem lại cho họ kết quả tốt.



Hình 3.5

PMO quản trị dự án CNTT.

Khi mọi người nghe thấy sẽ có thay đổi, điều tốt nhất mà bạn có thể mong đợi là họ sẽ cố hiểu là những thay đổi này mang điều gì cho họ. Hy vọng rằng, phản ứng của họ không phải là sự sợ hãi và chống cự.



Hình 3.6

Các giai đoạn của quá trình thay đổi.

Tiếp theo, tùy thuộc vào chất lượng và số lượng giao tiếp và thông tin nhận được, mọi người sẽ có thể phản kháng trực tiếp hoặc gián tiếp, hoặc sẽ chấp nhận sự thay đổi nếu họ nghĩ điều đó tốt cho công việc hoặc sự nghiệp của họ. Do đó, khi giao tiếp, hãy chú ý để đảm bảo đem lại kết quả như bạn mong đợi và những người này sẽ hỗ trợ những gì bạn cần.

Bạn nên lập kế hoạch cho việc liên lạc, giữ liên lạc và giao tiếp với những người sẽ chịu ảnh hưởng bởi những thay đổi mà dự án của bạn đem lại và nên giao tiếp để họ thấy được rõ ràng rằng sự thay đổi này là tốt cho họ.

Đảm bảo Chất lượng

Bởi trong Scrum việc kiểm thử rất quan trọng và có sự thay đổi về bản chất, nên bạn sẽ muốn xây dựng một mối quan hệ tốt với nhóm Đảm bảo chất lượng (Quality Assurance - QA).

Tùy thuộc vào việc công ty bạn đã sẵn sàng để triển khai Scrum hay chưa, chủ yếu là về hạ tầng kiểm thử, mà bạn cần phải làm nhiều hay ít ở phần này. Nếu Scrum đã là một phần của văn hóa, thì bạn có thể chỉ cần biết thủ tục và hiểu để tuân theo và tận dụng chúng.

Nếu Scrum vẫn chưa là một phần của văn hóa công ty, bạn cần giải thích Scrum với người quản lý Đảm bảo chất lượng, đặc biệt cho những người có nhiệm vụ kiểm thử và yêu cầu sự hỗ trợ của họ. Bạn nên đề nghị họ chuẩn bị một vài thành viên trong nhóm của họ để làm việc với dự án Scrum trong suốt vòng đời dự án và/hoặc giúp đỡ thiết lập môi trường kiểm thử phù hợp cho dự án Scrum, những điều này bạn sẽ tìm hiểu thêm ở Chương 9, “Tầm quan trọng của kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp”. Bằng cách này bạn có thể nhanh chóng sẵn sàng cho kiểm thử tự động, hồi quy và tích hợp liên tục.

Quản lý Vận hành

Dù thích hay không, bạn cũng không thể tránh né các nhà Quản lý vận hành. Họ là những người tham gia vào việc điều khiển mọi hoạt động tiền sản xuất và/hoặc sản xuất. Hãy chắc chắn rằng bạn có cơ hội để giải thích với họ điều bạn cần để mã nguồn chuyển giao vào cuối mỗi Sprint sẽ thỏa mãn kỳ vọng của họ. Trước hết, hãy nhớ rằng phòng phát triển và phòng vận hành có mục tiêu khác biệt và đôi khi ngược nhau.

Mục tiêu của bạn, như một phần của tổ chức phát triển, là tạo ra một luồng liên tục các giá trị kinh doanh tốt. Nhưng để làm điều đó bạn mang tới sự thay đổi; trong khi đó mục tiêu của vận hành là tạo ra một luồng giá trị liên tục trong khi vẫn đảm bảo rằng mọi thứ ổn định.

Bản chất chính của Scrum là cung cấp nhiều bản phát hành phần mềm hơn, thường là trong nhiều bản cập nhật nhỏ hơn, ngược lại với lượng những bản cập nhật lớn, mà nhóm vận hành sẽ cần xử lý.

Nhóm vận hành sẽ không và không thể bỏ các thủ tục kiểm soát đối với bạn chỉ vì các bản cập nhật của bạn là nhỏ. Làm như vậy sẽ gia tăng việc gián đoạn trong hệ thống

nghiệp vụ đang vận hành. Nhóm vận hành đơn giản cần phải chặt chẽ hơn bao giờ hết trước khi áp dụng các kiểm thử để thẩm định trước khi chạy thực tế. Toàn bộ quá trình thay đổi phải còn nguyên vẹn, dù các bản cập nhật nhỏ hơn.

Vì vậy, trừ khi công ty bạn đã sẵn sàng cho một số loại triển khai ở chế độ tự phục vụ (self-service deployment) hoặc chính xác hơn là việc tự động hóa quy trình triển khai chế độ tự phục vụ, nếu không hãy thử ứng dụng một số loại lịch trình để sắp xếp các hoạt động bởi nó sẽ giúp họ xử lý các bản cập nhật thường xuyên. Đổi lại, họ sẽ hỗ trợ cũng như cảm ơn bạn vì điều đó.

Theo thời gian, phòng vận hành sẽ thấy có nhiều thời gian hơn để hoàn thành những công việc khác, đồng thời sẽ vui vẻ hỗ trợ sự thay đổi cho phòng phát triển trong khi vẫn giữ được mọi kiểm soát với môi trường sản xuất.

Kiến trúc Doanh nghiệp (EA)

Nếu công ty của bạn có một nhóm gọi là kiến trúc doanh nghiệp (enterprise architecture - EA), thì hãy thử tìm hiểu xem họ ở vị trí nào hoặc mức độ ảnh hưởng của họ trong tổ chức CNTT.

Hãy dành thời gian để hiểu mối quan tâm của nhóm này và làm việc với họ để chỉ cho họ thấy ứng dụng của bạn phù hợp với kiến trúc doanh nghiệp như thế nào.

Hãy tìm ra điều mà họ cần ở bạn, nhưng tránh kéo họ vào quá nhiều cuộc họp trong khi bạn không thể chỉ ra bất kỳ giá trị hữu hình nào.

Nếu nhóm kiến trúc doanh nghiệp thực sự nghiêm túc trong quá trình tiến tới Kiến trúc hướng dịch vụ (Service-Oriented Architecture - SOA), sơ đồ tổng quan Kiến trúc doanh nghiệp (EA) của công ty bạn có thể giống như Hình 3.7.

Trong trường hợp này, hãy chắc chắn là bạn giải thích nhu cầu của mình với nhóm kiến trúc doanh nghiệp một cách kịp thời và yêu cầu họ cung cấp hoặc hỗ trợ cung cấp những dịch vụ khác, từ đó bạn có thể kết nối mọi phần trước khi kết thúc phân đoạn hoặc thời hạn.

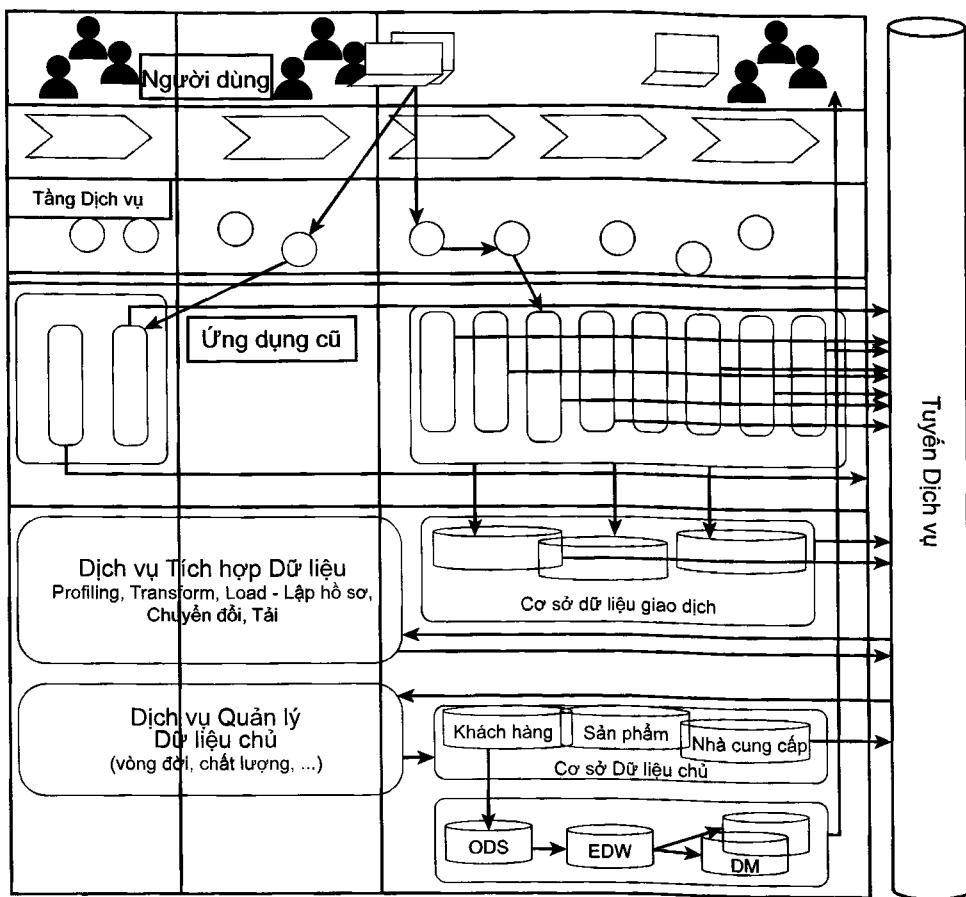
Tuy nhiên, nếu công ty vẫn dùng kiểu kiến trúc truyền thống thì khung làm việc kiến trúc doanh nghiệp sẽ giống như Hình 3.8.

Để nói về mỗi tầng của khung kiến trúc doanh nghiệp ở Hình 3.8 có thể phải hết cả một cuốn sách, nhưng với mục đích của cuốn sách này chúng ta chỉ tập trung vào tầng cuối cùng, tầng kiến trúc dữ liệu.

Dù nền tảng của bạn là kiến trúc dữ liệu hay kiến trúc ứng dụng, thì kiến trúc dữ liệu vẫn là một chủ đề rất quan trọng trong một doanh n

ghiệp. Vấn đề thường gặp là kiến trúc dữ liệu khá lộn xộn có thể làm nhóm phụ trách kho dữ liệu (data warehouse) doanh nghiệp đưa ra những báo cáo không thể tin được.

Chương 3 - Đảm bảo sự ứng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung



Hình 3.7

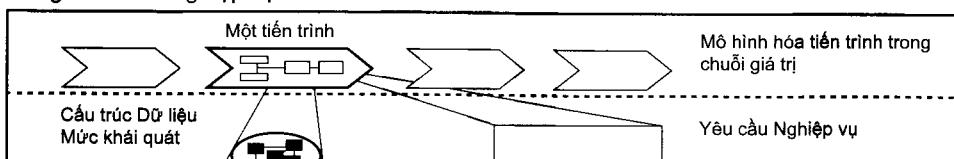
Dự án của bạn và Kiến trúc doanh nghiệp SOA.

Có nhiều ví dụ để minh họa, nhưng ví dụ hay được nhắc tới nhất đó là giá trị trong các báo cáo thường thay đổi từ phòng ban này tới phòng ban khác, ngay cả khi cùng nói đến số liệu bán hàng.

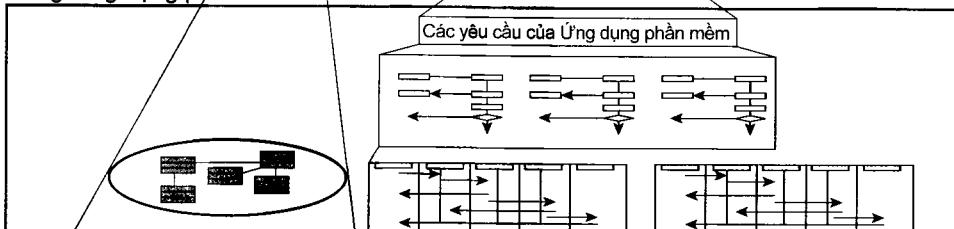
Dựa trên minh họa ở Hình 3.9, không ngạc nhiên rằng các báo cáo thường xuyên đột và là dữ liệu không đáng tin.

Tình trạng này thường gây thất vọng cho những nhà điều hành kinh doanh, đặc biệt là giám đốc tài chính (CFO). Kết quả là, CNTT thường cố gắng chuyển những dữ liệu đó thành thứ có tổ chức hơn như ở Hình 3.10, bằng cách tạo ra một tầng Quản lý dữ liệu chủ (Master Data Management) mới hoặc gọi là MDM.

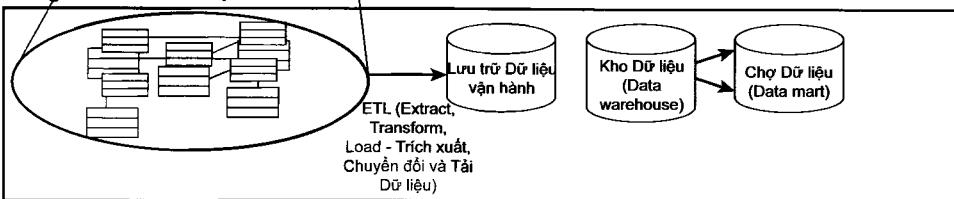
Tầng Kiến trúc nghiệp vụ



Tầng Ứng dụng phần mềm



Tầng Kiến trúc dữ liệu



Tầng Kiến trúc hạ tầng



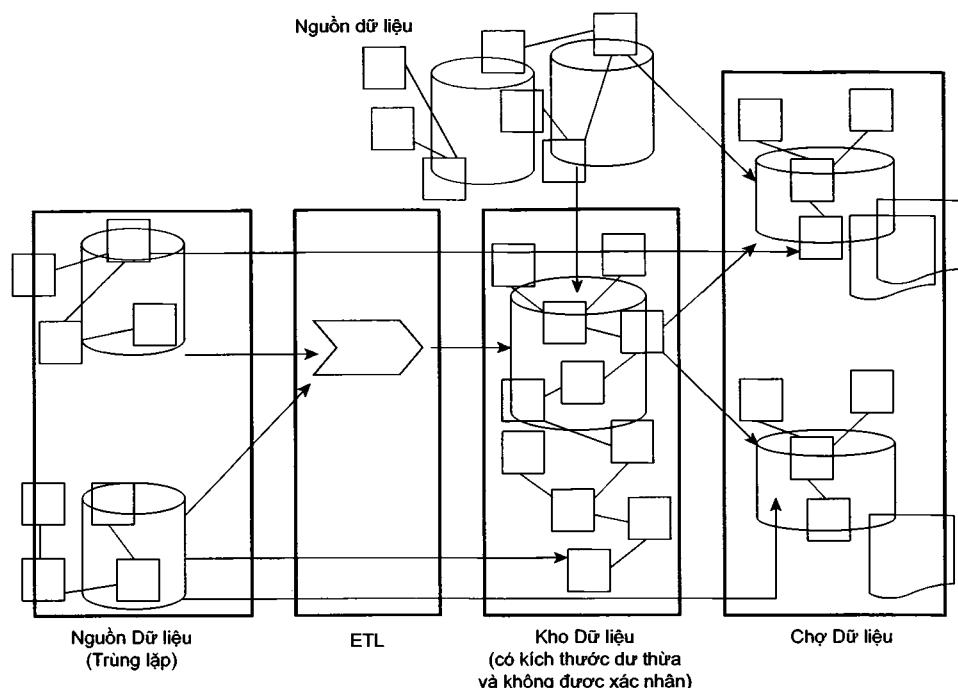
Hình 3.8

Dự án của bạn và kiến trúc doanh nghiệp truyền thống.

Bạn cần biết kiến trúc dữ liệu doanh nghiệp ở công ty hiện thời đang như thế nào và kế hoạch kiến trúc dữ liệu doanh nghiệp trong tương lai là gì.

Nắm được điều này sẽ đảm bảo cho thiết kế kiến trúc dữ liệu của ứng dụng mới theo cách phù hợp hoặc góp phần vào việc tạo ra kiến trúc dữ liệu doanh nghiệp tương lai.

Như bạn sẽ tìm hiểu trong Chương 6, “Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm” và Chương 7, “Từ tầm nhìn kiến trúc đến lập kế hoạch phát hành, lập kế hoạch Sprint và phát triển phần mềm song song”, cách tiếp cận kiến trúc khá ổn định mà chúng tôi khuyến nghị đó là dựa trên các yếu tố dữ liệu nghiệp vụ cốt lõi. Đây chính là lý do vì sao kiến trúc dữ liệu có thể giúp đóng góp vào việc tạo tầng Quản lý Dữ liệu chủ (MDM) (Hình 3.11), một thành phần chủ chốt để việc quản lý dữ liệu hoặc thông tin doanh nghiệp tốt hơn.



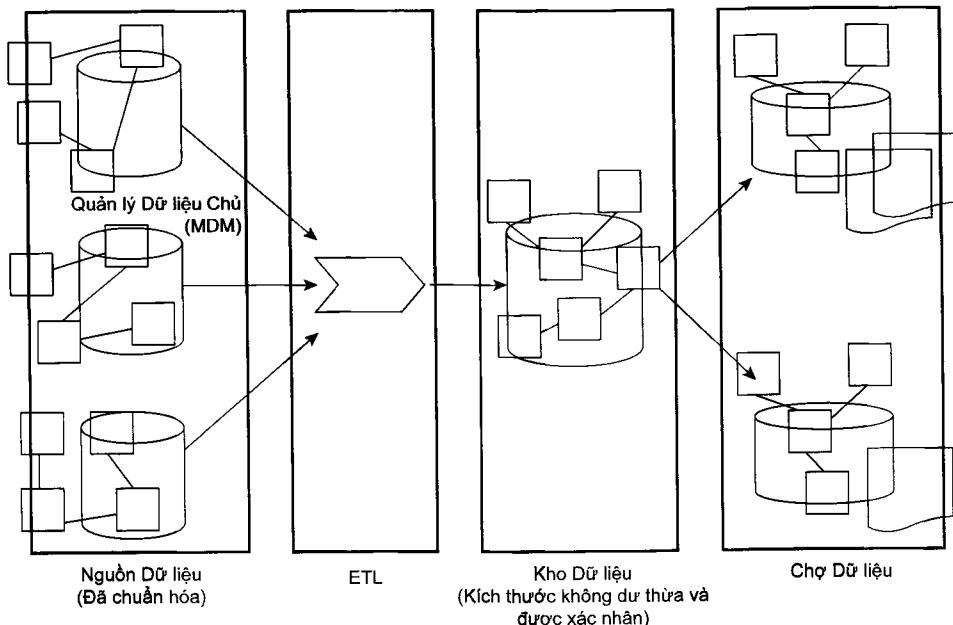
Hình 3.9

Sự hỗn loạn của dữ liệu doanh nghiệp hiện thời.

Biến người Quản lý Trực tiếp trở thành Đồng minh

Mục đích của quản lý cấp trung là “giữ cho con thuyền đi và tránh đụng vào đá”, nên trừ khi bạn báo cáo với CIO hoặc CEO, còn không hãy chắc chắn dành thời gian với quản lý cấp trung để đảm bảo rằng cô ấy hiểu Scrum một cách đầy đủ và hiểu những gì Scrum đòi hỏi.

Nếu quản lý của bạn không hiểu cách hoạt động của Scrum và dự án không tiến triển tốt hoặc chậm tiến độ, thì có nguy cơ là người quản lý trực tiếp sẽ nhanh chóng yêu cầu quay lại phong cách mệnh lệnh và kiểm soát trước đây.

**Hình 3.10**

Tái kiến trúc kiến trúc dữ liệu doanh nghiệp.

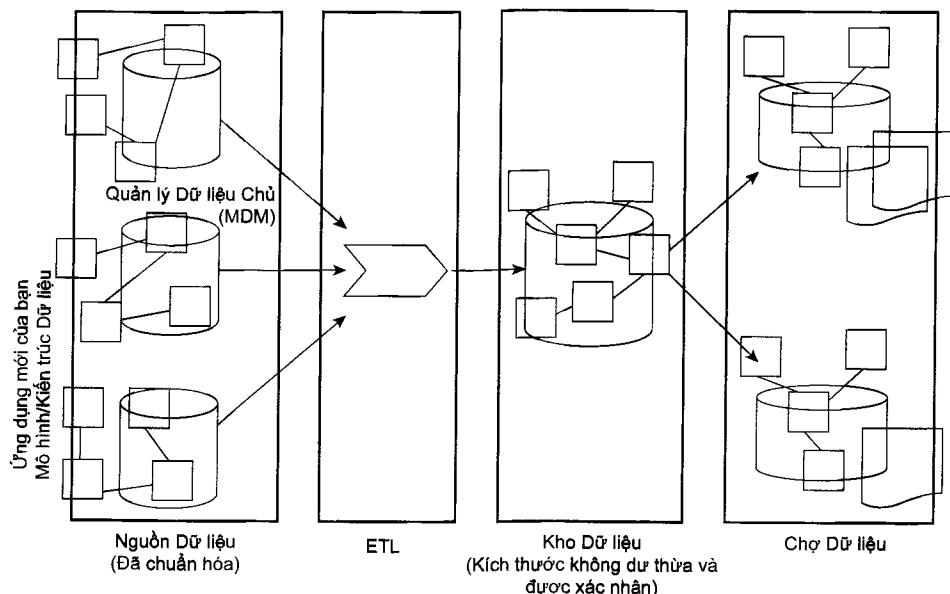
Bạn phải làm những gì có thể để truyền thụ cho người quản lý của mình trước khi bắt đầu: Scrum là gì, Scrum hoạt động như thế nào, trong quá trình triển khai dự án hằng ngày và Scrum hoạt động như thế nào khi mọi thứ diễn ra tốt đẹp và không tốt đẹp.

TÓM TẮT

Bên cạnh nhu cầu hiển nhiên là bạn cần đảm bảo sự đồng thuận từ quản lý kinh doanh cấp cao để họ tài trợ cho dự án, thì việc đạt được sự hỗ trợ của lãnh đạo cấp trung là tối quan trọng, vì bạn là thành viên của nhóm Scrum và bạn sẽ luôn phải tương tác với họ trong những hoạt động cơ bản hằng ngày. Đúng như người ta thường nói, đây là điểm mấu chốt. Những người này hoặc sẽ giúp hoặc làm hỏng dự án của bạn.

Không giống như khi đối thoại với quản lý kinh doanh chủ yếu dựa trên những con số tài chính, quan hệ với quản lý cấp trung yêu cầu sự khéo léo.

Chương 3 - Đảm bảo sự ủng hộ từ quản lý cấp cao và sự hỗ trợ từ quản lý cấp trung



Hình 3.11

Sự phù hợp của kiến trúc dữ liệu doanh nghiệp mới.

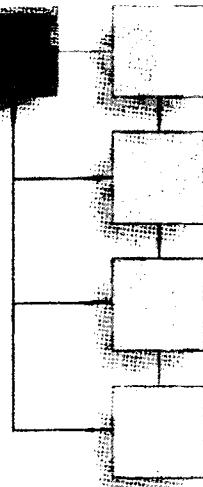
Tài chính là mối quan tâm của các lãnh đạo trong công ty. Quản lý cấp trung quan tâm tới khối lượng công việc, sự ghi nhận, an toàn công việc, ...

Trước khi bắt đầu dự án và trong khi làm việc với dự án, đừng chỉ có thông báo cho quản lý cấp trung về những thay đổi khi bạn triển khai Scrum; mà hãy cố gắng hỗ trợ họ với khối lượng công việc mới hoặc sự thay đổi mới. Điều này cùng với một chiến lược giao tiếp hiệu quả sẽ giúp bạn giành được sự hỗ trợ của họ, một điều kiện cần thiết để dự án thành công.

Cùng với lý do này, hãy nhớ làm việc thật chặt chẽ với nhóm kiến trúc doanh nghiệp để đảm bảo rằng kiến trúc ứng dụng mới phù hợp với kế hoạch và tầm nhìn của nhóm này, đặc biệt khi kiến trúc dữ liệu thường là ở trạng thái hỗn độn trong nhiều công ty và là vấn đề đau đầu cho CFO.

CHƯƠNG 4

THU THẬP CÁC YÊU CẦU TRỰC QUAN CHO PRODUCT BACKLOG



Giả sử bây giờ bạn đã có khả năng trình bày hiệu quả hơn về dự án lên quản lý kinh doanh cấp cao, những người sẽ dành sự hỗ trợ tối đa cho dự án và hiện thời bạn cũng đang háo hức triển khai. Tuy nhiên bạn nghĩ mình vẫn cần thêm điều gì đó cho dự án?

Câu trả lời là cần có một Product Backlog tốt và để thực hiện bạn cần một phương pháp tốt để thu thập các yêu cầu dưới dạng các User Story (hay còn gọi là hạng mục Product Backlog - PBI).

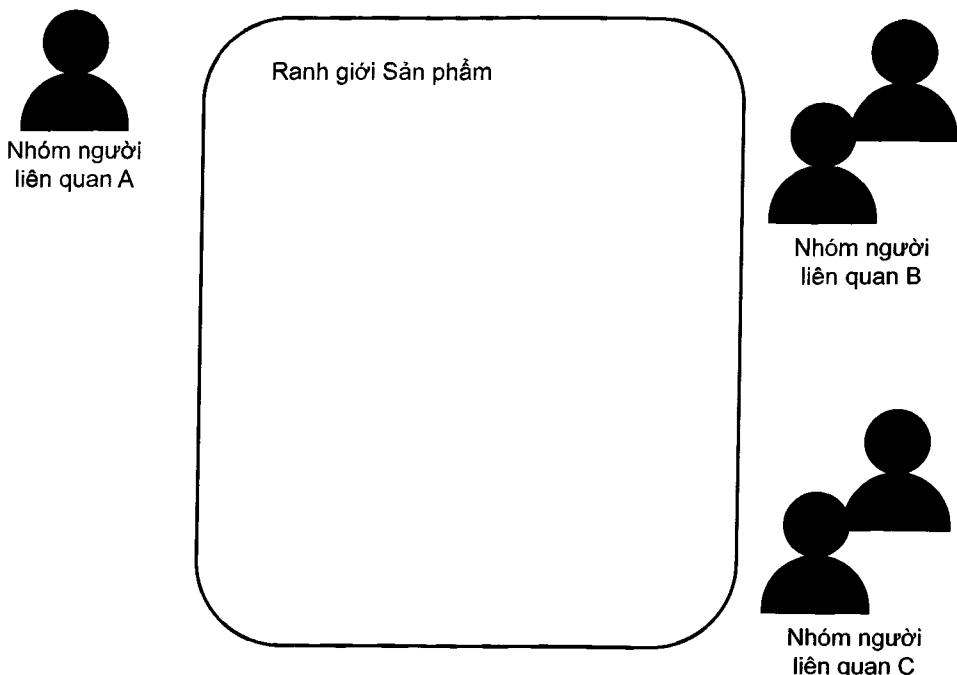
Dưới đây sẽ là một phương pháp mà chúng tôi đã sử dụng để trợ giúp nhiều nhóm thu thập các User Story cho dự án của họ. Đây là một đóng góp nhỏ của chúng tôi cho cộng đồng Agile nhằm khắc phục sự phức tạp của một số kỹ thuật khác, nhiều người đã thấy nó hữu ích, hy vọng bạn cũng sẽ như vậy.

QUY TRÌNH THU THẬP TRỰC QUAN CÁC YÊU CẦU CHO AGILE VÀ SCRUM

Quy trình này bao gồm hai giai đoạn, trước tiên giai đoạn một giúp xác định các bên liên quan và mục tiêu của họ. Sau đó giai đoạn hai sử dụng phép loại suy “rừng và cây” (forest and tree analogy) để giúp thu thập các yêu cầu từ những người đại diện cho các bên liên quan và liên hệ những yêu cầu này với những mục tiêu của các bên liên quan để sắp xếp thứ tự ưu tiên.

Bước một: Xác định những bên liên quan và Mục tiêu của họ

Xác định những nhóm người liên quan khác nhau đối với sản phẩm phần mềm mới.



Hình 4.1

Ranh giới sản phẩm và các bên liên quan.

Khi bạn mới bắt đầu dự án, hãy cố gắng xác định tất cả những người hưởng lợi, hoặc có tham gia, hoặc cần thiết cho sản phẩm phần mềm của mình (xem Hình 4.1).

Tiếp đó, rà soát lại các bên liên quan và mục tiêu của họ. Xác định mục tiêu bằng cách đặt các câu hỏi như là “Mục đích hay mục tiêu kinh doanh của bạn là gì?” “Tại sao bạn lại cần một sản phẩm phần mềm mới?” và “Làm thế nào để bạn xác định được mức độ hoàn thành các mục tiêu đó?”

Quy tắc SMART

Có nhiều phương pháp để xác định mục tiêu, một trong số đó là quy tắc SMART, chúng tôi đã sử dụng rộng rãi quy tắc này để xác định mục tiêu của rất nhiều nhóm:

- **Rõ ràng (Specific):** Tất cả mọi người đều có cùng một cách hiểu về mục tiêu.
- **Đo được (Measurable):** Chúng ta có thể xác định khách quan rằng mình đã đạt được mục tiêu hay chưa.

Bên liên quan	Mục tiêu	Độ đo
1. Phát triển Kinh doanh	Tăng 15% thị phần để đạt được 200.000 khách hàng vào cuối năm	Tăng 10.000 người dùng mỗi tháng
2. Dịch vụ Khách hàng	Giảm 20% số cuộc gọi của khách hàng mỗi quý.	Số cuộc gọi nhận được tại Call Center
	Giảm thời gian giải quyết một khiếu nại xuống 50% so với hiện tại	Thời gian nói chuyện với khách hàng qua điện thoại cho mỗi khiếu nại

Hình 4.2

Mục tiêu của các bên liên quan và độ đo. Độ đo có thể là những tiêu chí như tiết kiệm chi phí (30%), số lần gọi lên phòng Dịch vụ khách hàng (35%), hoặc số lượng người dùng đăng ký (35%).

- **Có thể đạt được (Achievable):** Các bên liên quan đồng ý về mục tiêu hướng đến.
- **Thực tế (Realistic):** Có khả năng đạt được mục tiêu của dự án với những tài nguyên sẵn có.
- **Dựa trên thời gian (Time-Based):** Có đủ thời gian để đạt được mục tiêu.

Hình 4.2 trình bày một ví dụ về những mục tiêu của các bên liên quan và độ đo dành cho các mục tiêu này.

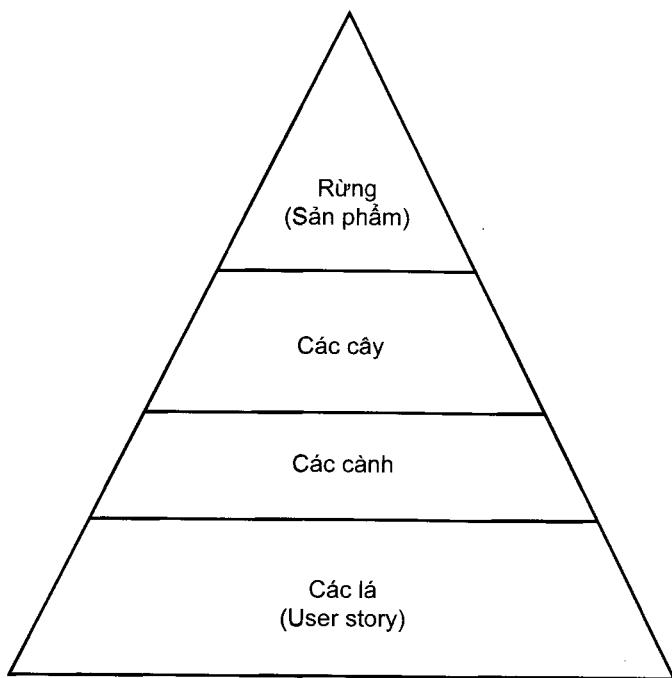
Bước hai: Thu thập yêu cầu cho Product Backlog

Trong bước này, bạn sẽ gặp những người đại diện cho các bên liên quan, mỗi người có vai trò riêng. Bước này nhằm cố gắng hiểu những gì người dùng cần và chuyển chúng thành các yêu cầu cho sản phẩm phần mềm mới.

Kỹ thuật chúng tôi sử dụng có tên là “rừng và cây”, được minh họa trong Hình 4.3, kỹ thuật này rất trực quan và dễ sử dụng.

Bắt đầu từ mức “rừng”, hay sản phẩm tổng thể. Hãy tự hỏi sản phẩm mới sẽ bao gồm những gì. Hay nói cách khác, có bao nhiêu “cây” trong “rừng” của bạn?

Chương 4 ▪ Thu thập các yêu cầu trực quan cho Product Backlog



Hình 4.3

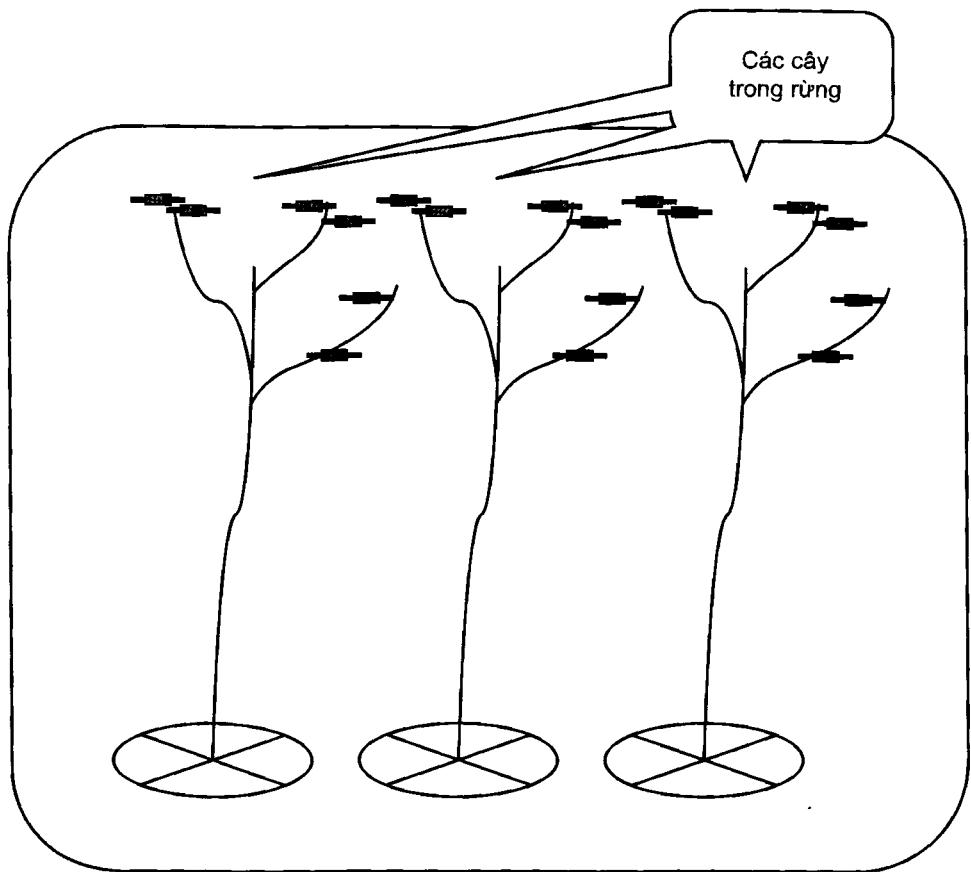
Phép loại suy “rừng và cây”.

Tiếp theo, chia nhỏ hơn nữa cây thành các cành (Hình 4.5).

Sau đó, chia cành thành các lá như được thể hiện trong Hình 4.6.

Ngoài việc đi từ trên xuống dưới như vậy, bạn có thể hỏi người dùng liệt kê tất cả những User story (hoặc “lá”) mà họ nghĩ đến trước tiên, cố gắng nhóm chúng lại với nhau, sử dụng phép loại suy “rừng và cây” này để lấy từ “lá” cho đến “cành”, rồi từ “cành” cho đến “cây” và sau cùng từ “cây” cho đến “rừng”.

Cũng theo cách này, Product Owner có thể sử dụng quy tắc SMART để kiểm tra mục tiêu của các bên liên quan, ông ta cùng nhóm có thể sử dụng quy tắc CUTFIT được trình bày trong phần sau để đảm bảo những User story, hoặc PBI, được viết ra một cách hợp lý, sẵn sàng để nhóm phát triển ước tính và phát triển.



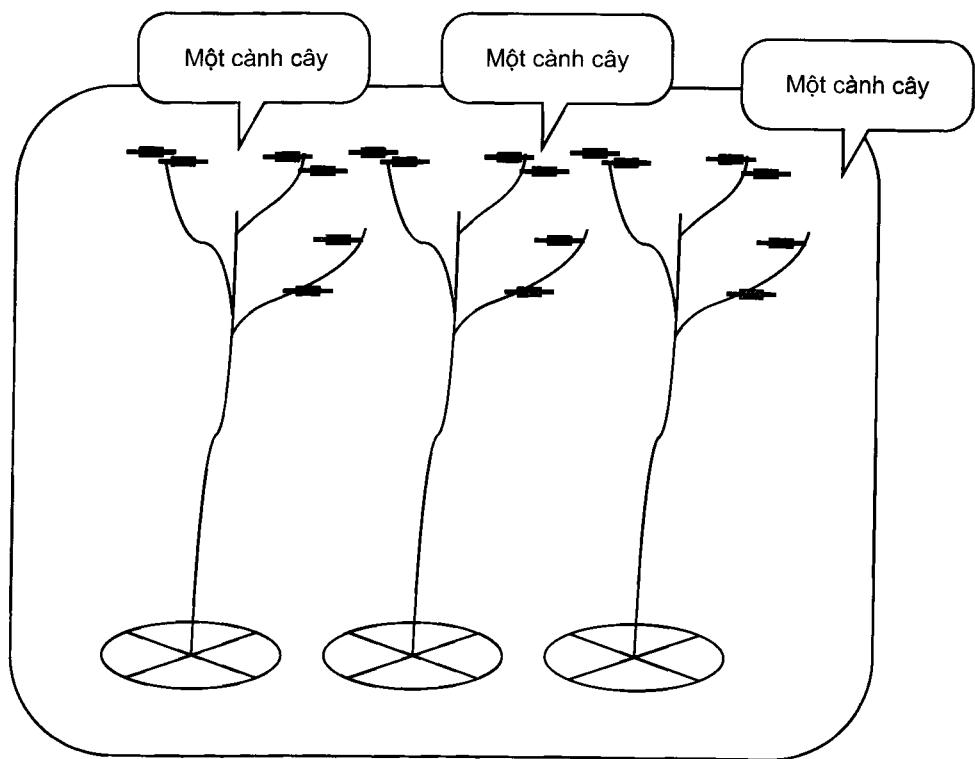
Hình 4.4

Hình ảnh một khu rừng.

Quy tắc CUTFIT

Có nhiều cách để giúp đảm bảo các yêu cầu được viết đúng, một trong số đó là quy tắc CUTFIT, được sử dụng rộng rãi để kiểm chứng các User story.

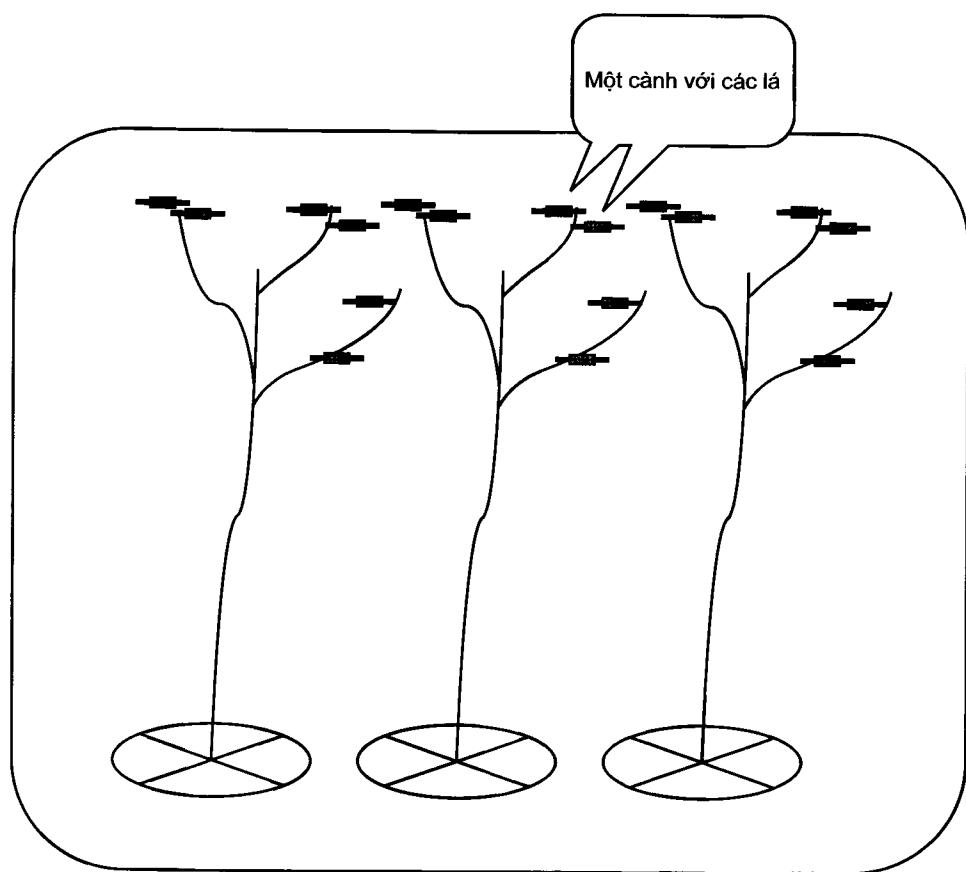
- **Nhất quán (Consistent):** Một yêu cầu được coi là nhất quán khi nó không mâu thuẫn với các yêu cầu khác.



Hình 4.5

Các cành cây.

- **Không mơ hồ (Unambiguous):** Những người kiểm tra yêu cầu phải có thể diễn giải nó theo một cách duy nhất, cho dù vai trò của họ là gì.
- **Kiểm thử được (Testable):** Chúng ta phải có khả năng tạo ra các test case (trường hợp kiểm thử) cho một yêu cầu. Nếu một yêu cầu không thể kiểm thử được thì việc xác định xem nó có được triển khai đúng hay không là vấn đề về quan điểm mà thôi.
- **Khả thi (Feasible):** Phải triển khai được từng yêu cầu trong những khả năng và hạn chế đã biết của môi trường hệ thống.
- **Độc lập (Independent):** Không có User story (PBI) này phụ thuộc vào User story (PBI) kia.
- **Theo dõi được (Traceable):** Bạn phải có khả năng liên kết từng yêu cầu đến với người dùng và mục đích của họ.



Hình 4.6

Cành và các lá của nó.

Một câu hỏi chúng tôi hay đặt ra là làm sao một người biết được họ đã hoàn thành việc viết một User story hay chưa mà không cần phải liệt kê ra tất cả các chi tiết như chúng ta thường làm trong môi trường thác nước truyền thống. Câu trả lời mà chúng tôi thường cho phép một người có thể dùng việc viết User story khi:

1. Người dùng không thể phân chia Story đó thành những Story đầu-cuối thêm được nữa, có nghĩa là những Story đó liên quan tới tất cả các tầng của ứng dụng.
2. Nhóm có thể chuyển thành các công việc (dao động từ 4 đến 8 giờ) từ những Story đó để bắt đầu việc phát triển.
3. Khi nhóm có thể bắt đầu ước tính điểm của Story đó dùng kỹ thuật dựa trên tiêu chí mà chúng tôi sẽ trình bày trong Chương 5 nhằm ước tính số điểm của Story.

MỘT VÍ DỤ

Dưới đây là một ví dụ đơn giản nhằm minh họa cho quy trình trực quan để thu thập (hay tìm kiếm) yêu cầu, ví dụ này không thể hiện toàn bộ sức mạnh của quy trình (đặc biệt là trong việc giúp sửa các lỗi do những kỹ thuật thu thập yêu cầu khác tạo ra) mà chỉ để minh họa quy trình này dễ sử dụng như thế nào.

Hãy bắt đầu bằng Hình 4.7, thể hiện ranh giới của sản phẩm và những bên liên quan.

Sau đó, hãy nói chuyện với những bên có liên quan, chúng ta có thể xác định một vài mục tiêu chính của họ, như được thể hiện trong Hình 4.8, hình này cũng trình bày độ đeo cho các mục tiêu này.

Bằng cách làm theo kỹ thuật được trình bày trong phần trước, chúng ta có thể xác định được ba cát (hoặc vùng quan tâm) của sản phẩm phần mềm mới, như thể hiện trong Hình 4.9.



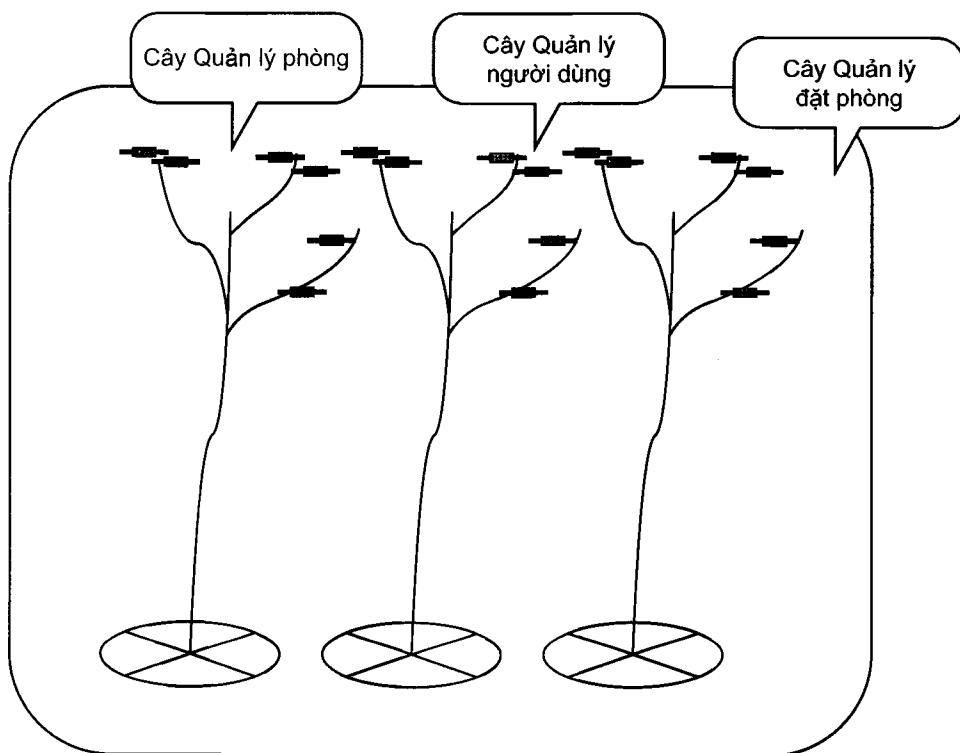
Hình 4.7

Ranh giới của sản phẩm và các bên liên quan.

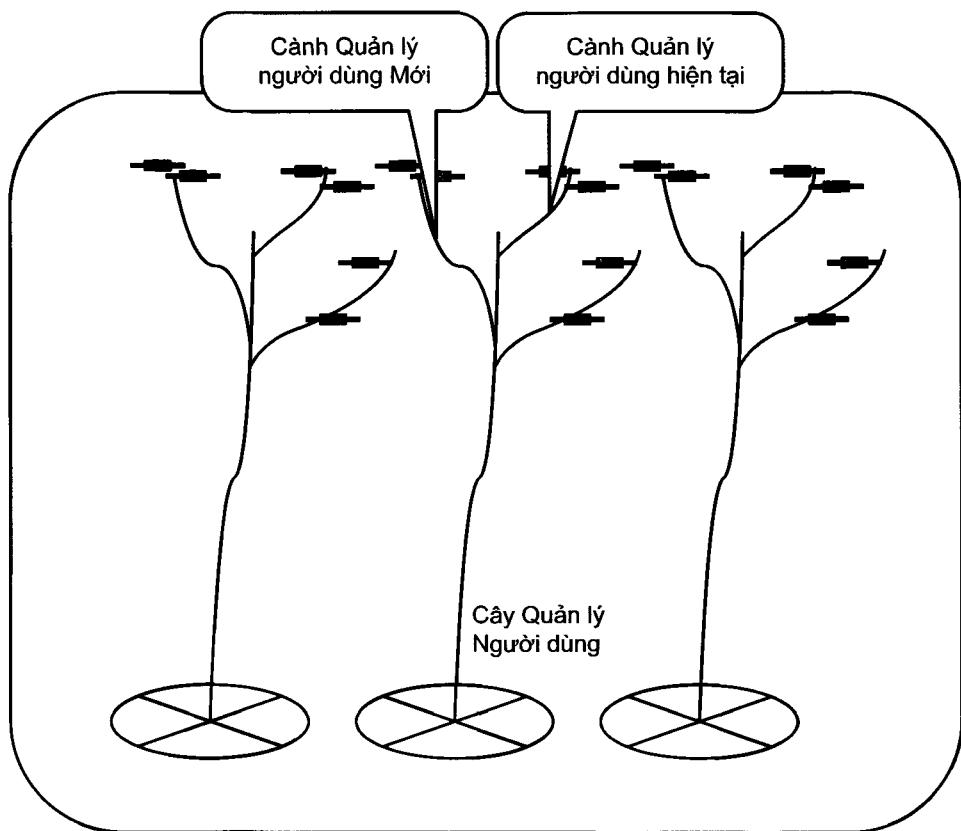
Các bên liên quan	Mục tiêu	Độ đo
1. Phát triển Kinh doanh	Cho phép khách hàng đăng ký mà không cần gọi điện đến dịch vụ khách hàng	Giao diện (GUI) phải cho phép người dùng đăng ký trong vòng ít hơn 2 phút
	Cho phép khách hàng đăng nhập vào phòng hội thoại để phản hồi các vấn đề về phần mềm	Cho phép khách hàng chỉ mất không quá một phút cho việc đăng nhập vào phòng hội thoại để phản hồi các vấn đề về phần mềm
2. Dịch vụ Khách hàng	Cho phép khách hàng lập hóa đơn cho một ngân hàng khác	Hệ thống phải cho phép khách hàng chỉnh sửa hóa đơn của họ 24x7

Hình 4.8

Những mục tiêu của người dùng và độ đo dành cho mỗi mục tiêu.

**Hình 4.9**

Phần mềm Đặt phòng riêng và các “cây” của nó.



Hình 4.10

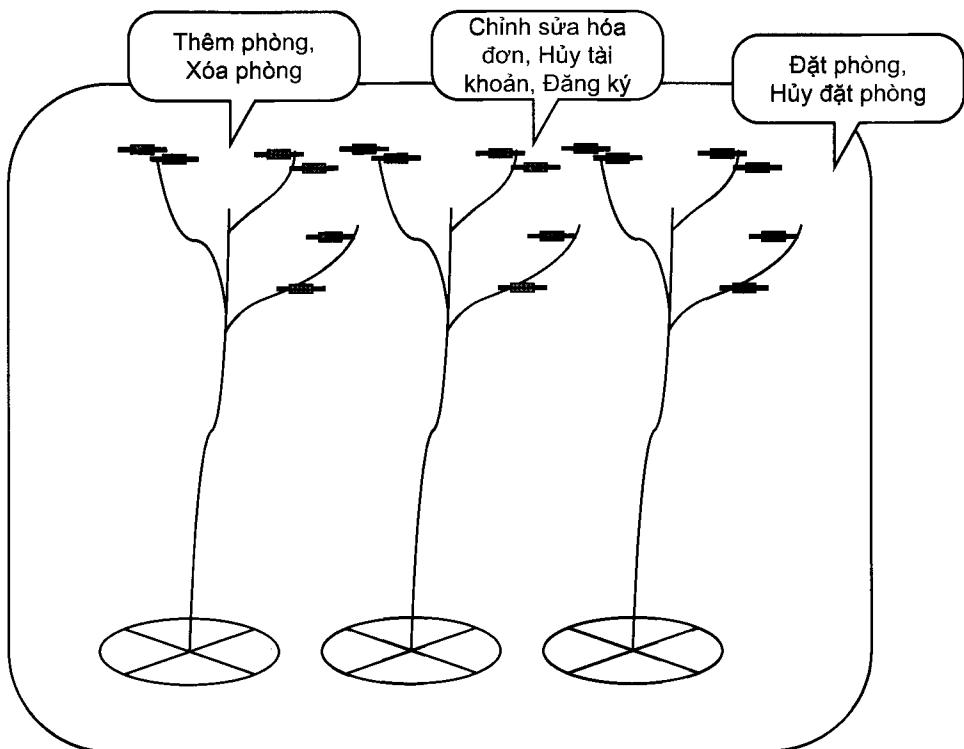
Cây quản lý người dùng và các cành của nó.

Cụ thể hơn, các cây trong sản phẩm phần mềm này gồm Quản lý người dùng, Quản lý phòng và Quản lý đặt phòng.

Như được thể hiện trong Hình 4.10, bằng việc nhìn vào “Cây Quản lý người dùng”, chúng ta có thể thấy rằng nó có hai cành, “Cành Quản lý người dùng mới” và “Cành Quản lý người dùng hiện tại”.

Cuối cùng, bằng việc nhìn vào tất cả các lá trên tất cả các cành ta có thể liệt kê toàn bộ lá như trong Hình 4.11.

Hình 4.12 mô tả sản phẩm phần mềm sẽ trông như thế nào với tất cả các lá, các cành và tất cả các cây trong rừng.

**Hình 4.11**

Phần mềm Đặt phòng riêng và các “lá” của nó.

Theo kết quả của quá trình sắp thứ tự ưu tiên, chúng ta có thể giả định rằng các bên liên quan trước tiên sẽ muốn tập trung vào những User story mang lại nhiều giá trị nhất cho công ty hoặc bộ phận kinh doanh của họ.

Sau khi quyết định được đưa ra, một tấm thẻ Story (story card) như trong Hình 4.13 có thể được dùng để mô tả một số nhiệm vụ cấp cao mà người dùng muốn thực hiện, kèm với tất cả những test case (“kiểm tra đảm bảo rằng...”) cần phải thực hiện đối với User story đó.

Chương 4 • Thu thập các yêu cầu cấu trúc quan cho Product Backlog



Hình 4.12

Cách tiếp cận rừng và cây đã được áp dụng cho phát triển sản phẩm phần mềm.

Là người dùng, tôi có thể/muốn [...], nhằm giúp [với mục đích]
Nhiệm vụ:
Kiểm tra: Kiểm tra:

Hình 4.13

Một thẻ Story nguyên bản.

TÓM TẮT

Một trong những nhiệm vụ quan trọng nhất của nhóm dự án Scrum là xác định các bên liên quan và mục tiêu của họ, nhằm lấy được những yêu cầu của người dùng đại diện cho các bên liên quan đó, mỗi người có vai trò riêng.

Trong khi tồn tại nhiều hướng tiếp cận khác nhau, chúng tôi cung cấp một quy trình đơn giản và trực quan gồm hai bước mà mọi người đều có thể sử dụng. Bước thứ nhất là xác định các bên liên quan và mục tiêu của họ. Bước thứ hai là xác định những người dùng đại diện cho các bên liên quan đó, cùng với những yêu cầu của họ. Chúng tôi đã thực hiện tất cả những công việc này bằng việc sử dụng một kỹ thuật trực quan được gọi là cách tiếp cận Rừng và Cây.

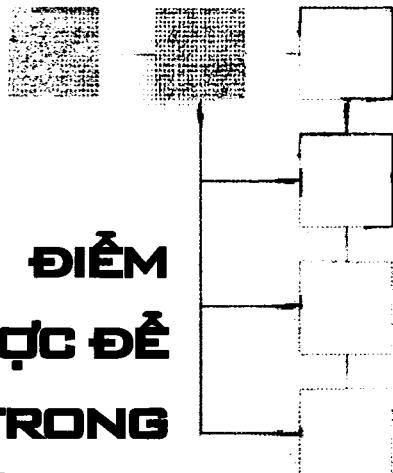
Để xác định xem liệu những mục tiêu của các bên liên quan đã được viết đúng hay chưa, họ có thể sử dụng một loạt các quy tắc được gọi là quy tắc SMART. Các quy tắc này yêu cầu mục tiêu phải rõ ràng, có thể đo được, đạt được, thực tế và dựa trên thời gian (tức là có thể đạt được trong một khung thời gian đã cho).

Để xác định xem liệu các yêu cầu (User story hay PBI) đã được viết hợp lý và sẵn sàng cho phát triển hay chưa thì Product Owner và nhóm có thể sử dụng quy tắc CUTFIT. Các quy tắc này yêu cầu một User story phải nhất quán, không mờ hồ, kiểm thử được, khả thi, độc lập và có thể theo dõi được.

Khi được hỏi về việc làm thế nào để một người có thể biết rằng họ đã hoàn thành công việc viết các User story để lập kế hoạch phát hành và kế hoạch Sprint (mà không cần liệt kê tất cả các chi tiết) hay chưa, chương này cung cấp cho bạn ba căn cứ để giúp xác định việc đó.

CHƯƠNG 5

TẠO RA ƯỚC TÍNH ĐIỂM STORY SO SÁNH ĐƯỢC ĐỀ TRIỂN KHAI SCRUM TRONG TOÀN DOANH NGHIỆP



Nếu bạn chưa có chút kinh nghiệm nào về lập kế hoạch và ước tính trong Agile hoặc Scrum thì chắc bạn cũng đã từng nghe nói hoặc đã đọc về Planning Poker (Bộ bài Lập kế hoạch). Nói một cách ngắn gọn thì Planning Poker là một kỹ thuật ước tính được sử dụng bởi các nhóm dự án Agile để cùng nhau ước tính kích cỡ tương đối của các Story. Các ước tính này sử dụng đơn vị đo gọi là điểm Story (Story Point).

Để việc sử dụng Planning Poker hiệu quả hơn, các nhóm dự án thường bao gồm các chuyên gia, những người đã từng có kinh nghiệm với nhiều công việc khác nhau trong phát triển phần mềm như xác định yêu cầu, phân tích, thiết kế, viết mã và kiểm thử.

VẤN ĐỀ VỚI ĐIỂM STORY KHÔNG SO SÁNH ĐƯỢC

Khi Agile hoặc Scrum được triển khai trong một vài dự án riêng biệt và không có nhu cầu triển khai trong toàn doanh nghiệp, giữa nhiều nhóm khác nhau, thì mọi thứ đều chấp nhận được. Nhưng khi vấn đề triển khai Agile hoặc Scrum trong toàn doanh nghiệp được quan tâm đến thì rõ ràng việc điểm Story không so sánh được giữa các nhóm khác nhau sẽ là nguy cơ mang đến những ảnh hưởng tiêu cực trong việc triển khai Agile hoặc Scrum.

Để rõ ràng hơn, những gì chúng tôi đề cập ở đây không phải là sự so sánh hoặc cạnh tranh không lành mạnh hoặc không cần thiết giữa các nhóm. Đơn giản chúng tôi chỉ nói rằng khi Scrum được triển khai giữa các nhóm khác nhau, thì ít nhất chúng ta cũng phải mong chờ khả năng cập nhật công tác quản lý khi có một sự thay đổi trong thành phần của một nhóm dự án, chẳng hạn việc thuyên chuyển nhân sự và việc thay đổi đó có ảnh hưởng đến thời hạn của dự án hay không.

Trừ khi bạn đang làm việc trong một công ty mà bạn là nhân viên duy nhất. Nếu không khi chúng ta không có bất cứ thứ gì có thể so sánh được giữa các nhóm. Chúng ta không biết làm thế nào để đưa ra Dự toán để Hoàn thành (Estimate to Complete - ETC)

cho quản lý kinh doanh hoặc CNTT khi các thành viên xin thôi việc ở công ty hoặc chuyển ra khỏi dự án?

Từ đó, bạn có thể thấy được tại sao đây lại là vấn đề đối với một số PMO trong việc lập kế hoạch triển khai Scrum hoặc Agile trong toàn doanh nghiệp khi không so sánh được điểm Story và tốc độ giữa các nhóm (cũng dựa trên điểm Story).

CÁC VẤN ĐỀ VỀ VĂN HÓA VỚI PLANNING POKER

Ngoài những việc đã nêu ở trên, xuất phát từ việc Agile và Scrum được phổ biến từ nước Mỹ sang các nước khác, chúng tôi đã quan sát và thấy rằng Planning Poker không phải luôn luôn hoạt động tốt trong những nền văn hóa khác với phương Tây, đặc biệt là ở Châu Á, quê hương của chúng tôi. Lý do là ở trong nhiều nền văn hóa khác phương Tây, việc tôn trọng những người lớn tuổi và những người có vai trò lãnh đạo thường hạn chế các thành viên trong việc đưa ra các ước tính khác với ước tính của người lớn tuổi hoặc của thành viên được xếp hạng cao hơn.

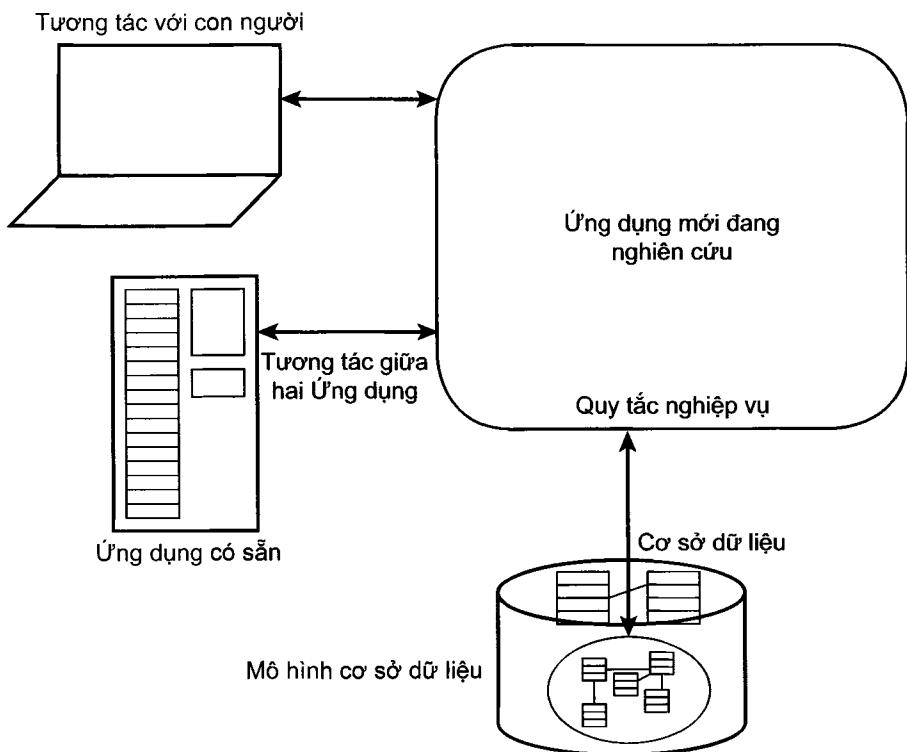
Mặc dù có những hạn chế nêu trên, chúng tôi phải nói rằng Planning Poker đã có sự ảnh hưởng rất tốt đối với cộng đồng của chúng ta nói chung, đặc biệt là trong những giai đoạn đầu của phong trào Scrum hoặc Agile. Nói như vậy, có lẽ là đã đến lúc chúng ta cần tìm ra một hướng tiếp cận khác khách quan hơn, đặc biệt là khi việc triển khai Agile hoặc Scrum trên toàn doanh nghiệp và cấp phát tài nguyên là cần thiết.

QUY TRÌNH ƯỚC TÍNH DỰA THEO TIÊU CHÍ KHÁCH QUAN

Sẽ là không thành thực nếu nói rằng sau nhiều năm nghiên cứu ở phòng thí nghiệm uy tín, chúng tôi đã phát minh ra cách ước tính này để gây ấn tượng với bạn. Thay vào đó, chúng tôi sẽ nói rằng cách tiếp cận được giới thiệu ở đây là sự kết hợp những ý tưởng có nguồn gốc từ các chuyên gia nổi tiếng mà chúng tôi đã học được sau nhiều năm quản lý và thực hiện ước tính nhiều dự án ở nhiều lĩnh vực khác nhau. Một điều nữa mà chúng tôi muốn nói là phương pháp này đã được áp dụng trong những dự án thật và những dự án này đã được cài đặt.

Ở Hình 5.1, bạn sẽ thấy rằng một ứng dụng không có gì hơn là (1) một vài người dùng nghiệp vụ muốn tương tác với mã nguồn chạy được, mã nguồn này cài đặt (2) một vài quy tắc nghiệp vụ theo (3) một mô hình chứa một vài thực thể nghiệp vụ, giá trị của những thực thể này được lưu trữ trong cơ sở dữ liệu vật lý, (4) cơ sở dữ liệu này cho phép tạo, đọc, cập nhật, hoặc xóa.

Quy trình Ước tính dựa theo tiêu chí khách quan



Hình 5.1

Những khía cạnh khác nhau của một ứng dụng.

Do đó, đây là cách chúng ta sẽ ước tính User story hoặc hạng mục Product Backlog, từng loại một (Hình 5.2).

1. Loại tương tác
2. Quy tắc nghiệp vụ

Loại tương tác	Mô tả	Giá trị
Đơn giản	Giao diện được định nghĩa rõ ràng	1
Trung bình	Giao diện động	2
Phức tạp	Tương tác với con người	3

Hình 5.2

Loại tương tác.

3. Số lượng thực thể được thao tác
4. Dữ liệu được tạo, đọc, cập nhật và xóa (CRUD - created, read, updated, deleted)

Chương 5 • Tạo ra ước tính điểm story so sánh được để triển khai Scrum trong toàn doanh nghiệp

Theo bảng trong Hình 5.2 thì, nếu Story cần một tương tác với con người, bạn nên cho nó giá trị 3. Tuy nhiên, nếu một Story chỉ cần tương tác với ứng dụng khác, theo một giao thức được định nghĩa rõ ràng, thì User story đó chỉ nên nhận giá trị 1 cho việc tương tác.

Tiếp theo, chúng ta hãy tính toán mức độ phức tạp dựa vào số lượng quy tắc nghiệp vụ được áp dụng.

Quy tắc nghiệp vụ	Mô tả	Giá trị
Đơn giản	1 quy tắc	1
Trung bình	1-3 quy tắc	2
Phức tạp	>3 quy tắc	3

Hình 5.3

Mức độ phức tạp của quy tắc nghiệp vụ.

Nếu có một quy tắc nghiệp vụ, thì bạn nên cho Story đó giá trị 1. Nếu có nhiều hơn một và ít hơn ba quy tắc nghiệp vụ thì story đó nên nhận giá trị 2. Nếu nhiều hơn ba quy tắc, hãy cho Story đó giá trị 3.

Dù loại tương tác ở bảng của Hình 5.3 khá rõ ràng, chúng tôi vẫn cho rằng bạn có những thắc mắc về khái niệm quy tắc nghiệp vụ. Một quy tắc nghiệp vụ là một chỉ thị cho bạn biết khi nào bạn có thể hoặc không thể làm gì.

Vậy đâu là sự khác biệt giữa quy tắc nghiệp vụ và yêu cầu nghiệp vụ? Để đơn giản, thì một yêu cầu nghiệp vụ là điều bạn cần có ở bản cài đặt và tuân thủ theo một quy tắc nghiệp vụ, quy tắc này bị chi phối bởi luật hoặc chính sách của công ty.

Ví dụ về một quy tắc nghiệp vụ:

Người bảo trợ phải từ 18 tuổi trở lên.

Đây có thể là một yêu cầu nghiệp vụ đảm bảo quy tắc đó:

Người bảo trợ phải sở hữu giấy phép lái xe hoặc hộ chiếu hợp lệ có ảnh để xác minh rằng người bảo trợ đó ít nhất 18 tuổi.

Tiếp theo, chúng ta cần xác định số lượng thực tế dữ liệu cần thiết để thực thi User story này (Hình 5.4).

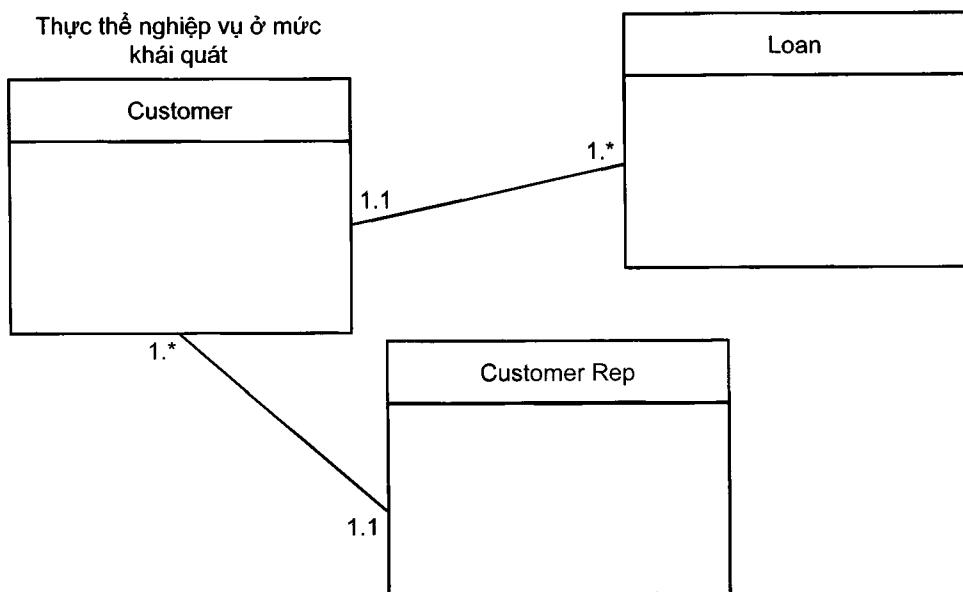
Số lượng thực tế được thao tác cho thấy rằng nếu lượng thực tế dữ liệu chỉ là một, thì bạn nên cho Story đó giá trị 1, nhưng nếu số lượng thực tế dữ liệu nằm giữa hai và ba thì Story nên có giá trị 2, ...

Quy trình Uớc tính dựa theo tiêu chí khách quan

Thực thể	Mô tả	Giá trị
Đơn giản	1 Thực thể	1
Trung bình	1-3 Thực thể	2
Phức tạp	>3 Thực thể	3

Hình 5.4:

Số lượng thực thể nghiệp vụ được thao tác.



Hình 5.5

Một cấu trúc dữ liệu đơn giản với ba thực thể nghiệp vụ.

Hình trên là một ví dụ về thực thể nghiệp vụ. Trong mô hình dữ liệu nhỏ này (Hình 5.5), chúng ta có tất cả ba thực thể dữ liệu là, (1) Khách hàng (Customer), (2) Khoản vay (Loan), (3) Đại diện khách hàng (Customer Rep) để phục vụ khác hàng đó.

Cuối cùng, chúng ta cần xác định hệ số thao tác dữ liệu (CRUD).

Các hệ số CRUD (Tạo, Đọc, Cập nhật, Xóa) này được minh họa ở Hình 5.6:

Chương 5 • Tạo ra ước tính điểm story so sánh được để triển khai Scrum trong toàn doanh nghiệp

Loại thao tác dữ liệu	Mô tả	Giá trị
Đơn giản	Đọc, Xóa	1
Trung bình	Tạo	2
Phức tạp	Cập nhật	3

Hình 5.6

CRUD.

Bây giờ hãy tưởng tượng rằng có một User story tên là “Thêm phòng (hợp)”. Chúng ta sẽ thử cùng nhau tính Điểm Chưa hiệu chỉnh (Unadjusted Point - UP) trước khi xem xét tới các khía cạnh môi trường (environment dimension - ED) của dự án.

- A. Loại tương tác = 3 điểm
- B. Quy tắc nghiệp vụ = 1 điểm
- C. Số lượng thực thể được thao tác = 1 điểm
- D. Tạo, Đọc, Cập nhật, Xóa (CRUD) = 2 điểm

UP = 7 điểm (Điểm chưa hiệu chỉnh cho Story hoặc PBI “Thêm phòng”).

Tiếp theo, chúng ta sẽ xem xét các Khía cạnh Môi Trường (ED), chúng có thể ảnh hưởng tích cực hoặc tiêu cực tới những cố gắng của nhóm trong việc chuyển giao story “Thêm phòng”:

1. Khía cạnh Tổ chức
2. Khía cạnh Hạ tầng phát triển
3. Khía cạnh Nhóm
4. Khía cạnh Công nghệ
5. Khía cạnh Quy trình
6. Khía cạnh Nghiệp vụ

Ở mỗi khía cạnh này, giá trị cao chỉ năng lực hoặc khả năng của nhóm cao, trong khi giá trị thấp chỉ năng lực hoặc khả năng của nhóm thấp. Điểm thấp nhất sẽ là không, trong khi giá trị dương chỉ năng lực hoặc khả năng của nhóm cao với 2 là giá trị lớn nhất. Như đã làm ở những bảng trên, bạn nên lắn lướt xử lý từng khía cạnh này, từng loại một và quan sát giá trị mà bạn thu được với mỗi câu hỏi.

Giải quyết hết những bảng này (Hình 5.7 – 5.12), bạn sẽ nhìn thấy tổng điểm ước tính cho Story tính đến tất cả các khía cạnh môi trường (ED). Tổng này sẽ nằm trong khoảng từ 0 tới 36.

Quy trình Ước tính dựa theo tiêu chí khách quan

Yếu tố	Khoảng Giá trị (0/2)
1. Đã có những phòng ban khác nhau cùng làm việc thành công trong một dự án Scrum?	
2. Có sự chống đối mạnh mẽ với Scrum trong tổ chức?	
3. Có tồn tại sự hỗ trợ lớn về Scrum giữa những phòng ban khác nhau trong công ty?	

Hình 5.7

Khía cạnh tổ chức.

Yếu tố	Khoảng Giá trị (0/2)
1. Kiểm thử tự động đã được áp dụng và trở thành một kỹ thuật phổ biến hay chưa?	
2. Kiểm thử tích hợp liên tục đã được áp dụng và trở thành một kỹ thuật phổ biến hay chưa?	
3. Môi trường build (build environment) hằng ngày đã được áp dụng và trở thành một kỹ thuật phổ biến hay chưa?	

Hình 5.8

Khía cạnh hạ tầng phát triển.

Tùy thuộc vào giá trị của tổng này, chúng ta sẽ có ba kịch bản:

1. **Nếu giá trị ED nằm giữa 0 và 11, thì hệ số nhân C sẽ là 2.** Điều này có nghĩa là với các khía cạnh môi trường như vậy thì nhóm sẽ không thể chuyển giao được nhiều Story hơn trong Sprint so với trường hợp có điểm ED cao hơn.

Yếu tố	Khoảng Giá trị (0/2)
1. Scrum là hoàn toàn mới đối với nhóm?	
2. Có phải trước đó các thành viên trong nhóm đã từng làm việc thành công với nhau?	
3. Các thành viên trong nhóm hiểu và tôn trọng lẫn nhau?	

Hình 5.9

Khía cạnh nhóm.

Chương 5 • Tạo ra ước tính điểm story so sánh được để triển khai Scrum trong toàn doanh nghiệp

Yếu tố	Khoảng Giá trị (0/2)
1. Nhóm phát triển có nhiều kinh nghiệm với ngôn ngữ lập trình?	
2. Các thành viên trong nhóm phát triển có nhiều kinh nghiệm với công nghệ được sử dụng?	
3. Môi trường chạy ứng dụng thực tế với Scrum đã sẵn sàng?	

Hình 5.10

Khía cạnh công nghệ.

2. Nếu giá trị ED nằm giữa 12 và 23, thì hệ số nhân C sẽ là 1. Điều này có nghĩa là môi trường làm cho công việc của nhóm không dễ hơn cũng không khó hơn.
3. Nếu giá trị ED nằm giữa 24 và 36, thì hệ số nhân C sẽ là $\frac{1}{2}$. Điều này có nghĩa là môi trường đang giúp nhóm chuyển giao được nhiều story hơn trong một Sprint.

Yếu tố	Khoảng Giá trị (0/2)
1. Scrum có phải là khung quy trình được chấp thuận trong công ty hay không?	
2. Trong công ty có sự hỗ trợ tốt cho Scrum hay không?	
3. Trong công ty có sự phản đối đáng kể nào đối với Scrum hay không?	

Hình 5.11

Khía cạnh quy trình.

Yếu tố	Khoảng Giá trị (0/2)
1. Có một Product Owner nào hoàn toàn sẵn sàng và gắn bó với nhóm hay không?	•
2. Có phải Product Owner đã quen thuộc với Scrum nhưng vẫn thiếu kinh nghiệm thực tế hay không?	
3. Product Owner đã từng thành công với Scrum trước đây hay chưa?	

Hình 5.12

Khía cạnh nghiệp vụ.

Do đó, để tính toán tổng điểm cho mỗi Story, đơn giản hãy sử dụng công thức sau:

$$AP \text{ (Điểm đã hiệu chỉnh)} = UP \text{ (Điểm chưa hiệu chỉnh)} * C \text{ (Hệ số nhân)}$$

$$PPS \text{ (Điểm cho mỗi Story)} = (AP * ED) / 36$$

Ví dụ

Hãy xem xét lại tính năng “Thêm phòng” và giả thiết rằng giá trị của các khía cạnh môi trường (ED) lần lượt được liệt kê dưới đây:

1.	Tổ chức	= 3
2.	Hạ tầng	= 2
3.	Nhóm	= 4
4.	Công nghệ	= 3
5.	Quy trình	= 2
6.	Nghiệp vụ	= 4

Cộng tất cả những giá trị này cho ta ED bằng 18. Bởi ED bằng 18, như đã đề cập trước đó, điều này có nghĩa rằng hệ số nhân (C) bằng 1.

Nói cách khác, phép tính cho Story “Thêm phòng” sẽ bằng:

$$AP = UP \times C$$

$$AP = 7 \times 1 = 7$$

(Với UP bằng 7 điểm như đã tính ở trên). Vậy,

$$PPS = (AP \times ED) / 36$$

$$PPS = (7 \times 18) / 36 = 126 / 36 = 3,5 \text{ điểm.}$$

Sử dụng cùng công thức này cho các Story khác, như bảng trong Hình 5.13, sẽ cung cấp cho ta ước tính tổng quát về toàn bộ sản phẩm được xây dựng.

Giờ ta thấy rõ ràng ưu điểm của loại tính toán này là dựa trên các tiêu chí khách quan; do đó nó phù hợp cho việc so sánh giữa những nhóm khác nhau, thậm chí cho những thành viên khác nhau của cùng một nhóm dự án.

Với những gì đã trải qua, có vẻ như chúng ta sẽ có thể cải tiến những tấm thẻ story phổ biến để chúng trông giống như ở Hình 5.14 và để tận dụng toàn bộ các lợi thế của quá trình ước tính vừa được trình bày.

Chương 5 ▪ Tạo ra ước tính điểm story so sánh được để triển khai Scrum trong toàn doanh nghiệp

Đặc điểm				Tổng UP (Điểm Chưa Hiệu chỉnh)	Hệ số nhân	AP (Điểm Đã Hiệu chỉnh)	ED (Khi cạnh Môi trường)	PPS (=AP * ED) / 36
Loại Tương tác	Quy tắc Nghiệp vụ	Thực thi	Loại Thao tác Dữ liệu					
Sprint 1								
Đăng ký	3	1	2	7	1	7	12	3,5
Thêm phòng	3	1	2	7	1	7	12	3,5
Xóa phòng	3	1	3	8	1	8	12	4
Sprint 2								
Chỉnh sửa thông tin hóa đơn	3	1	3	8	1	8	12	4
Hủy tài khoản	3	1	1	3	8	1	8	12
Duyệt các bản ghi	3	1	1	3	8	1	8	12
Tổng (Sprint 1 + Sprint 2)							108	23

Hình 5.13

Má trận ước tính tổng thể.

Là người dùng, tôi có thể/muốn [...], nhằm giúp [với mục đích]
Nhiệm vụ:
Kiểm tra:
Kiểm tra:

Loại tương tác:
Số lượng Quy tắc nghiệp vụ:
Số lượng Thực thể dữ liệu:
CRUD:

Hình 5.14

Tấm thẻ Story đã cài tiến.

TÓM TẮT

Như hệ quả của thành công với Scrum, ngày càng nhiều công ty dự tính triển khai Scrum cho toàn bộ phòng CNTT của họ.

Một trong những khó khăn của việc này là điểm Story giữa các nhóm không so sánh được với nhau. Do trọng số tốc độ là khác nhau giữa các nhóm, bạn có thể thấy được lý do tại sao việc điểm Story không so sánh được giữa các nhóm lại trở thành một vấn đề mà những PMO trong Agile cần tính tới khi lên kế hoạch triển khai Scrum cho toàn bộ doanh nghiệp.

Trong chương này, chúng tôi đã giới thiệu một kỹ thuật mới dựa trên quy trình ước tính theo tiêu chí khách quan, là một chuỗi những bảng tương đối đơn giản để hướng dẫn nhóm ước tính những Story hoặc PBI khác nhau. Nhờ đơn giản mà phương pháp này cho phép chúng ta so sánh khách quan giữa các nhóm cũng như những thành viên trong cùng một nhóm Scrum.

CHƯƠNG 6

ẢNH HƯỞNG CỦA TẦM NHÌN KIẾN TRÚC TỚI TỐC ĐỘ NHÓM VÀ CHẤT LƯỢNG PHẦN MỀM

Tiêu đề của chương nghe có vẻ kỹ thuật, nhưng chúng tôi hy vọng sẽ trình bày được thông tin theo cách mà cả những người phi kỹ thuật cũng có thể hiểu được các khái niệm trong chương này.

Bởi việc mọi thành viên trong nhóm Scrum hiểu ý nghĩa của kiến trúc và tầm nhìn kiến trúc là rất quan trọng, nên chúng ta hãy dành một vài phút để định nghĩa hai thuật ngữ này trước khi tiếp tục.

Hãy tưởng tượng rằng bạn đang trả tiền cho việc xây dựng ngôi nhà mơ ước của mình (sản phẩm phần mềm mới). Tất nhiên, bạn không muốn rằng đội xây dựng xuất hiện trước bãi đất trống và bắt đầu công việc của họ (lập trình) ngay từ đầu.

Điều bạn muốn đầu tiên là tiếp xúc với các thành viên đội xây dựng để mô tả cho họ về hình dung và những yêu cầu của bạn với ngôi nhà (sản phẩm phần mềm). Ví dụ, bạn sẽ nói với họ rằng hình dung (tầm nhìn) của bạn là một ngôi nhà hai tầng với nhiều cửa kính nhất có thể (User story hoặc yêu cầu).

Cũng có thể bạn nói với họ rằng mình có kế hoạch sinh thêm một con nữa cùng với hai con đã có (một yêu cầu khác). Cũng có thể bạn muốn lũ trẻ sống ở tầng dưới (một ràng buộc) bởi bạn không thể chịu được tiếng ồn mà lũ trẻ chạy nhảy trên đầu gáy ra, đây là một yêu cầu khác (yêu cầu sản phẩm phần mềm).

Sau khi nghe những yêu cầu của bạn, nhóm xây dựng sẽ tạo một số bản vẽ (tầm nhìn kiến trúc), ở mức tổng quan, về hình dáng của ngôi nhà và cách bố trí những thành phần chính.

Đây chính là cách mà những người xây dựng (nhà phát triển phần mềm) sẽ lấy ý tưởng về cái họ nên làm và làm nó ở đâu (kiến trúc được xây dựng theo thời gian).

Chúng tôi gọi bản vẽ tổng thể có được từ sự hình dung về các nhu cầu là tầm nhìn kiến trúc, hoặc bản thiết kế kiến trúc tổng thể, trong khi kết quả mà nhóm có được từ công việc xây dựng sẽ là kiến trúc ngôi nhà (kiến trúc phần mềm), một loại khung sườn chính hỗ trợ để có được tổng thể của ngôi nhà.

Chương 6 • Ảnh hưởng của Tâm nhìn kiến trúc tới tốc độ Nhóm và Chất lượng phần mềm

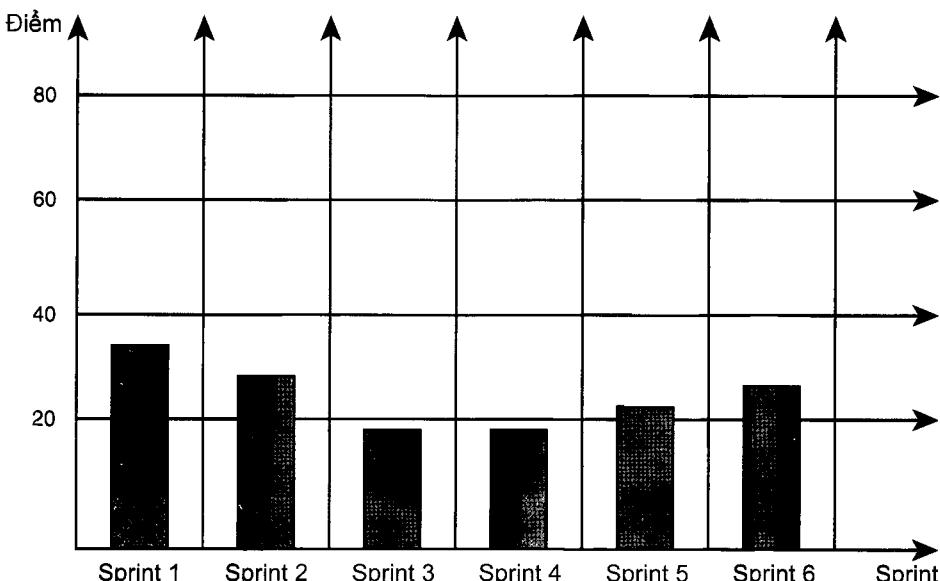
Bây giờ, hãy quay lại điều chúng ta đã thảo luận ở đầu chương này, một trong những điều thú vị nhất mà chúng tôi đã quan sát được trong những dự án Scrum khác nhau là tốc độ nhóm (số lượng User story mà nhóm dự án có thể chuyển giao trong mỗi Sprint) thường dao động theo thời gian. Và dự án càng phức tạp, thì tốc độ nhóm càng dao động, thường dẫn tới việc giảm tốc độ nhóm và ảnh hưởng tới khả năng chuyển giao của họ.

Vấn đề này thường có nhiều lý do, nhưng một trong những lý do chính, từ góc độ kiến trúc, là nhóm thường bận tâm để hình dung hoặc chỉnh sửa kiến trúc phần mềm trong khi chạy dự án. Điều đó dẫn tới, tốc độ của họ thường dao động như ở Hình 6.1.

Nếu bạn đang xây dựng một website hoặc một ứng dụng nhỏ (có thể ví như việc xây dựng một ngôi nhà cho chó), thì việc chỉnh sửa này có thể không mất nhiều thời gian, nhưng nếu bạn đang xây dựng một ứng dụng lớn, thì những chỉnh sửa khi đang thực hiện dự án có thể mất nhiều công sức và thời gian. Những chỉnh sửa này cũng có thể dẫn tới việc cần phải thiết kế lại và chạy lại tất cả kiểm thử ở một số chức năng đã chuyển giao trước đó. Các kiểm thử cũng có thể cần thay đổi so với trước đó để có thể chạy lại được.

Mới chỉ vài năm trước thì nhiều nhóm phần mềm thấy rất thích thú với việc vẽ các sơ đồ kiến trúc và thiết kế sử dụng tất cả những công cụ hiện đại nhất và dùng UML cho Phân tích và Thiết kế Hướng Đối tượng (Object-Oriented Analysis and Design - OOAD). UML là một kỹ thuật tốt dành cho OOAD, nhưng đôi khi, có quá nhiều sự dư thừa, đặc biệt khi tất cả những công việc nặng nhọc đó chỉ để làm đẹp tài liệu và không đủ để có được phần mềm chạy tốt.

Nhưng việc hoàn toàn không thiết kế gì, ngay cả ý đồ, thì sẽ chỉ có hại cho năng suất và hiệu quả của nhóm và có thể nhiều hơn thế.



Hình 6.1

Sự dao động của tốc độ do thiếu ý đồ kiến trúc.

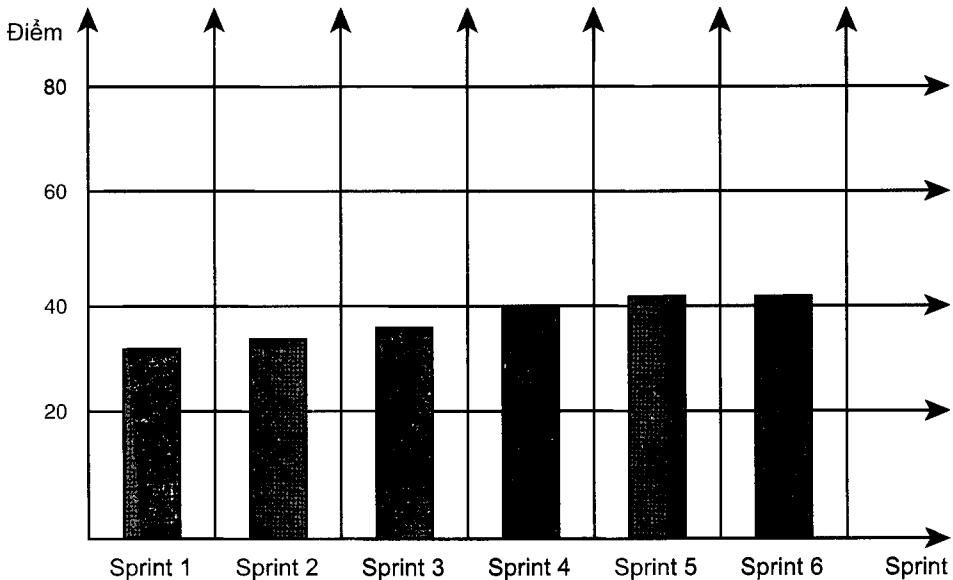
TẦM QUAN TRỌNG CỦA TẦM NHÌN KIẾN TRÚC

Dù triển khai Scrum hay không thì bây giờ chắc bạn cũng đoán rằng tầm nhìn kiến trúc là yếu tố cốt lõi trong phát triển phần mềm, mặc dù bạn phải xây dựng nó trong quá trình phát triển.

Khi có tầm nhìn kiến trúc rõ ràng (hay một số người còn gọi là ý đồ kiến trúc), thì nhóm có thể sẽ chậm ở khi bắt đầu một bản phát hành hoặc Sprint. Tuy nhiên, tốc độ của họ thường khôi phục lại nhanh chóng và giữ ở mức khá tốt như ở Hình 6.2. Điều này tăng sự tự tin và khả năng của nhóm để làm việc trong thời gian dài.

Từ góc độ kỹ thuật, khi không có tầm nhìn kiến trúc, nhà phát triển thường nhận một lượng lớn User story từ Product Owner (Hình 6.3) để xử lý mà không có bất kỳ ý tưởng về sản phẩm cuối hoặc thậm chí cả cách ghép nối những thành phần khác nhau.

Từ đó, nguyên nhân khiến tốc độ nhóm thay đổi quá nhiều là do họ bị lôi kéo vào việc cố gắng di chuyển tất cả những hình chữ nhật nhỏ (đại diện cho các User story), thường là từng cái một vào vị trí phù hợp. Hình 6.4, 6.5, 6.6 và 6.7 minh họa thời gian và công sức mà nhóm cần bỏ ra khi không có tầm nhìn kiến trúc.



Hình 6.2

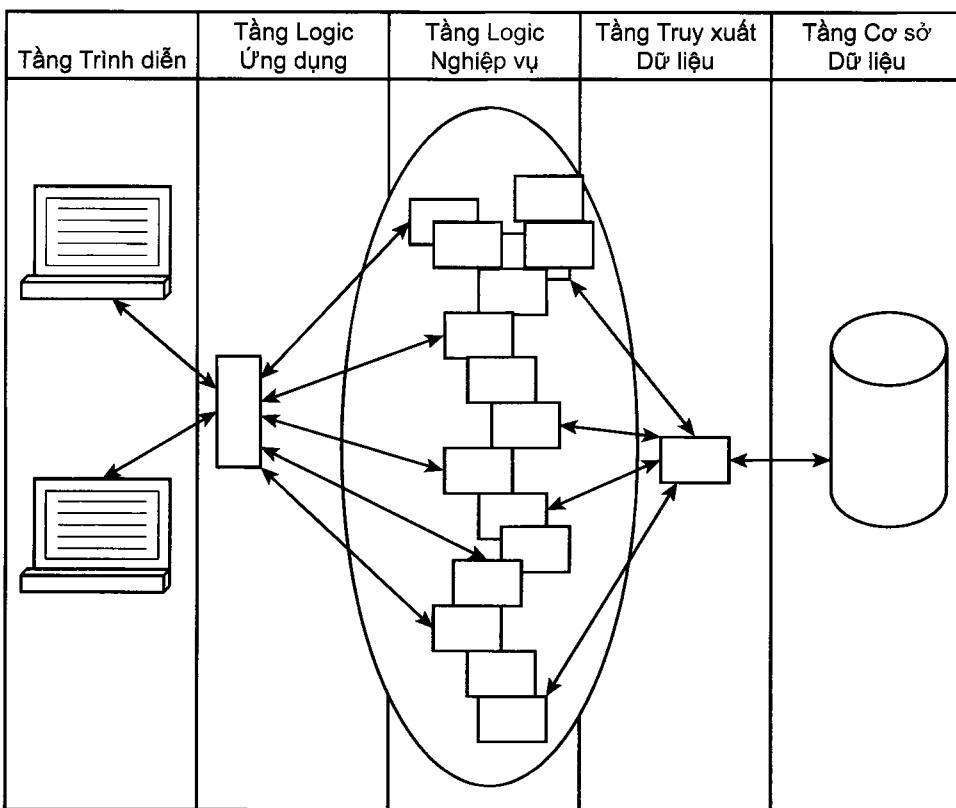
Tốc độ tăng và đồng nhất hơn khi có tầm nhìn kiến trúc.

Điều thú vị mà chúng ta có thể quan sát thấy ở đây đó là công sức của nhóm cuối cùng sẽ cho ra một kiến trúc (Hình 6.7) giống với kiến trúc được phác họa ra trong trường hợp họ dành một ít thời gian ban đầu để hình dung ra tầm nhìn kiến trúc trước và không vội đưa tất cả những User story đã được phân tích chi tiết vào các Sprint.

LÀM THẾ NÀO ĐỂ XÁC ĐỊNH TẦM NHÌN KIẾN TRÚC

Nhóm có thể phải thực hiện hai điều để đạt được tầm nhìn kiến trúc. Một trong số đó là xem lại tầm nhìn và mục tiêu của sản phẩm để xác định các dữ liệu nghiệp vụ chính, thứ sẽ hỗ trợ tầm nhìn sản phẩm, hoặc liên quan tới lĩnh vực nghiệp vụ đó, như một số chuyên gia đã đặt ra.

Một cách khác là sử dụng kỹ thuật thu thập yêu cầu trực quan đã được trình bày trong Chương 4, "Thu thập các yêu cầu trực quan cho Product Backlog", để quan sát những User Story nào chia sẻ cùng nhóm dữ liệu nghiệp vụ. Ví dụ, tất cả các Story đặt mua sách cần có một số dữ liệu chung liên quan tới cuốn sách đó và các đặc điểm của cuốn sách.



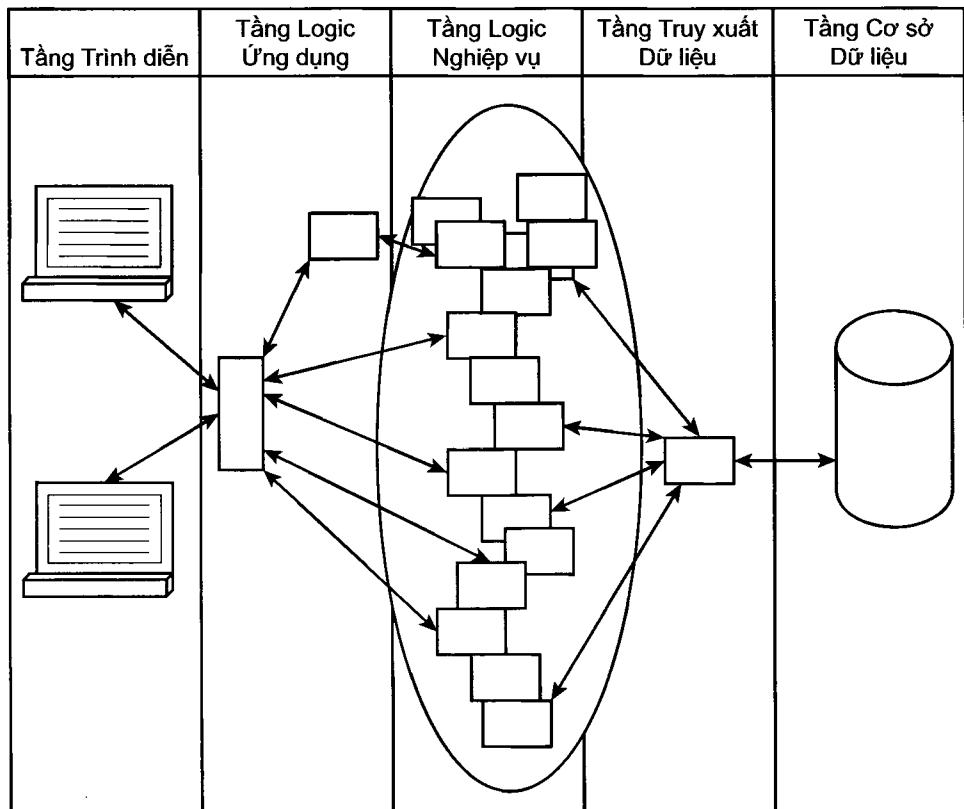
Hình 6.3

Viết mã ứng dụng mà không có kế hoạch hoặc ý đồ về kiến trúc.

Điều này không có nghĩa rằng tất cả các User story có cùng dữ liệu sẽ không cần xây dựng cùng nhau, nhưng ít nhất chúng sẽ là những ứng viên để có thể xây dựng cùng nhau.

Theo chiến lược này, nhóm sẽ có thể cần xác định những User story có khả năng nhóm lại dựa trên nghiệp vụ chung, nhờ đó mà có được các thành phần dữ liệu chung, ổn định như thể hiện trong Hình 6.8, với những thành phần cốt lõi nằm trong vòng tròn trong cùng.

Nhóm sẽ muốn phân chia các Story thêm nữa sau lần phân nhóm đầu tiên này, để phân chia những Story thực hiện công việc tạo mới ra khỏi những Story thực hiện công việc cập nhật, đọc hoặc xóa. Điều này được minh họa trong Hình 6.9.



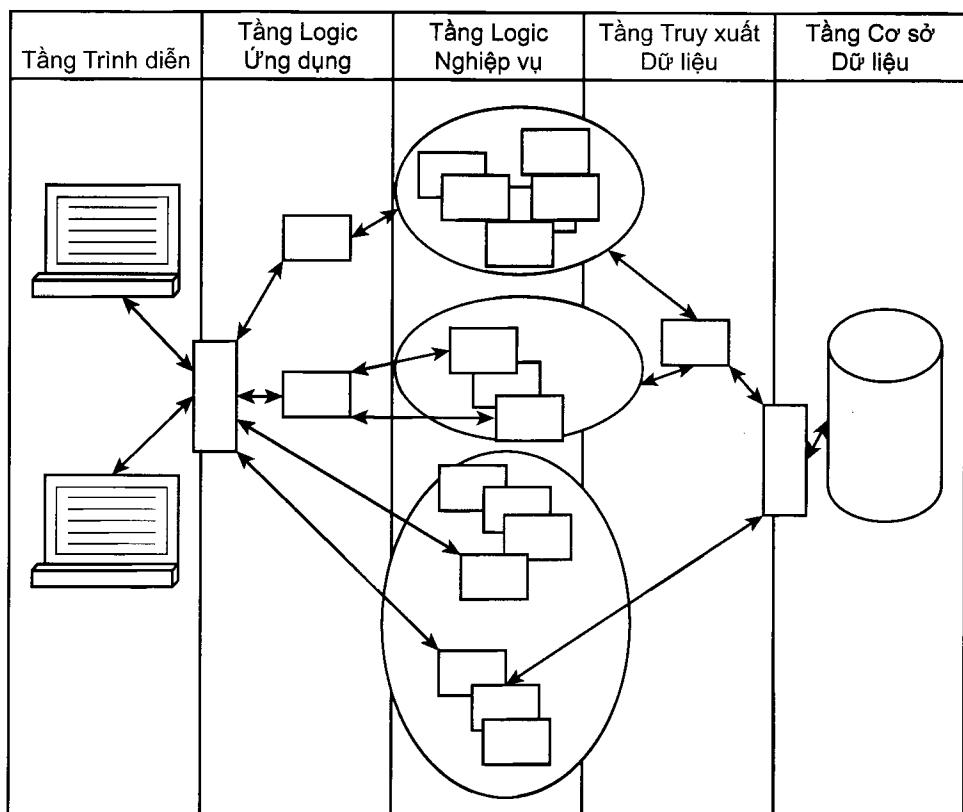
Hình 6.4

Tìm ra kiến trúc trong quá trình triển khai Sprint 2.

Nếu muốn, nhóm có thể tiếp tục phân chia các User story thêm một bước nữa, giống như Hình 6-10.

Ưu điểm của tất cả những nỗ lực bổ sung này là cuối cùng nhóm có thể tổ chức việc phát triển song song hoặc đồng thời giống như minh họa trong Hình 6.11. Điều này tương tự như cách các nhóm tổ chức công việc của họ với kanban, một kỹ thuật tinh gọn nổi tiếng về những đóng góp cho Hệ thống Sản xuất Toyota (Toyota Production System - TPS). Và kỹ thuật này đã thu hút được khá nhiều sự quan tâm trong lĩnh vực phát triển phần mềm, đặc biệt là trong công tác bảo trì và hỗ trợ.

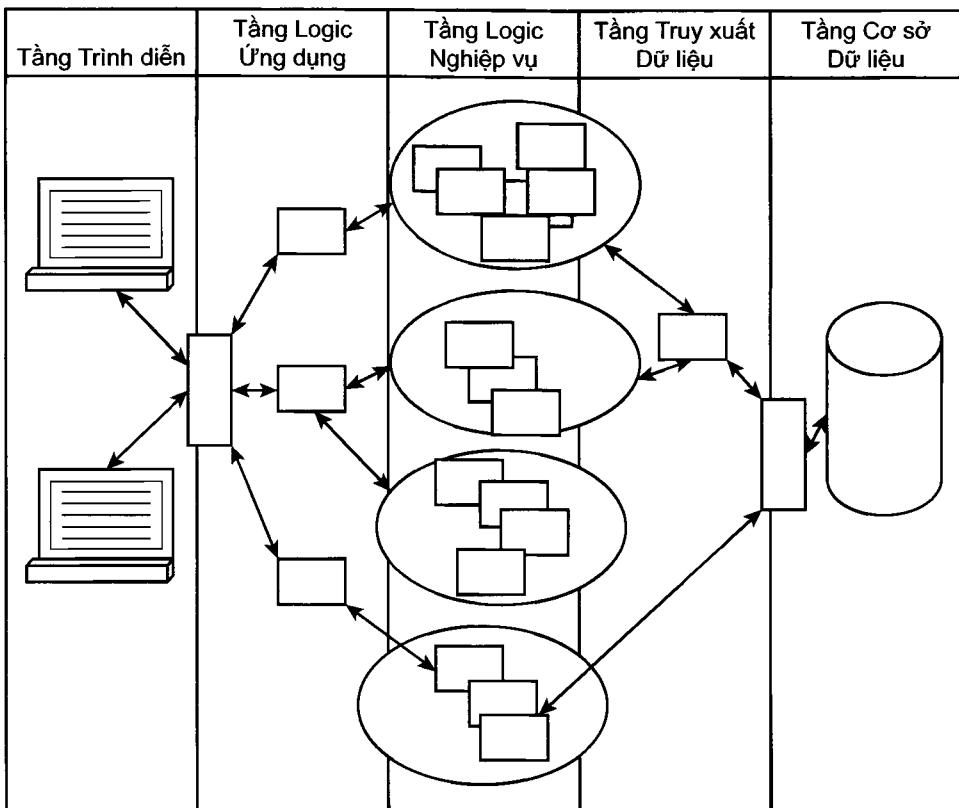
Chương 6 • Ảnh hưởng của Tầm nhìn kiến trúc tới tốc độ Nhóm và Chất lượng phần mềm



Hình 6.5

Tìm ra kiến trúc trong quá trình triển khai Sprint 3.

Ngoài lợi ích này, một lợi thế khác của quản lý công việc dựa trên những thành phần dữ liệu chung là Product Owner có thể đánh giá lại độ ưu tiên, chuyển đổi giữa các User story cụ thể như thể hiện trong Hình 6.12 và 6.13, mà không ảnh hưởng tới tốc độ nhóm do việc đánh giá lại độ ưu tiên này, miễn là các thành phần dữ liệu chính đã được xác định và hoàn tất trước đó. Điều này đúng chỉ cần trước hết nhóm phải quy định nền tảng dữ liệu bằng cách tạo ra tất cả những tạo lập cho mọi thực thể dữ liệu căn bản.



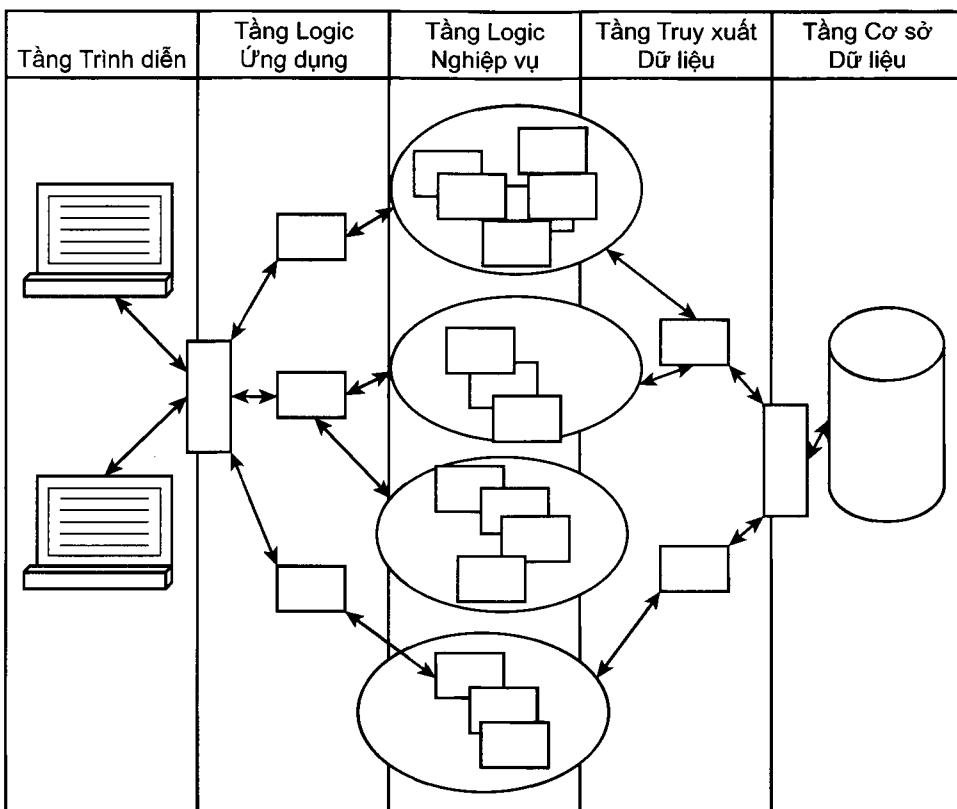
Hình 6.6

Tìm ra kiến trúc trong quá trình triển khai Sprint 4.

LỢI ÍCH KHÁC CỦA VIỆC CÓ TẦM NHÌN KIẾN TRÚC

Bên cạnh những lợi ích kể trên, việc xác định sớm dữ liệu nghiệp vụ chung cũng có thể mang lại thêm lợi ích cho nhóm. Đó là việc giúp nhóm đặt ra một kiến trúc dữ liệu tốt và ổn định, đây là điều cần thiết để tạo ra báo cáo kho dữ liệu. Trừ khi công ty bạn chỉ sử dụng một ứng dụng, còn không tầm nhìn kiến trúc sẽ giúp ứng dụng dễ dàng phù hợp với kiến trúc dữ liệu doanh nghiệp và các ứng dụng còn lại của công ty.

Chương 6 • Ảnh hưởng của Tầm nhìn kiến trúc tới tốc độ Nhóm và Chất lượng phần mềm

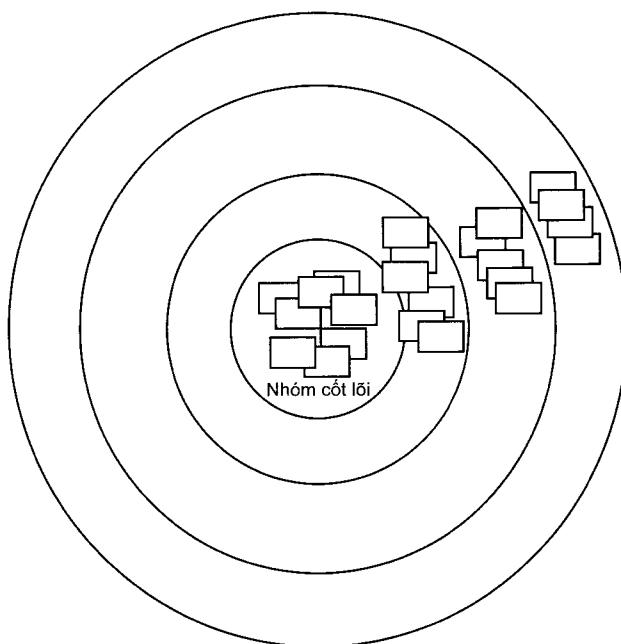


Hình 6.7

Tìm ra kiến trúc trong quá trình triển khai Sprint 5.

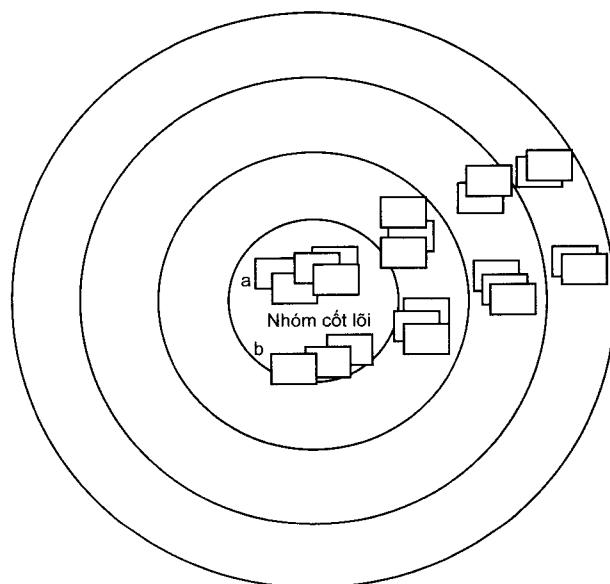
Giống như ở trên, bạn có hai phương pháp có thể giúp tạo ra kiến trúc dữ liệu cho ứng dụng thư viện. Trước tiên là triển khai các User story theo chiều ngang giống như Hình 6.14 và đặc biệt là Hình 6.15. Hình 6.15 mở rộng thêm cho tầng kiến trúc dữ liệu đầu tiên (trong Hình 6.14) bằng cách bổ sung thêm các phân loại dành cho những loại dữ liệu đã có, chẳng hạn thêm các phân loại cho Book (Sách) là “Teen Book (Sách cho Thanh thiếu niên)” và “Toddler Book (Sách cho Trẻ em)” và bổ sung thêm các phân loại cho Patron (Khách hàng) là “Woman (Phụ nữ)” và “Girl (Thiếu nữ)”.

Một cách làm khác đó là triển khai theo chiều dọc, như minh họa trong Hình 6.16 và 6.17, mỗi Sprint sẽ tiến hành triển khai một loại thực thể dữ liệu mới. Trước tiên bạn sẽ triển khai với thực thể Book (Sách) trong suốt Sprint đầu, sau đó tiến hành với thực thể Patron (Khách hàng) trong Sprint thứ hai và cuối cùng là thực thể Day (Calendar – Lịch làm việc) với Sprint thứ ba và cứ như vậy.



Hình 6.8

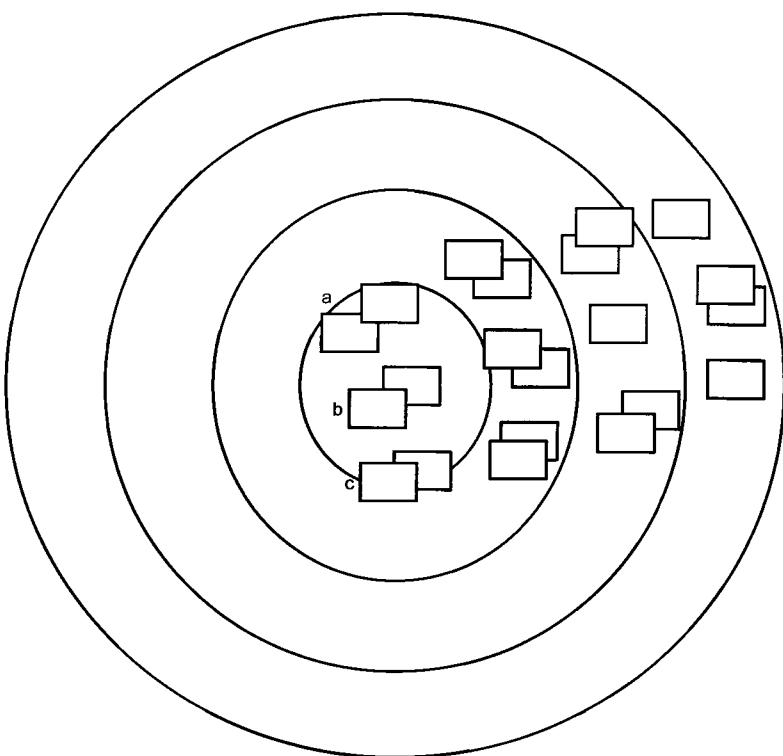
Phân nhóm các User story/PBI dựa trên việc xác định dữ liệu chung.



Hình 6.9

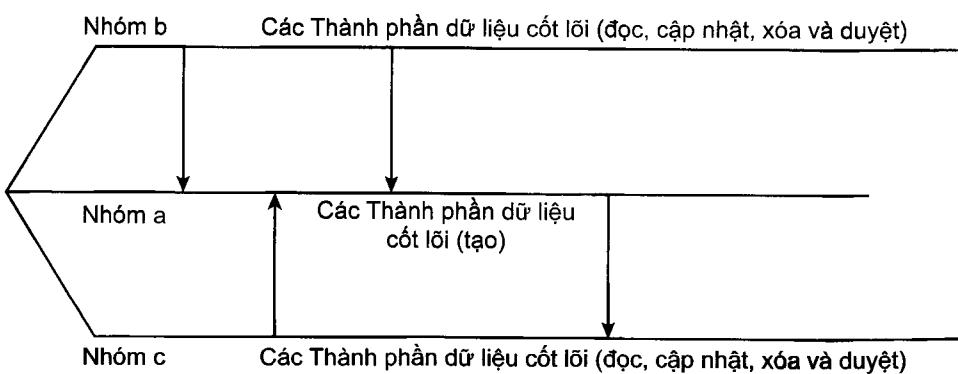
Phân chia các User story dựa trên dữ liệu nghiệp vụ chung thành những cụm tính năng nhỏ hơn.

Chương 6 ▪ Ảnh hưởng của Tầm nhìn kiến trúc tới tốc độ Nhóm và Chất lượng phần mềm



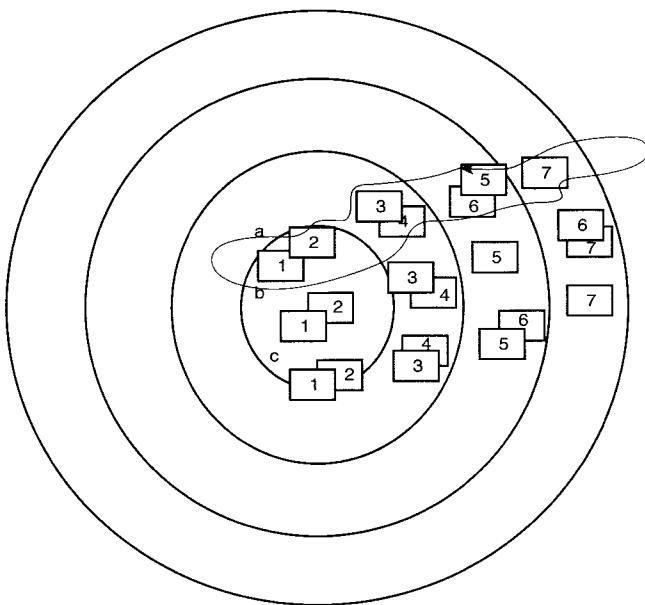
Hình 6.10

Phân chia các User story dựa trên dữ liệu nghiệp vụ chung thành những cụm tính năng nhỏ hơn nữa.



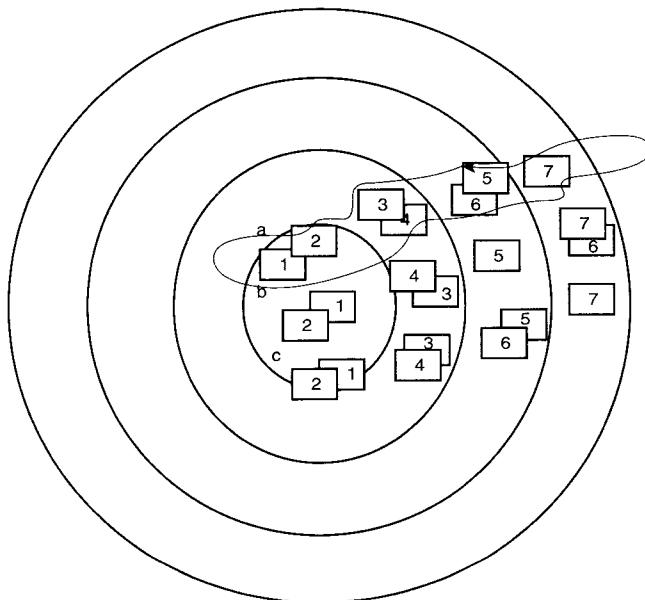
Hình 6.11

Triển khai đồng thời các User story/PBI.



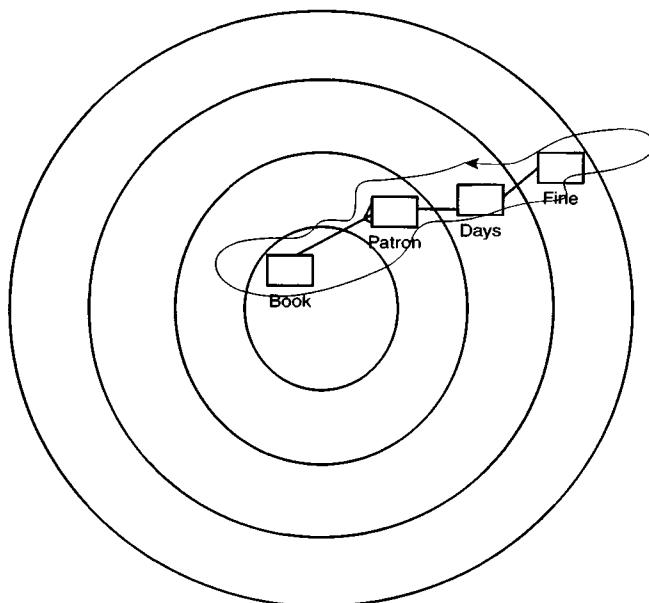
Hình 6.12

Thay đổi mức ưu tiên sản phẩm.



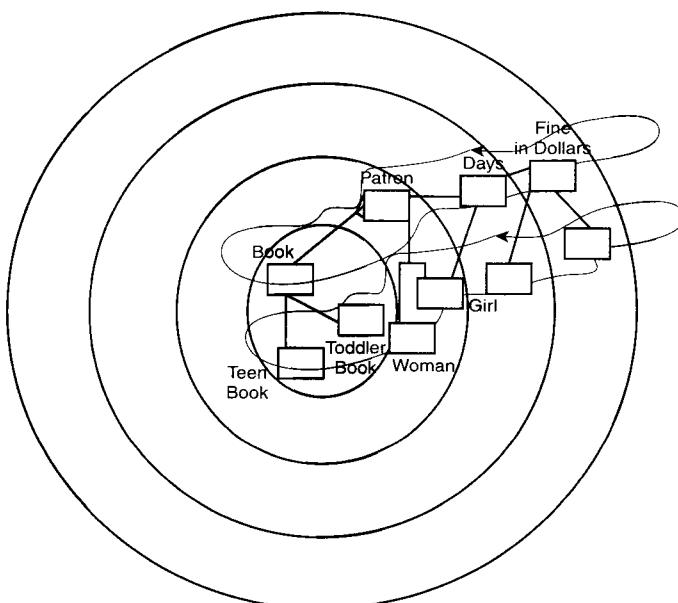
Hình 6.13

Thay đổi mức ưu tiên sản phẩm không gây ảnh hưởng tới việc làm việc nhóm.



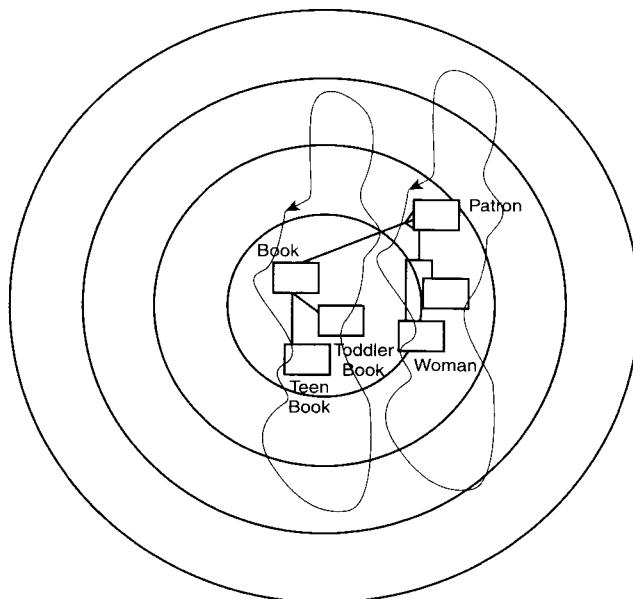
Hình 6.14

Kiến trúc dữ liệu khởi đầu đang trong quá trình hình thành.
(Tham khảo bảng chú thích dịch hình ở cuối chương này).



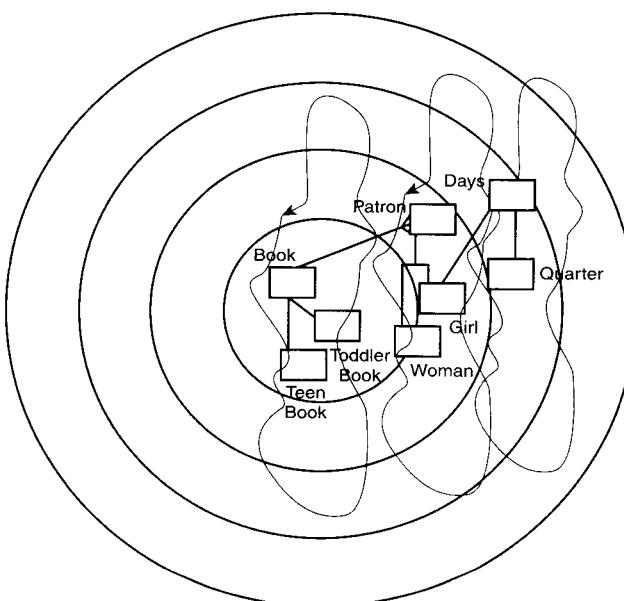
Hình 6.15

Kiến trúc dữ liệu theo chiều ngang đang trong quá trình hình thành.



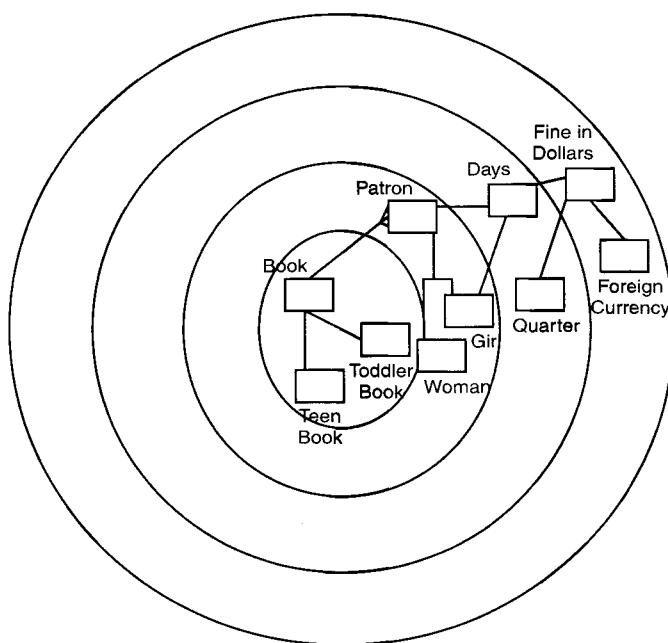
Hình 6.16

Kiến trúc dữ liệu theo chiều dọc đang trong quá trình hình thành.



Hình 6.17

Kiến trúc dữ liệu đang trong quá trình hình thành.



Hình 6.18

Kiến trúc dữ liệu cuối cùng của ứng dụng.

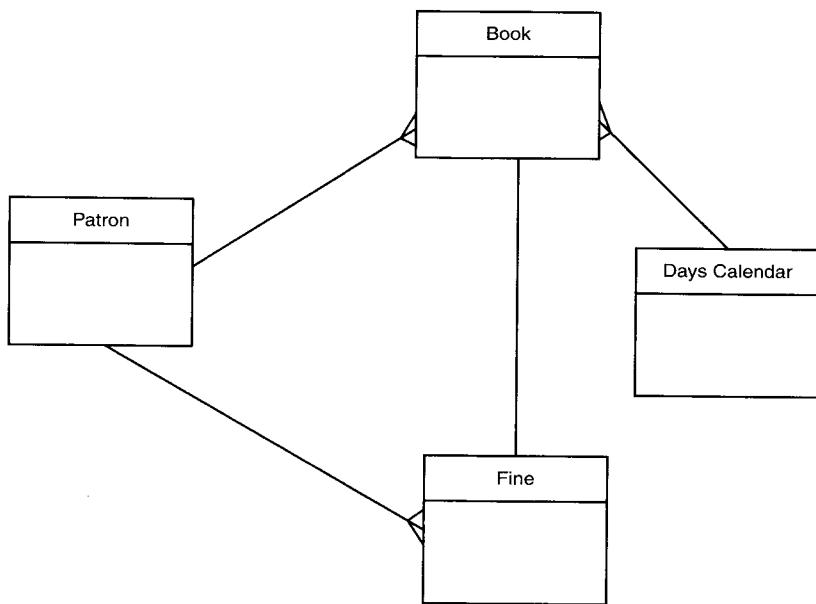
Cả hai chiến lược này sẽ hình thành ra mô hình dữ liệu cuối cùng được minh họa trong Hình 6.18.

Khi xem xét kỹ hơn, chúng ta có thể giả định rằng mô hình dữ liệu trong Hình 6.18 sẽ có tất cả những quan hệ và các số lượng thực thể tham gia vào quan hệ như mô hình dữ liệu trong Hình 6.19.

Hình 6.20 minh họa phần mở rộng của mô hình dữ liệu giao dịch trước đây với hai phân loại thực thể con mới là Teenager (Thanh thiếu niên) và Adult (Người lớn), được thêm vào cho thực thể Patron (Khách hàng).

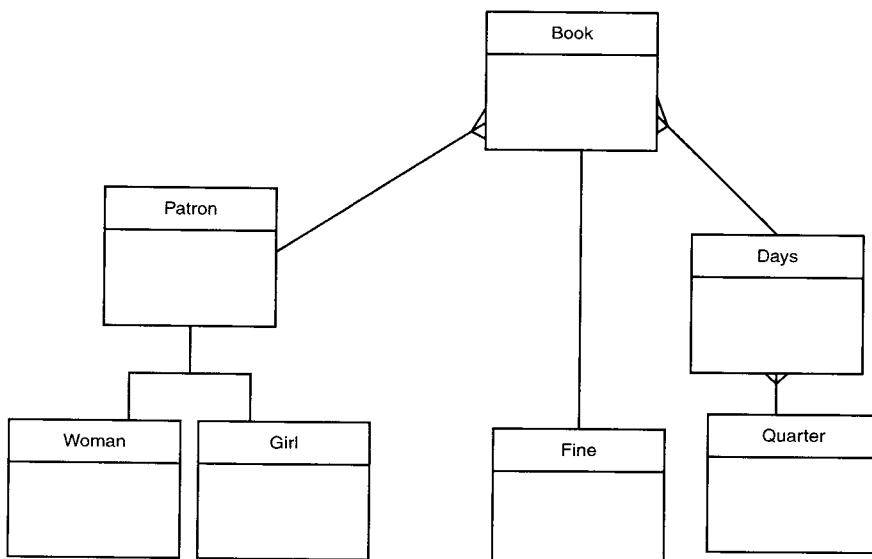
Hình 6.21 minh họa cách bạn có thể xây dựng lược đồ hình sao cho ứng dụng này. Trước hết bạn bổ sung bảng sự kiện gọi là Visit (Ghé thăm) vào trung tâm của mô hình. Sau đó bạn bao quanh nó với các bảng Quarter (Quý), Patron (Khách hàng) và Library Location (Thư viện), những bảng này được gọi là Các khía cạnh liên quan và là những thành phần có được từ mô hình dữ liệu giao dịch mô tả trong Hình 6.20.

Hình 6.22 cho thấy ứng dụng của bạn cuối cùng sẽ phù hợp với kiến trúc dữ liệu doanh nghiệp tổng thể của công ty như thế nào. Thay vì các ứng dụng được xây dựng theo cách mà bạn chỉ có thể hy vọng rằng chúng sẽ phù hợp với kiến trúc doanh nghiệp, nỗ lực của bạn sẽ tạo nên một loại tầm nhìn kiến trúc có thể đảm bảo rằng hệ thống của bạn sẽ phù hợp hoàn toàn với kiến trúc doanh nghiệp CNTT của công ty.



Hình 6.19

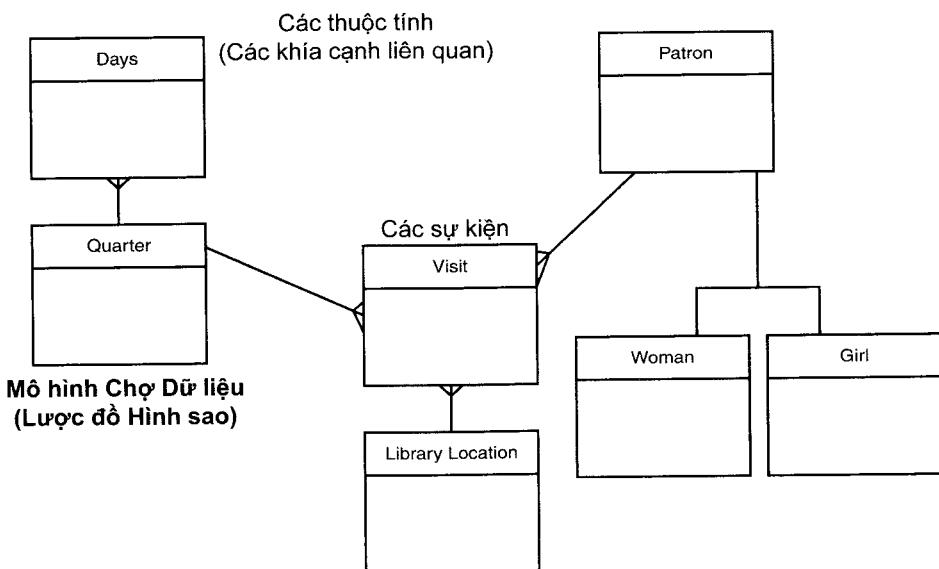
Mô hình dữ liệu giao dịch.



Hình 6.20

Mở rộng mô hình dữ liệu giao dịch bao gồm cả Woman và Girl.

Chương 6 • Ảnh hưởng của Tầm nhìn kiến trúc tới tốc độ Nhóm và Chất lượng phần mềm

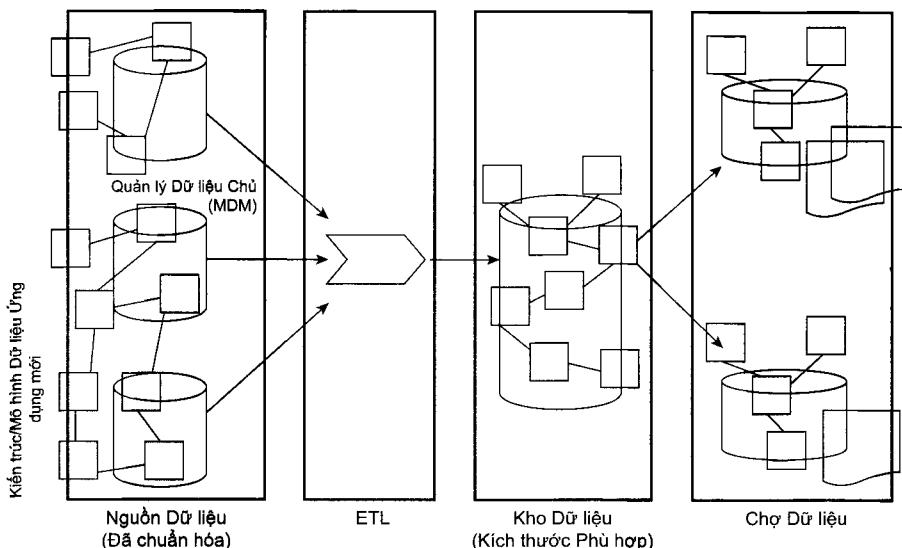


Hình 6.21

Lược đồ hình sao chợ dữ liệu.

Bảng chú thích cho Hình 6.15 đến 6.21

Book	Sách
Teen Book	Sách cho Thanh thiếu niên
Toddler Book	Sách cho Trẻ em
Patron	Khách hàng
Girl	Thiếu nữ
Woman	Phụ nữ
Days	Ngày
Quarter	Quý
Fine	Phạt
Foreign Currency	Ngoại tệ
Visit	Ghé thăm
Library Location	Vị trí thư viện

**Hình 6.22**

Sự phù hợp của kiến trúc dữ liệu doanh nghiệp mới.

TÓM TẮT

Là các chuyên gia phần mềm, nhiều người trong số chúng ta đã quen với việc sử dụng tất cả các loại công cụ để tạo ra nhiều bản thiết kế về kiến trúc. Nhưng khi XP¹ xuất hiện, nó đã khiến chúng ta thấy chỉ nên thiết kế khi chúng ta làm. Thành công của Scrum dường như đã khuếch trương cho hiện tượng này.

Bỗng dưng, dường như không ai còn muốn bàn thêm về thuật ngữ kiến trúc hoặc thiết kế nữa và chúng được xem như chỉ tồn tại trong quá khứ.

Nhưng khi các bài học bắt đầu xuất hiện từ những dự án Scrum, một điều đã trở nên rõ ràng rằng nếu không có một tầm nhìn kiến trúc những User story mà nhóm có được từ Product Owner sẽ chỉ là một tập các tấm thẻ và sẽ không có một ý tưởng rõ ràng nào để hình dung ra sản phẩm cuối cùng.

Như một hệ quả, nhóm sẽ tiêu tốn thời gian và nỗ lực để có được những Story khác nhau nhằm phù hợp với một số dạng hình ảnh mù mờ của một kiến trúc luôn luôn thay đổi.

Với một tầm nhìn kiến trúc rõ ràng đã được xác định từ đầu, các quan sát cho thấy tốc độ nhóm không dao động nhiều, nhưng thay vào đó là sự gia tăng thời gian ngoài giờ. Các thành viên trong nhóm cảm thấy tin tưởng hơn vào khả năng chuyển giao của họ.

¹ XP: eXtreme Programming (Lập trình Cực hạn) - là một trong những phương pháp phát triển phần mềm linh hoạt

Chương 6 ▪ Ảnh hưởng của Tầm nhìn Kiến trúc tới Tốc độ Nhóm và Chất lượng Phần mềm

Để đi tới tầm nhìn kiến trúc, trước tiên chúng ta phải xác định các User story chia sẻ dữ liệu nghiệp vụ chung và xem xét triển khai chúng cùng nhau. Thông qua việc phân tách các User story theo thành phần dữ liệu nghiệp vụ chung, nhóm có thể tổ chức công việc của mình và tiến hành phát triển song song. Đây là những điều mà nhiều nhà quản lý phần mềm mơ ước bởi vì họ nhìn thấy lợi ích tiềm tàng đối với hiệu suất của nhóm.

Làm việc với dữ liệu nghiệp vụ chung cũng cho phép nhóm tạo nên một loại kiến trúc dữ liệu ổn định, thành phần quan trọng để có được những phân tích và báo cáo dữ liệu tốt. Một lợi ích khác đó là cho phép Product Owner thay đổi thứ tự ưu tiên của dự án mà không ảnh hưởng tiêu cực tới tốc độ nhóm.

CHƯƠNG 7

TỪ TẦM NHÌN KIẾN TRÚC ĐẾN LẬP KẾ HOẠCH PHÁT HÀNH, LẬP KẾ HOẠCH SPRINT VÀ PHÁT TRIỂN PHẦN MỀM SONG SONG



Bạn đã tìm hiểu về lợi ích của việc có một tầm nhìn kiến trúc sớm, không chỉ đối với khía cạnh tốc độ mà còn đối với chất lượng phần mềm. Và không chỉ dừng lại ở đó, việc có một tầm nhìn kiến trúc sớm còn tiếp tục mang đến nhiều lợi ích khác trong việc lập kế hoạch phát hành và lập kế hoạch Sprint.

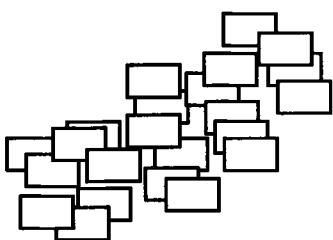
Khi được trang bị kiến thức này, bạn sẽ làm việc tốt hơn với Product Owner để đưa ra các gợi ý nhằm điều chỉnh chuỗi công việc. Những điều chỉnh nhằm mang lại giá trị kinh doanh cao nhất. Cùng lúc đó, bạn có thể duy trì một tốc độ tốt và dần xây dựng một nền tảng vững chắc cho ứng dụng và kiến trúc dữ liệu.

Cho dù Product Owner thường muốn tập trung hơn vào khách hàng hoặc các yêu cầu của người dùng, nhưng trong Scrum, một Product Owner tốt cũng phải tập trung chú ý đến cả các gợi ý từ nhóm phát triển nữa.

TỪ TẦM NHÌN KIẾN TRÚC ĐẾN LẬP KẾ HOẠCH PHÁT HÀNH VÀ LẬP KẾ HOẠCH SPRINT

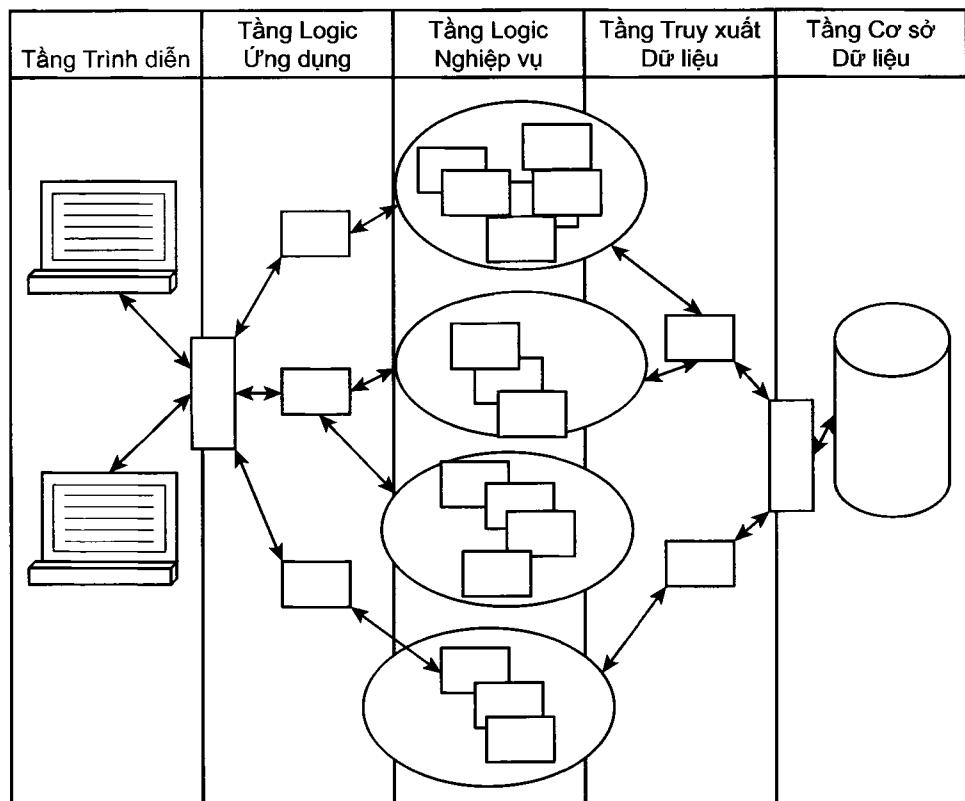
Thay vì thụ động để cho Product Owner quyết định những công việc mà nhóm sẽ làm trong một Sprint nhất định, kinh nghiệm cho thấy rằng nhóm sẽ có thêm nhiều lợi ích nếu họ có thái độ tích cực cộng tác với Product Owner.

Chương 7 - Từ Tầm nhìn kiến trúc đến Lập kế hoạch Phát hành, Lập kế hoạch Sprint và Phát triển Phần mềm song song



Hình 7.1

Thanh tra tập các User Story cho một hệ thống thư viện trung tâm.



Hình 7.2

Xác định tầm nhìn kiến trúc cho hệ thống thư viện trung tâm.

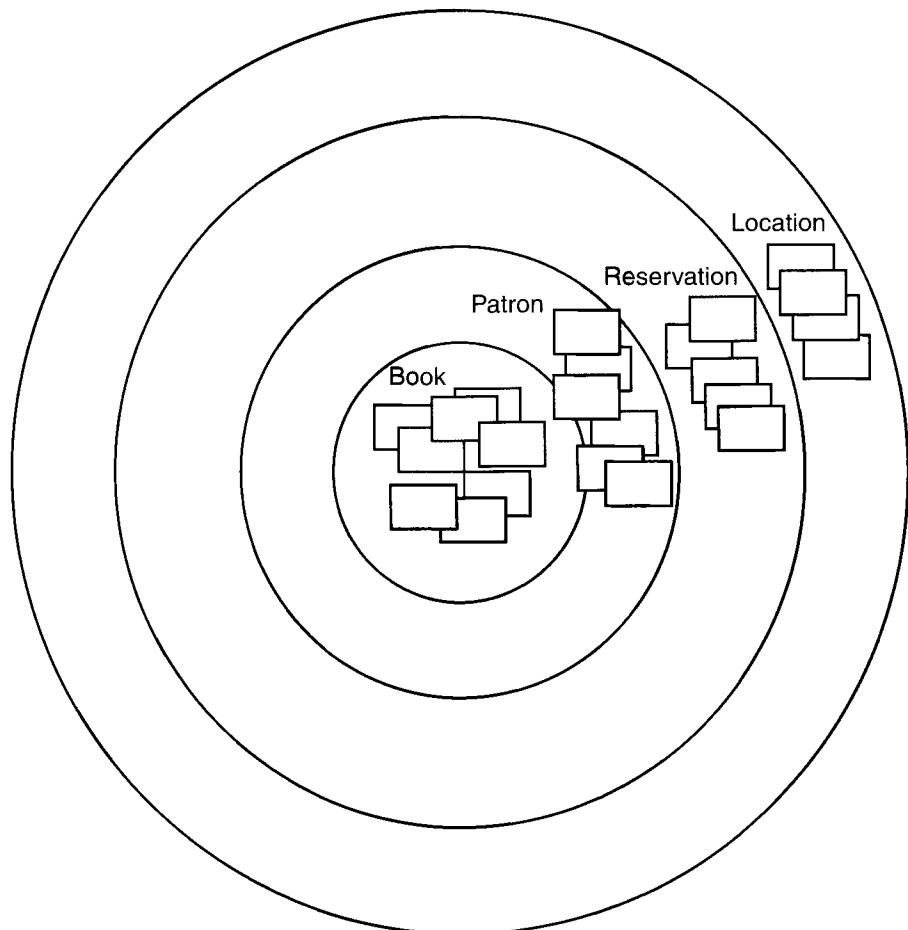
Nhóm cần phải có một cái nhìn tích cực với tập các Story trước khi lập kế hoạch nhằm xem thử chúng có dẫn đến một tầm nhìn kiến trúc hay không.

Giả định rằng sau một cuộc họp thì tập các User story đã được xác định như trong Hình 7.1.

Từ Tầm nhìn Kiến trúc đến Lập kế hoạch Phát hành và Lập kế hoạch Sprint

Nhóm có thể nhìn lướt qua các User story đó và có khả năng sẽ xác định được một kiến trúc ở mức tổng quan giống như được thể hiện trong Hình 7.2.

Tiếp theo, bằng cách sử dụng cách tiếp cận dựa trên các thành phần dữ liệu chung như đã được đề cập trong Chương 6, “*Ảnh hưởng của tầm nhìn kiến trúc đến tốc độ nhóm và chất lượng phần mềm*”, nhóm có thể phân chia các User story thành các vòng khác nhau như trong Hình 7.3.

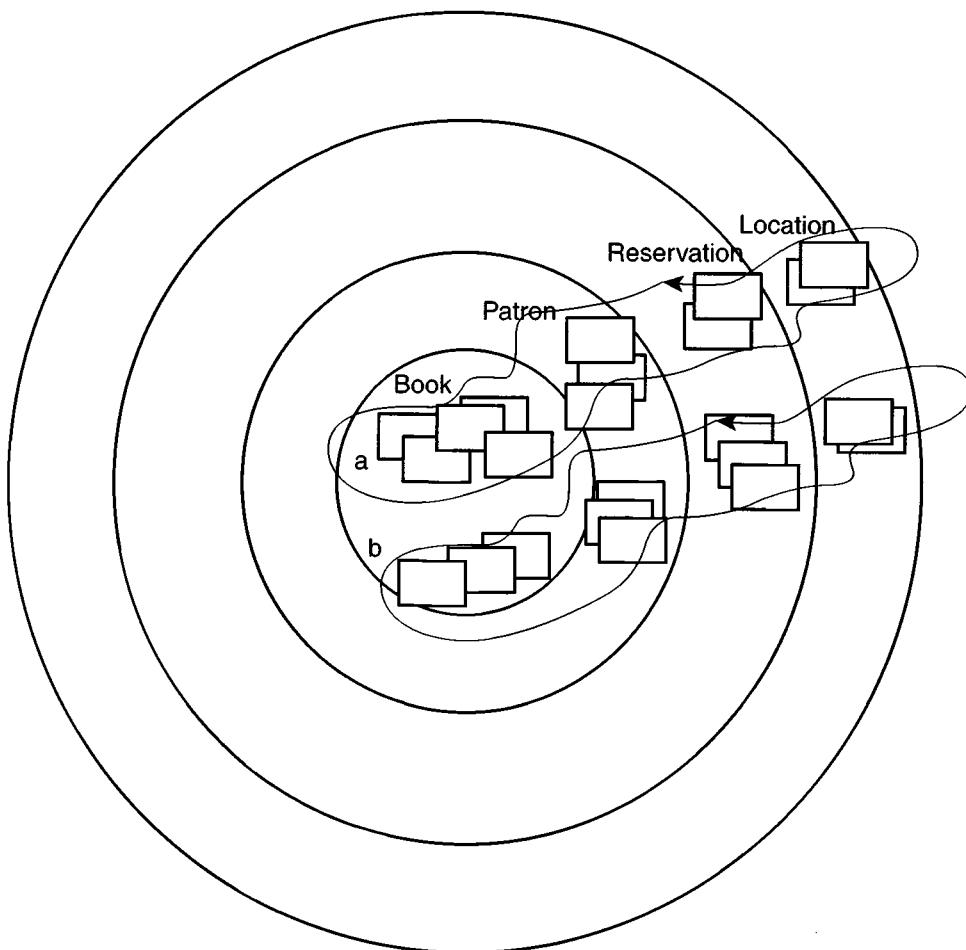


Book	Sách
Patron	Khách hàng
Reservation	Đặt chỗ
Location	Địa điểm

Hình 7.3

Phân chia các User Story theo các thành phần dữ liệu chung.

Chương 7 • Từ Tầm nhìn kiến trúc đến Lập kế hoạch Phát hành, Lập kế hoạch Sprint và Phát triển Phần mềm song song

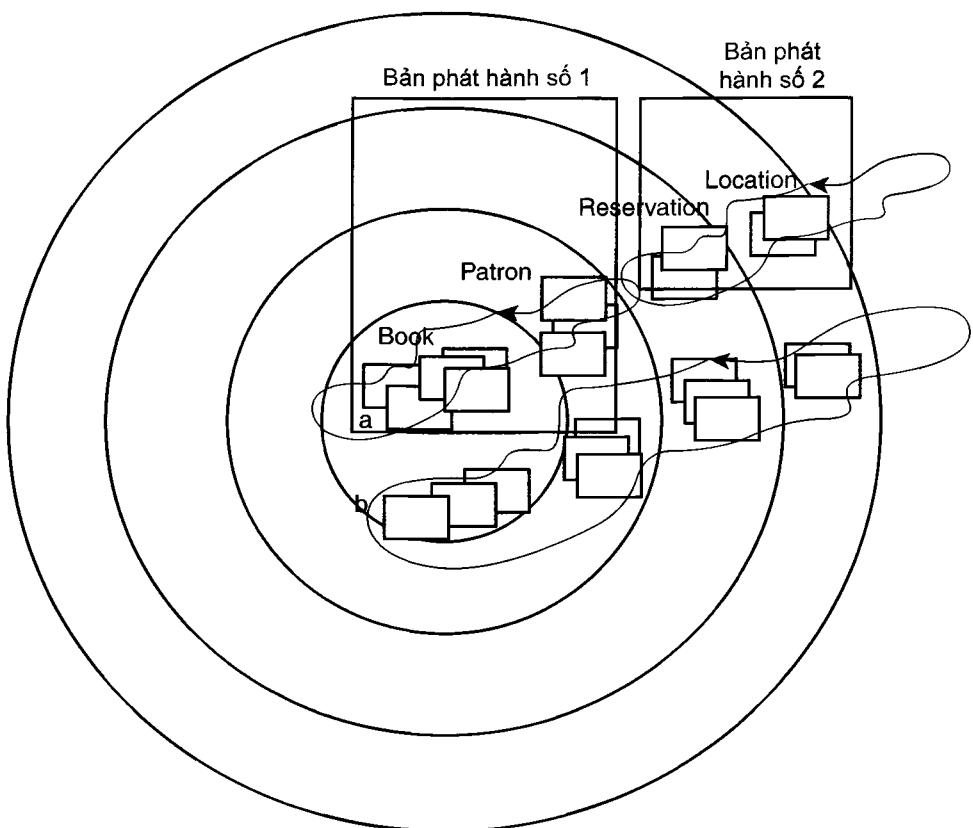


Hình 7.4

Tiếp tục phân chia các User Story theo các thành phần dữ liệu chung.

Bằng việc tiếp tục phân chia thì nhóm có thể có khả năng phân loại các User story thành các nhóm nhỏ hơn. Hoạt động này dẫn đến việc phân tách các thành phần dữ liệu cần phải tạo mới ra khỏi những thành phần dữ liệu chỉ đọc, cập nhật hoặc xóa, như được thể hiện trong Hình 7.4.

Từ đây, sẽ có hai lựa chọn cho nhóm để hình dung bản phát hành, đó là triển khai theo lát cắt ngang hay theo lát cắt dọc.



Hình 7.5

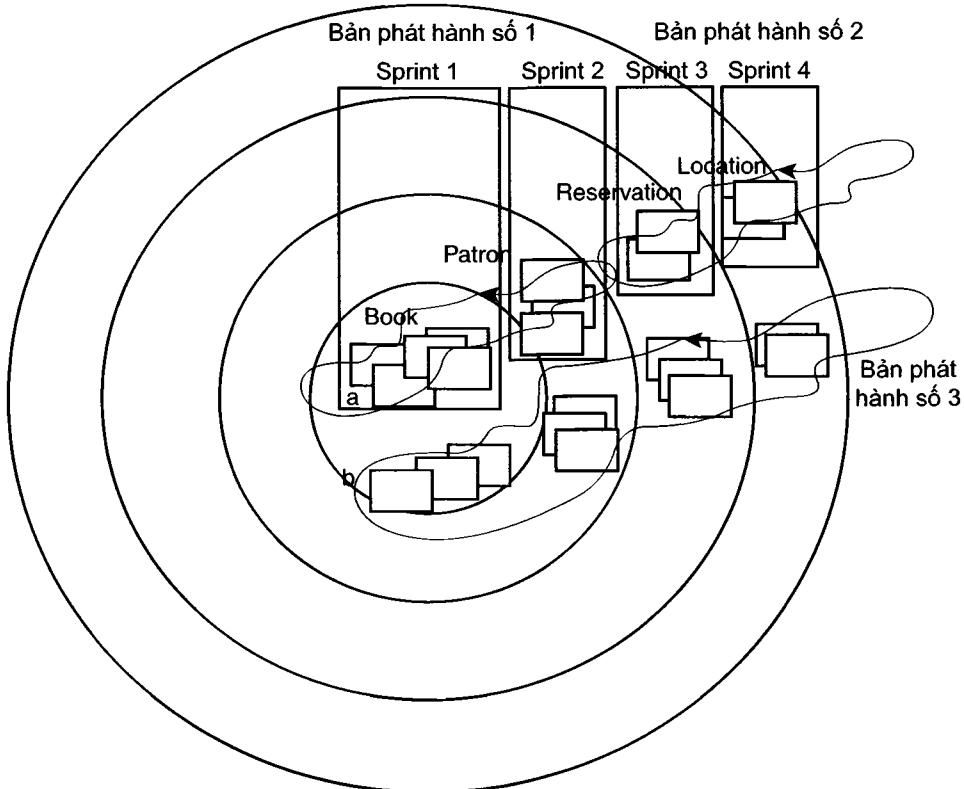
Tổ chức bản phát hành theo lát cắt ngang.

Triển khai theo lát cắt ngang có nghĩa là nhóm có thể hình dung được việc phát triển sản phẩm phần mềm thông qua việc tạo nền tảng cho những thành phần dữ liệu và thực thể quan trọng trên toàn bộ các vòng tròn, bắt đầu từ nhóm cốt lõi ở vòng tròn chính giữa cho đến vòng tròn ngoài cùng.

Trong trường hợp hệ thống thư viện trung tâm, điều này đồng nghĩa với việc trước tiên sẽ xây dựng nền tảng cho các thực thể Book (sách) và Patron (khách hàng) trong bản phát hành số 1, còn các thực thể Reservation (đặt chỗ) và Location (địa điểm) được xây dựng trong bản phát hành số 2, như minh họa trong Hình 7.5.

Tiếp theo, tùy theo tính chất của dữ liệu chung (và có thể cả tốc độ nhóm) mà bản phát hành số 1 có thể được chia thành hai Sprint. Sprint 1 sẽ tập trung vào phần đầu tiên là khái niệm Book (sách), còn Sprint 2 thì tập trung vào phần đầu tiên của khái niệm Patron (khách hàng).

Chương 7 • Từ Tầm nhìn kiến trúc đến Lập kế hoạch Phát hành, Lập kế hoạch Sprint và Phát triển Phần mềm song song

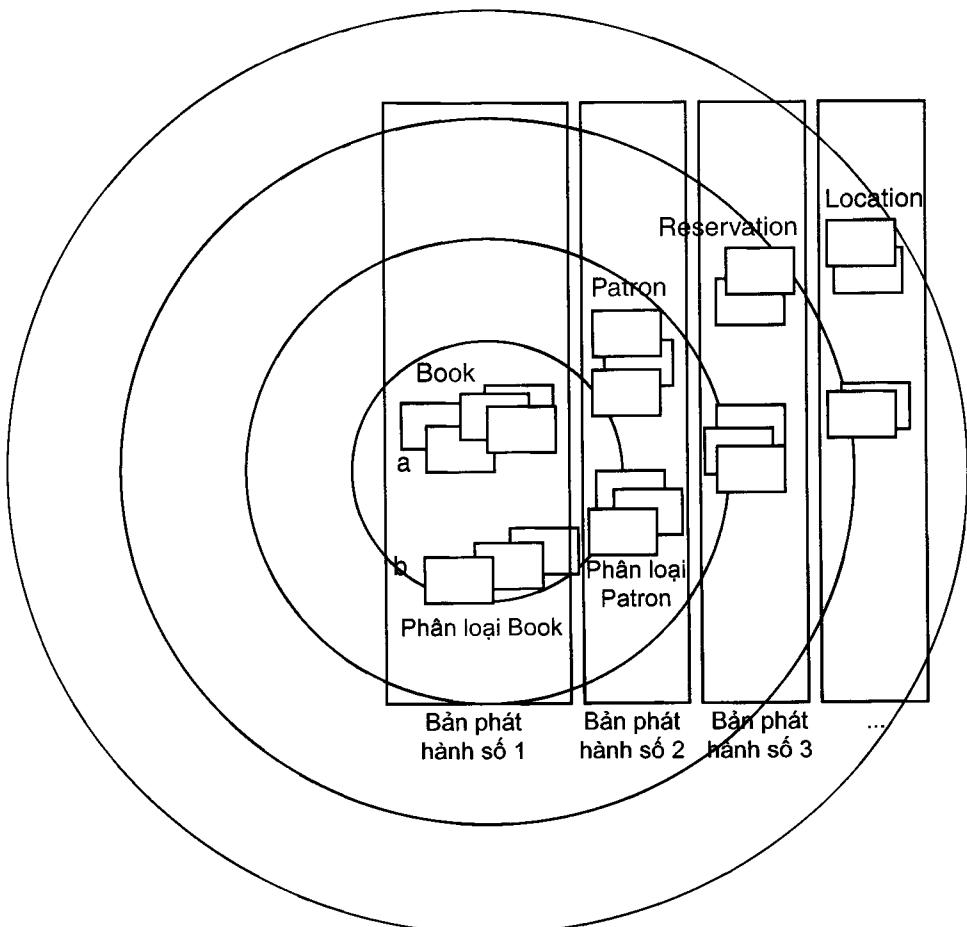


Hình 7.6

Phân chia bản phát hành theo lát cắt ngang thành các Sprint.

Có thể áp dụng cách tiếp cận đó cho bản phát hành số 2. Bản phát hành này cũng được chia thành hai Sprint, Sprint 3 tập trung vào khái niệm Reservation (đặt chỗ) trong khi Sprint 4 tập trung vào khái niệm Location (địa điểm) (Hình 7.6).

Triển khai theo lát cắt dọc có nghĩa là nhóm tiến hành phát triển sản phẩm phần mềm bằng cách xây dựng nền tảng với tất cả các thành phần dữ liệu và thực thể, lần lượt trên từng vòng tròn. Quay lại với hệ thống thư viện trung tâm, điều này có nghĩa là trước tiên nhóm có thể xây dựng nền tảng cho tất cả mọi thứ liên quan đến khái niệm Book ngay ở trong bản phát hành số 1; sau đó, tất cả các chi tiết của khái niệm Patron có thể nằm trong bản phát hành số 2 và cứ như vậy (Hình 7.7).



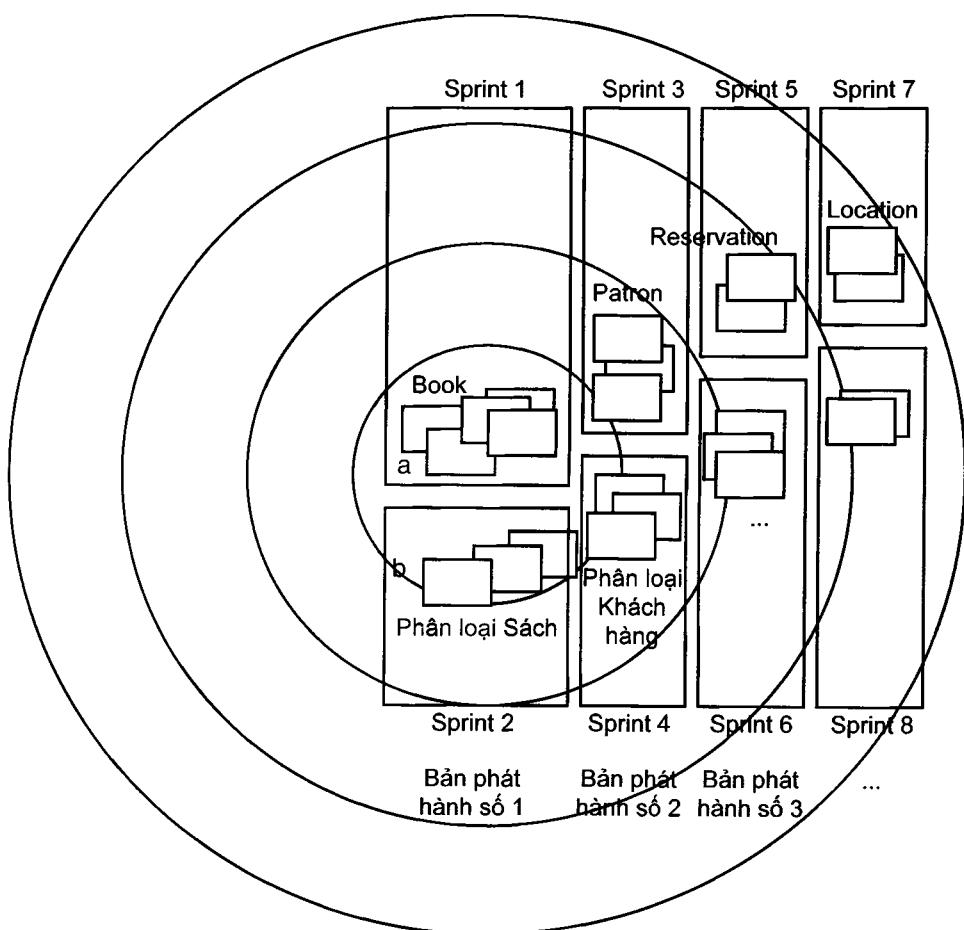
Hình 7.7

Tổ chức bản phát hành theo lát cắt đọc.

Tiếp theo, chúng ta áp dụng cùng một tính chất của dữ liệu chung (và có thể là cả tốc độ của nhóm) cho loại lát cắt thứ hai này. Bản phát hành số 1 (Hình 7.7) tập trung hoàn toàn vào khái niệm Book, có thể được chia làm hai Sprint. Sprint 1 tập trung vào những phần cơ bản của khái niệm Book, trong khi Sprint 2 sẽ tập trung các phần mở rộng của nó đó là các phân loại cho khái niệm Book (phân loại sách).

Có thể áp dụng cách làm này cho bản phát hành số 2, như được minh họa trong Hình 7.7. Bản phát hành này cũng có thể được chia thành hai Sprint (Hình 7.8), Sprint 3 tập trung vào các phần cơ bản của khái niệm Patron trong khi Sprint 4 tập trung các phần mở rộng của nó đó là các phân loại cho khái niệm Patron (phân loại khách hàng). Và cứ tiếp tục như vậy với các bản phát hành tiếp theo (Hình 7.8).

Chương 7 ▪ Từ Tầm nhìn kiến trúc đến Lập kế hoạch Phát hành, Lập kế hoạch Sprint và Phát triển Phần mềm song song



Hình 7.8

Chia bản phát hành ngang thành các Sprint.

Sau một thời gian, bạn sẽ nhận ra rằng mình không theo dõi được toàn bộ mục tiêu của bản phát hành và của Sprint, có thể bạn muốn sắp xếp chúng, kèm theo các mô tả khái quát tương ứng của các Story bằng cách sử dụng một ma trận như trong Hình 7.9.

Ngoài việc mang lại lợi ích cho nhóm Scrum trong việc lập kế hoạch phát hành và lập kế hoạch Sprint thì lược đồ ở Hình 7.9 còn có thể dễ dàng trở thành một bản Sprint Backlog mới và đã được cải tiến (Hình 7.10) để hỗ trợ việc phân công công việc và theo dõi tiến độ của nhóm.

Bản phát hành	Sprint	Story	Mục tiêu
1. Bản phát hành số 1	1. Sprint 1 (Book)	1. Story 1 2. Story 2
2. Bản phát hành số 2	2. Sprint 2 (Patron)	3. Story 3 4. Story 4

Hình 7.9

Theo dõi đối sánh các mục tiêu Sprint và mục tiêu của bản phát hành (triển khai theo lát cắt ngang).

Vì chúng ta đang nói đến việc theo dõi tiến độ của nhóm, cho nên có lẽ chúng ta sẽ muốn làm rõ hơn về một khái niệm được gọi là nợ kỹ thuật (technical debt), đây được xem là sự tích lũy những công việc kỹ thuật cần làm lại nhằm sửa chữa thiết kế của phần mềm.

Xuất phát từ kinh nghiệm, chúng tôi nhận thấy rằng nợ kỹ thuật có mối quan hệ tỷ lệ nghịch với chất lượng phần mềm (Hình 7.11). Do nợ kỹ thuật ẩn giấu đằng sau sự thật là bản phát hành đã bị muộn cho nên chúng tôi hoàn toàn phản đối sự tồn tại của nó.

Nếu bạn báo cáo rằng mình đang kiểm soát tốt việc hoàn thành mục tiêu Sprint, trong khi biết rằng mình đang có một đồng nợ kỹ thuật, thì bạn đang không trung thực với chính mình và với các thành viên khác trong nhóm.

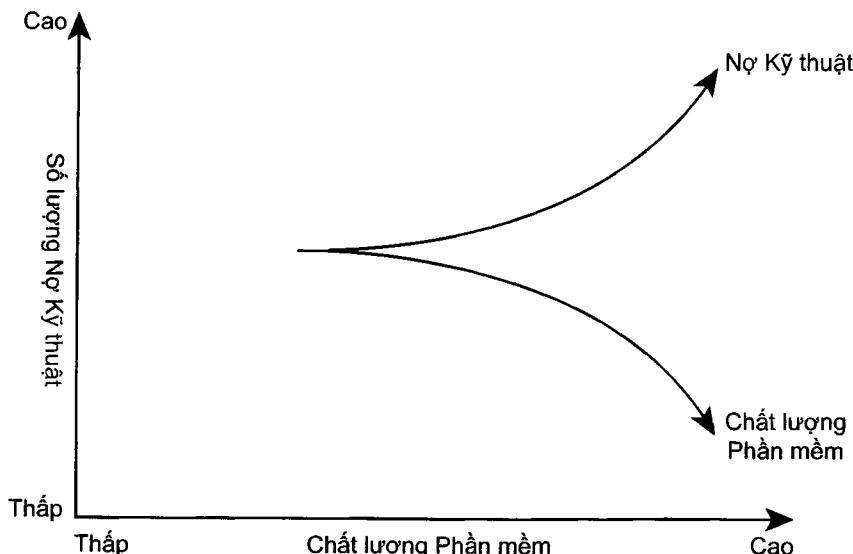
Do đó, bằng bất cứ giá nào, cần phải ngăn cản việc nợ kỹ thuật ảnh hưởng đến việc theo dõi tiến độ thực tế của dự án.

		Lượng công việc còn lại theo giờ								
Backlog	Nhiệm vụ	Thành viên	Ước tính (Giờ)	Ngày 1	Ngày 2	Ngày 3	Ngày 4	Ngày 5	Ngày 6	
Bản phát hành số 1	Mục tiêu:									
1. Sprint 1	Mục tiêu:									
Story 1	.Thiết kế/Viết mã Giao diện người dùng (UI)	John	4	2	2	
	.Thiết kế/Viết mã Tầng ứng dụng	Frank	2	2	1	
	.Thiết kế và Viết mã Tầng nghiệp vụ	Andrew	5	2	2	
	.Thiết kế và Viết mã Tầng truy xuất dữ liệu	Laura	6	5	6	
	.Thiết kế và viết mã Tầng cơ sở dữ liệu	Lola	4	3	3	
	...									
Story n	.Thiết kế/Viết mã Giao diện người dùng (UI)	John	7	5	
	.Thiết kế/Viết mã Tầng Ứng dụng	Andrew	5	5	
								
2. Sprint 2	Mục tiêu:									
Story n+1	.Thiết kế/Viết mã Giao diện người dùng (UI)	John	3	2	1	
								
		Tổng cộng								

Hình 7.10

Một Sprint Backlog được tổ chức theo Sprint và bản phát hành.

Từ Phát triển Phần mềm Tăng trưởng đến Phát triển Phần mềm song song



Hình 7.11

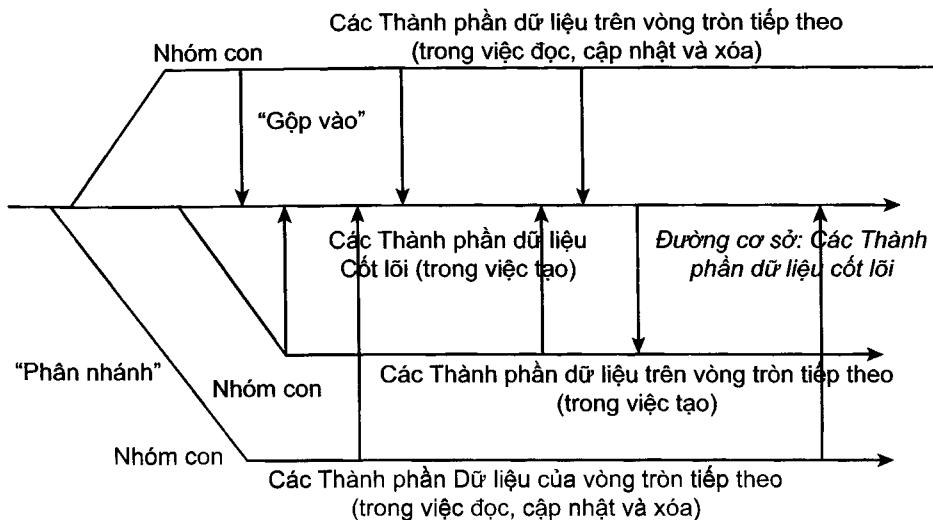
Mỗi quan hệ tỷ lệ nghịch giữa nợ kỹ thuật và chất lượng phần mềm.

TỪ PHÁT TRIỂN PHẦN MỀM TĂNG TRƯỞNG ĐẾN PHÁT TRIỂN PHẦN MỀM SONG SONG

Ngoài việc đem lại sự hỗ trợ cho nhóm Scrum về mặt lập kế hoạch phát hành và lập kế hoạch Sprint thì tầm nhìn kiến trúc còn mang lại lợi ích cho một khía cạnh khác đó là: phát triển phần mềm song song (thậm chí với một nhóm Scrum gồm 7 thành viên).

Nói cách khác, thông qua việc tổ chức công việc xung quanh các thành phần dữ liệu chung và khái niệm CRUD, bạn sẽ có khả năng tăng tốc độ phát triển bằng cách phân các nhóm con nhỏ hơn (thậm chí với một nhóm Scrum 7 thành viên) làm việc song song và trên các User story cụ thể, không phụ thuộc nhau như được thể hiện trong Hình 7.12. Việc này, đi kèm với một cơ chế tích hợp liên tục hiệu quả (sử dụng công cụ chặng hạn như Git) sẽ giúp nhóm Scrum tăng tốc độ một cách đáng kể.

Chương 7 - Từ Tầm nhìn kiến trúc đến Lập kế hoạch Phát hành, Lập kế hoạch Sprint và Phát triển Phần mềm song song



Hình 7.12

Các nhóm con tiến hành phát triển song song.

Bởi vì có nhiều hơn một nhóm đang làm việc cùng nhau, bạn phải thay đổi các câu hỏi trong buổi họp đứng hằng ngày (còn gọi là Họp Scrum Hằng ngày).

Ngoài ba câu hỏi nổi tiếng được đưa ra trong những cuộc họp là:

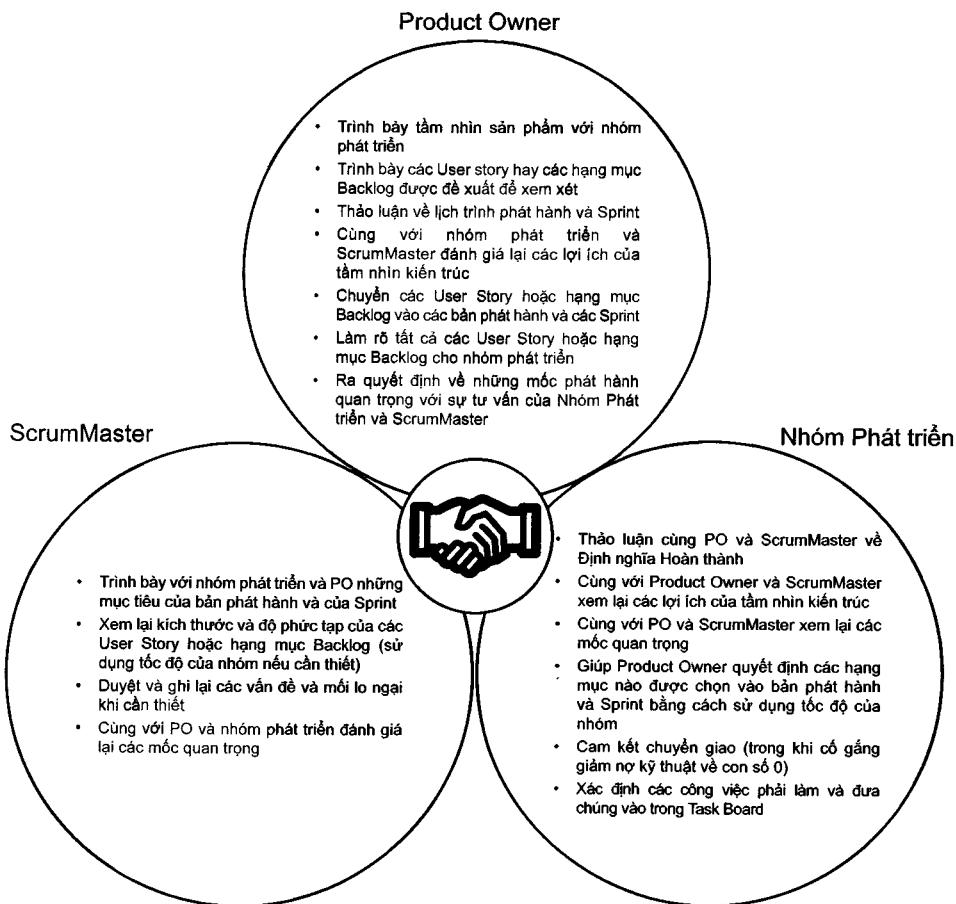
1. Bạn đã làm gì ngày hôm qua?
 2. Bạn dự định làm gì ngày mai?
 3. Có điều gì gây khó khăn cho bạn trong quá trình triển khai hay không?

Chúng tôi khuyên bạn nên thêm hai câu hỏi nữa, chúng sẽ giúp các nhóm con có thể đồng bộ hóa một cách dễ dàng:

1. Bạn đang làm gì có lợi cho nhóm con của tôi không?
 2. Bạn đang làm gì gây chậm tốc độ cho nhóm của tôi không?

Bản tổng kết trong Hình 7.13 là để nhắc nhở mọi người trong nhóm Scrum về trách nhiệm của họ đối với những công việc họ phải làm.

Từ Phát triển Phần mềm Tăng trưởng đến Phát triển Phần mềm song song



Hình 7.13

Cộng tác trong việc lập kế hoạch phát hành và lập kế hoạch Sprint.

TÓM TẮT

Việc xác định tầm nhìn kiến trúc từ sớm không chỉ giúp nhóm tránh khỏi vấn đề tốc độ nhóm bị dao động mà còn có thể làm tăng chất lượng phần mềm và tạo nền tảng cho một ứng dụng và kiến trúc dữ liệu tốt.

Ngoài ra việc xác định tầm nhìn kiến trúc sớm có thể giúp nhóm phát triển có một sự cộng tác tích cực, tốt hơn với Product Owner trong suốt buổi họp lập kế hoạch phát hành và lập kế hoạch Sprint.

Kết quả là, nhóm Scrum có thể giúp tăng giá trị kinh doanh cho các bên liên quan và sự bền vững của ứng dụng, trong khi vẫn có được sự hài lòng về làm việc nhóm và sự hiệu quả trong cộng tác.

Cuối cùng, bằng việc sử dụng hướng tiếp cận kiến trúc dữ liệu chung như đã được trình bày trong cuốn sách này, bạn có thể cho phép nhóm tăng tốc độ phát triển bằng cách chia nhóm Scrum thành các nhóm nhỏ hơn (còn được gọi là các nhóm chức năng) để thực hiện các công việc song song. Việc này đi kèm với một cơ chế tích hợp liên tục hiệu quả thì sẽ cho phép nhóm tăng tốc độ và khả năng kết thúc công việc sớm hơn dự định. Bất kể là có hay không có Scrum, thì những lợi ích này cũng là mơ ước của các quản trị viên dự án phần mềm và các nhà quản lý phần mềm trong nhiều năm qua.

CHƯƠNG 8

PRODUCT OWNER

Trong dự án Agile, mọi thành viên đều cần thiết, nhưng dự án Agile không thể thành công nếu thiếu một Product Owner tốt, người bảo vệ tầm nhìn và các mục tiêu về sản phẩm, bởi dự án Agile hoặc Scrum tập trung vào việc chuyển giao các kết quả và các giá trị kinh doanh.

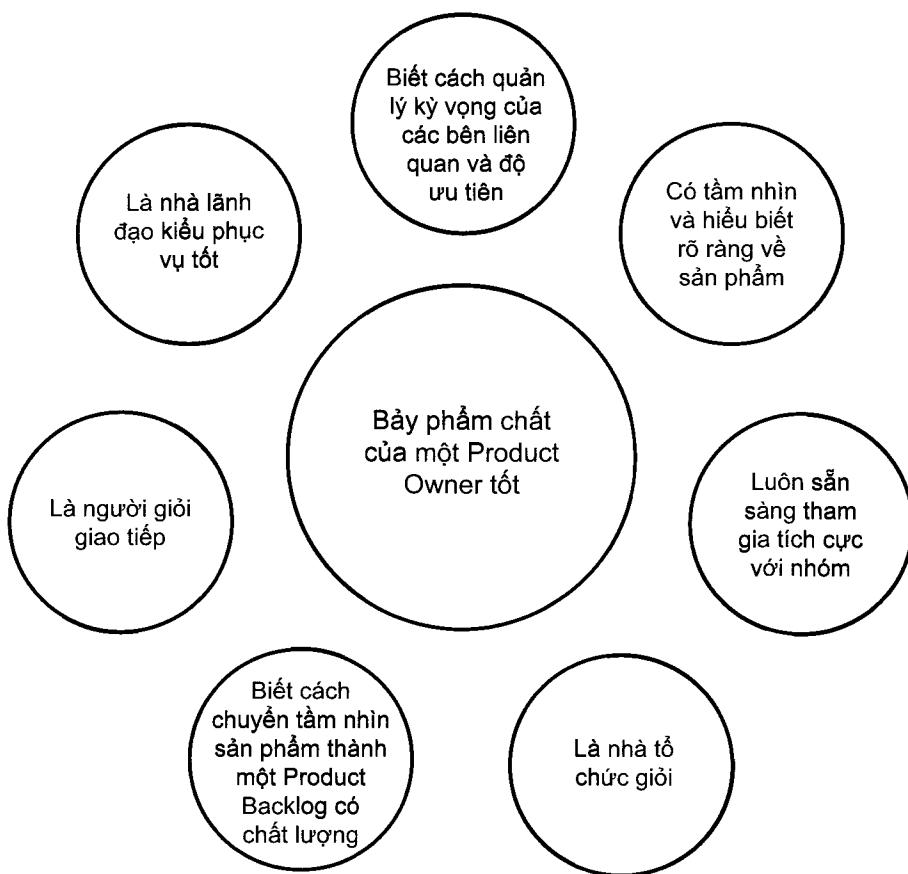
Hơn nữa, dự án Agile hoặc Scrum không chỉ cần một Product Owner tốt mà cần một Product Owner tuyệt vời.

Vậy câu hỏi đặt ra là làm sao để một người trở thành một Product Owner tốt nhất có thể?

Dựa vào kinh nghiệm của mình, chúng tôi tin rằng một Product Owner nên có bảy phẩm chất quan trọng. Những phẩm chất này được liệt kê trong Hình 8.1.

Bảy phẩm chất của một Product Owner tuyệt vời là:

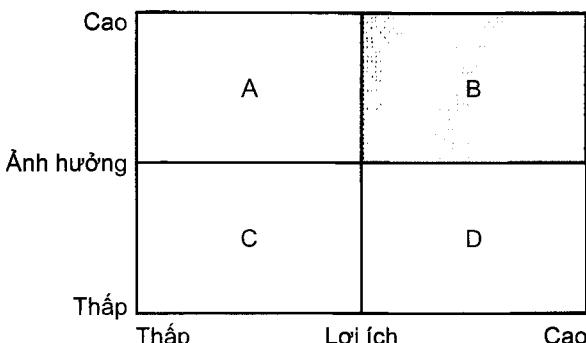
1. Biết cách để quản lý thành công kỳ vọng của các bên liên quan và đôi khi là sự mâu thuẫn về độ ưu tiên.
2. Có tầm nhìn và hiểu biết rõ ràng về sản phẩm.
3. Biết cách thu thập các yêu cầu để chuyển tầm nhìn sản phẩm thành một Product Backlog có chất lượng.
4. Luôn sẵn sàng tham gia với nhóm một cách tích cực, không chỉ trong Sprint, mà cả trong việc lập kế hoạch phát hành và lập kế hoạch Sprint.



Hình 8.1

Bảy phẩm chất của Product Owner.

5. Là nhà tổ chức giỏi để có thể sắp xếp được nhiều hoạt động trong khi vẫn giữ được quan điểm và sự bình tĩnh.
6. Biết cách truyền đạt tầm nhìn sản phẩm; không chỉ tới nhóm, mà cả với doanh nghiệp, do đó sự tin tưởng vào nhóm không bị lung lay trong vòng đời dự án.
7. Là một lãnh đạo tốt, có khả năng hướng dẫn, huấn luyện và hỗ trợ nhóm khi cần thiết trong khi vẫn đảm bảo thu được giá trị kinh doanh mong đợi từ CNTT.



Hình 8.2

Ma trận quản lý bên liên quan

Bây giờ hãy xem xét lần lượt từng phẩm chất này:

QUẢN LÝ KỲ VỌNG CỦA CÁC BÊN LIÊN QUAN VÀ ĐỘ ƯU TIÊN

Một trong những hoạt động quan trọng nhất của Product Owner là tương tác với các bên liên quan, quản lý kỳ vọng của họ và những mâu thuẫn thường xảy ra về độ ưu tiên.

Do không có đủ thời gian cần thiết để làm việc với các bên liên quan, nên chúng tôi gợi ý bạn nên học để hiểu và quản lý kỳ vọng của họ theo những cách khác nhau tùy thuộc vào mức độ ảnh hưởng của họ đối với sự thành công trong tương lai của dự án (Hình 8.2).

Điều này có nghĩa là hầu hết thời gian của bạn sẽ dành để làm việc với các bên liên quan thuộc loại B (trong Hình 8.2), đây là những người có nhiều ảnh hưởng và quyền lợi đối với dự án. Đồng thời, bạn cần cố gắng hướng các bên liên quan thuộc loại A đồng thuận với loại B và hướng loại C đồng thuận với loại D.

CÓ TẦM NHÌN VÀ HIỂU BIẾT RÕ RÀNG VỀ SẢN PHẨM

Tầm nhìn sản phẩm rõ ràng sẽ giúp Product Owner dễ dàng thiết lập các mục tiêu và độ ưu tiên, điều đó mang lại giá trị rất lớn cho nhóm trong việc lập kế hoạch phát hành và lập kế hoạch Sprint.

Mỗi người có cách tiếp cận tầm nhìn sản phẩm khác nhau, nhưng ở đây chúng tôi sử dụng một kỹ thuật đơn giản có tên là 5W:

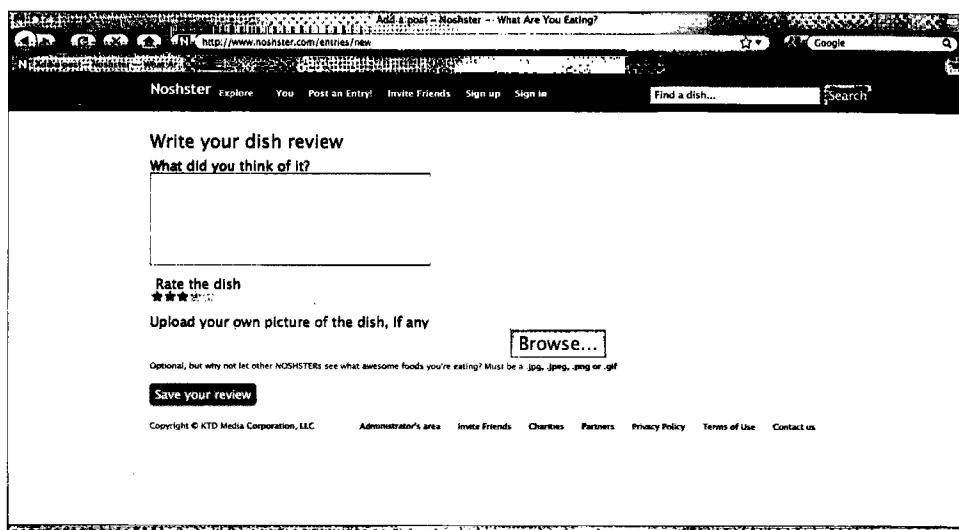
- Ai (Whom)? (Khách hàng hướng tới.)
- Tại sao (Why)? (Bằng cách nào mà sản phẩm này đặc biệt?)
- Cái gì (What)? (Sản phẩm này mang lại điều gì?)
- Ở đâu (Where)? (Địa điểm)

Chương 8 • Product Owner

- **Khi nào?** (Khoảng thời gian.)

Chúng ta hãy xem thử việc xây dựng một website với mục đích minh họa để trả lời những câu hỏi này (Xem Hình 8.3).

- **AI?** Tất cả những thực khách sành ăn?
- **Tại sao?** Tìm những món ngon nhất trên toàn thế giới.
- **Cái gì?** Xếp hạng tất cả các món ăn, thay vì các nhà hàng.
- **Ở đâu?** Tất cả các vùng trên thế giới.
- **Khi nào?** 24/7



Hình 8.3

Noshter: Một mạng xã hội về ẩm thực toàn cầu.

BIẾT CÁCH THU THẬP YÊU CẦU CHO PRODUCT BACKLOG

Mặc dù việc Product Owner có tầm nhìn sản phẩm là quan trọng, nhưng khả năng đưa ra một danh sách User story tốt cho Product backlog thậm chí còn quan trọng hơn.

Bởi thế nếu bạn thấy mình hoặc Product Owner (vẫn) chưa có đủ kỹ năng để thu thập yêu cầu khách hàng, thì đừng do dự trong việc quay lại Chương 4, “Thu thập các yêu cầu trực quan cho Product Backlog”, để xem lại cách thu thập các yêu cầu sao cho nhà phát triển có thể hiểu được dễ dàng và chuyển chúng thành các công việc.

TỰ LÀM MÌNH LUÔN SẴN SÀNG

Có thể bạn đã may mắn khi làm việc với một Product Owner có tầm nhìn sản phẩm rõ ràng, biết cách thu thập các yêu cầu và các hạng mục trong backlog.

Tuy nhiên, điều đó là không đủ nếu Product Owner không sẵn sàng làm việc và đối thoại với nhóm phát triển.

Product Owner nên là người luôn sẵn sàng, lý tưởng là hằng ngày, để tương tác với nhóm và tham dự cuộc họp sơ kết.

Nếu bạn ở một công ty mà Product Owner không làm được như vậy, thì bạn nên ưu tiên việc giải thích cho nhà quản lý rằng nhóm cần một người như thế và lý do nhóm cần một người vừa hiểu biết về nghiệp vụ vừa được trao quyền để tương tác thường xuyên với nhóm và ra các quyết định về nghiệp vụ.

BIẾT CÁCH TRỞ THÀNH MỘT NHÀ TỔ CHỨC GIỎI

Trừ khi bạn may mắn làm việc trong một công ty nhỏ, nơi mà Product Owner có nhiều thời gian, nếu không, có khả năng Product Owner là người luôn bận rộn xử lý mọi loại ưu tiên, từ làm việc với phòng marketing, tới xử lý với những vấn đề nghiệp vụ trong phòng ban của chính mình.

Trong trường hợp này, bạn nên là người khôn khéo, nhưng không nên do dự nhắc nhở Product Owner rằng việc tham gia tích cực của cô ta là tối cần thiết và là một phần của quy trình.

BIẾT CÁCH ĐỂ GIAO TIẾP TỐT HƠN NGƯỜI BÌNH THƯỜNG

Bên cạnh việc có một tầm nhìn sản phẩm tốt và khả năng viết ra các yêu cầu khả thi và dễ hiểu thì việc tốt nhất mà Product Owner có thể làm là tích cực ủng hộ nhóm với quản lý kinh doanh.

Để làm điều này, Product Owner sẽ cần làm việc tích cực với ban quản lý cũng như người dùng, luôn giúp họ hiểu trạng thái của nhóm liên quan tới mục tiêu phát hành và giá trị kinh doanh.

BIẾT TẤT CẢ VỀ LÃNH ĐẠO KIỂU PHỤC VỤ

Cuối cùng, Product Owner nên là người biết cách để trở thành một nhà lãnh đạo kiểu phục vụ (servant leader) tốt, người có thể hướng dẫn, hỗ trợ, cố vấn và khi cần thiết là huấn luyện nhóm để đạt được tầm nhìn và mục tiêu của dự án.

Bằng cách xem xét những phẩm chất trên, bạn có thể tự hỏi mình liệu Product Owner của nhóm mình đã có tất cả những phẩm chất cần thiết để làm việc hiệu quả chưa.

Nếu câu trả lời là có, thì bạn không phải làm gì hoặc lo lắng gì nữa.

Tuy nhiên, nếu bạn nhận thấy có một sự thiếu sót nào đó, đây có thể là lúc bạn phải làm việc với ScrumMaster để đào tạo, hoặc nhắc nhở Product Owner về vai trò của mình.

TÓM TẮT

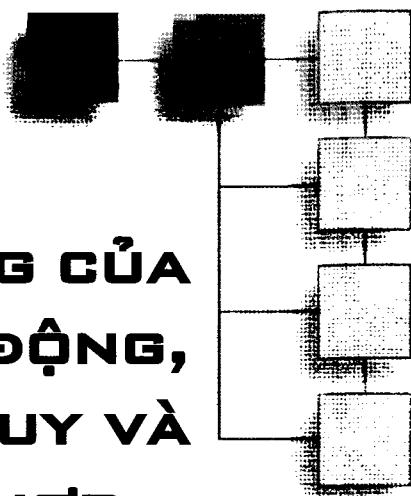
Mỗi người trong nhóm Scrum đều có một vai trò nào đó. Câu hỏi ai là người mang lại nhiều giá trị kinh doanh nhất để bảo đảm cho dự án. Câu trả lời nên là Product Owner.

Câu hỏi tiếp theo là một người có thể trở thành Product Owner tốt nhất có thể bằng cách nào?

Kinh nghiệm của chúng tôi trong những dự án thật cho thấy Product Owner nên có bảy phẩm chất như đã được trình bày trong chương.

CHƯƠNG 9

TẦM QUAN TRỌNG CỦA KIỂM THỬ TỰ ĐỘNG, KIỂM THỬ HỒI QUY VÀ KIỂM THỬ TÍCH HỢP



Trong mô hình thác nước (waterfall) truyền thống, kiểm thử luôn được tiến hành ở cuối vòng đời của quy trình phát triển, dù đó là kiểm thử hệ thống hay kiểm thử chấp nhận người dùng, như được minh họa ở Hình 9.1.

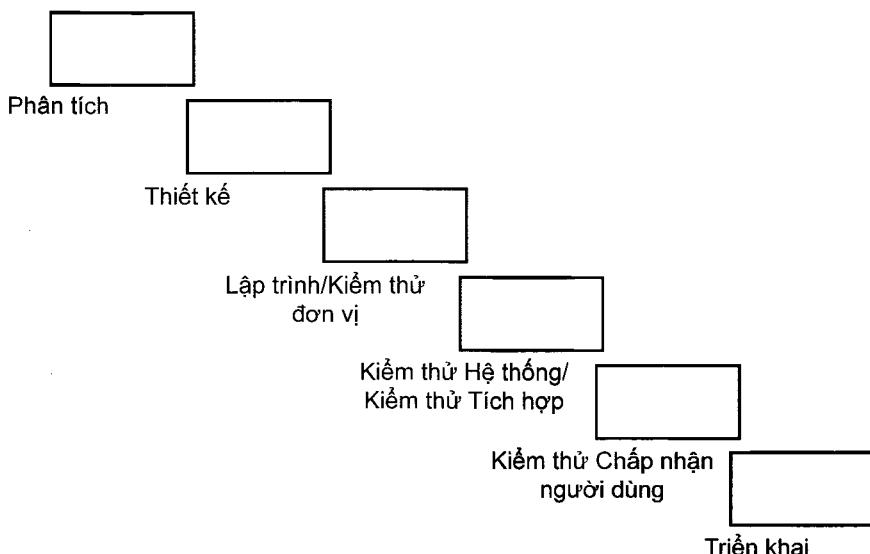
Không bàn luận thêm về tính hiệu quả của mô hình này, chúng ta chỉ nói về dự án Agile hoặc Scrum, ở đó kiểm thử không còn được thực thi ở cuối chu trình phát triển mà được “lồng” vào suốt những phân đoạn (Sprint) khác nhau, như thể hiện trong Hình 9.2.

Không nghi ngờ gì về việc kiểm thử là một trong những điểm khác biệt lớn nhất trong Scrum. Việc nhóm có hay không có khả năng tạo ra đều đặn những phần tăng trưởng sản phẩm có thể chuyển giao được phụ thuộc nhiều nhất vào cách tổ chức và thực thi kiểm thử.

Ngay cả với dự án Scrum nhỏ thì cũng khó tưởng tượng ra cách để nhóm có thể chuyển giao đều đặn nếu không có sẵn một vài cơ chế kiểm thử tự động đang hoạt động tốt.

Xét về lâu dài, càng có hạ tầng kiểm thử tự động tốt, thì tốc độ nhóm càng có xu hướng tăng lên (Hình 9.3).

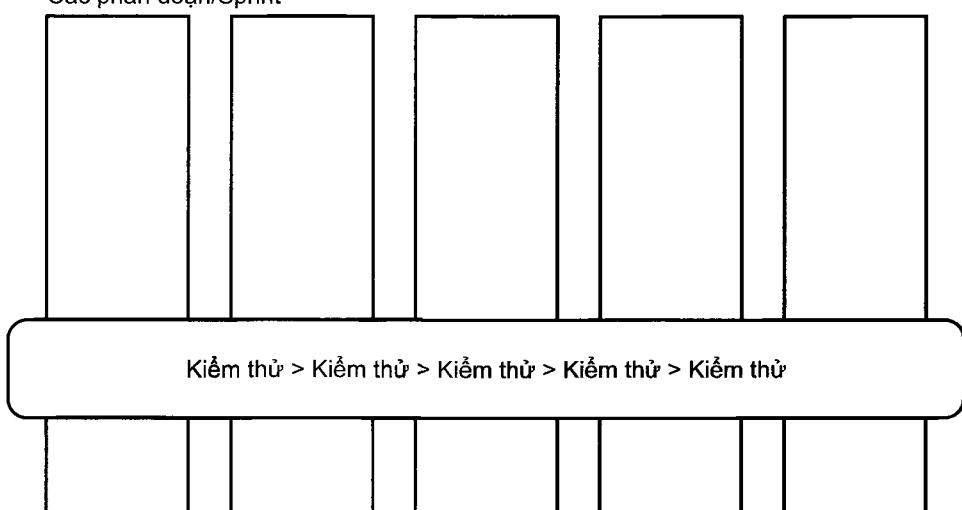
Chương 9 • Tầm quan trọng của Kiểm thử Tự động, Kiểm thử Hồi quy và Kiểm thử Tích hợp



Hình 9.1

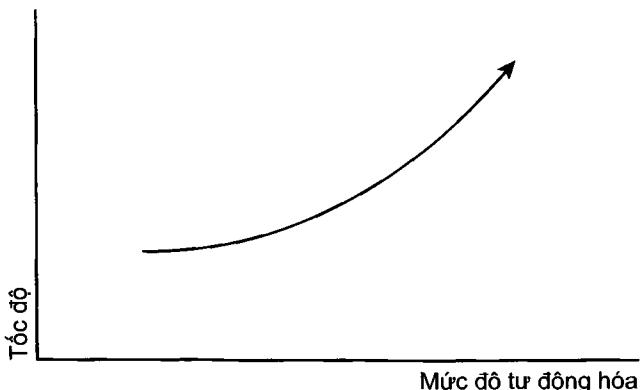
Kiểm thử trong quy trình làm phần mềm thác nước truyền thống.

Các phân đoạn/Sprint



Hình 9.2

Vị trí mới của kiểm thử trong khung quy trình Agile/Scrum.



Hình 9.3

Mức độ tự động hóa.

TẦM QUAN TRỌNG CỦA ĐỊNH NGHĨA HOÀN THÀNH

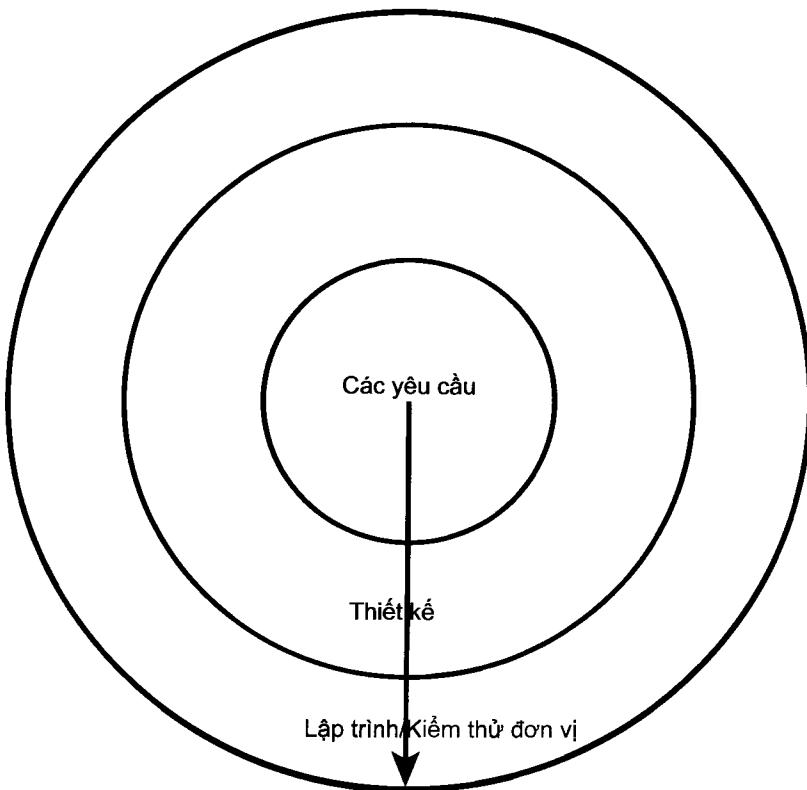
Trước khi tiếp tục thảo luận về kiểm thử, chúng ta hãy nói về định nghĩa của từ *hoàn thành*. Điều này giúp nhóm xác định sẽ làm những loại kiểm thử nào. Chúng tôi đã đề cập tới một số loại kiểm thử như kiểm thử chấp nhận người dùng hoặc kiểm thử kỹ thuật.

Cũng giống như hầu hết các chuyên gia, kể cả Henrik Kniberg, trong cuốn sách của ông, *Scrum and XP from the Trenches* (tạm dịch: Scrum và XP từ những chiến壕), chúng tôi cho rằng *hoàn thành* nghĩa là “sẵn sàng để triển khai cho sản xuất (thực tế).” Nhưng cũng giống như Kniberg, đôi khi chúng tôi cũng phải chấp nhận rằng định nghĩa hoàn thành có thể khác đi.

Định nghĩa hoàn thành thường khác nhau, tùy thuộc vào tình hình dự án. Trong mục tiếp theo, chúng ta sẽ xem xét hai hoặc ba định nghĩa hoàn thành mà chúng tôi đã thấy trong những dự án thực tế.

Định nghĩa hoàn thành đầu tiên trong Hình 9.4 cho thấy nhóm đã quyết định rằng công việc được coi là hoàn thành khi hoàn tất việc lập trình và kiểm thử đơn vị.

Ví dụ trong Hình 9.4 minh họa nhiều nhóm đang làm việc song song trên cùng một dòng sản phẩm. Những nhóm này quyết định rằng việc chuyển giao các kết quả công việc khác nhau sau khi đã kết thúc kiểm thử đơn vị là có lợi nhất đối với họ, do đó nhiệm vụ tích hợp và ổn định có thể được tiến hành trước khi chuyển sang phân đoạn tiếp theo.



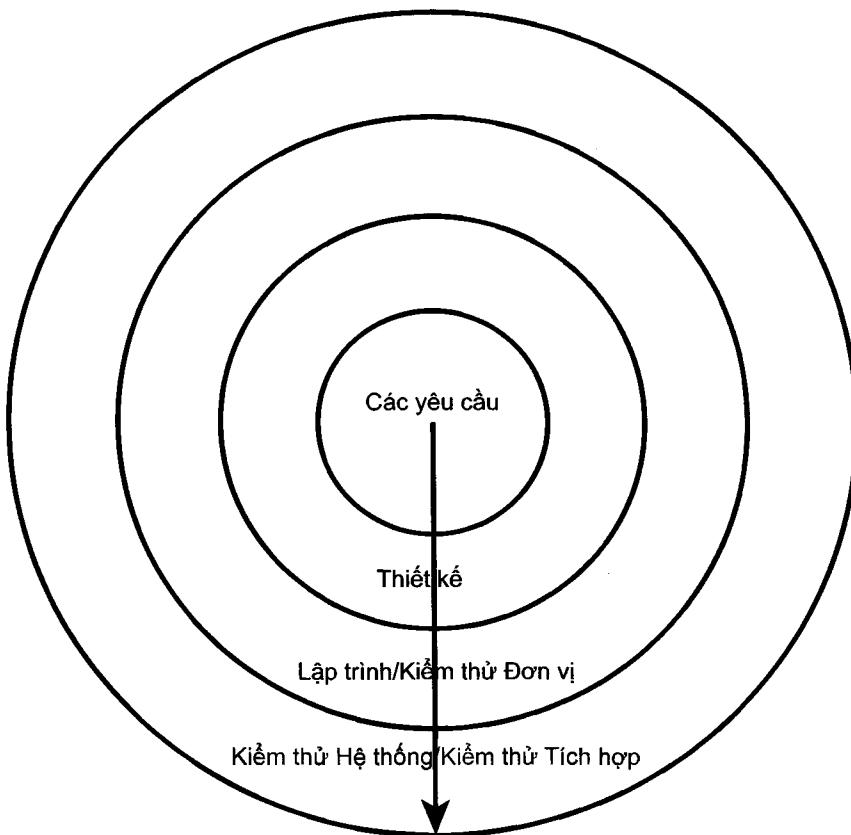
Hình 9.4

Một định nghĩa hoàn thành.

Đây là một tình huống mà bạn có thể gặp trong đời thực – công việc được coi là hoàn thành khi đạt mức kiểm thử đơn vị. Cách tiếp cận này khả quan khi nhóm đã tính tới nỗ lực của việc tích hợp và kiểm thử là cần thiết trước khi triển khai cài đặt.

Hình 9.5 cho thấy một kịch bản khác ở đó nhóm cho rằng công việc chỉ hoàn thành khi mọi User story mới đã được tích hợp và kiểm thử trước buổi sơ kết Sprint.

Hình 9.6 đưa ra kịch bản tốt nhất, trong đó người dùng nghiệp vụ là một phần của nhóm Scrum, và họ có trách nhiệm thực hiện các kiểm thử chấp nhận trước khi công việc được cho là hoàn thành. Dù dự án có sử dụng Scrum hay không, việc phát triển với kiểm thử chấp nhận người dùng ngày càng trở thành một phương pháp thực tiễn phổ biến và nên được đưa vào các phân đoạn phát hành càng sớm càng tốt.



Hình 9.5

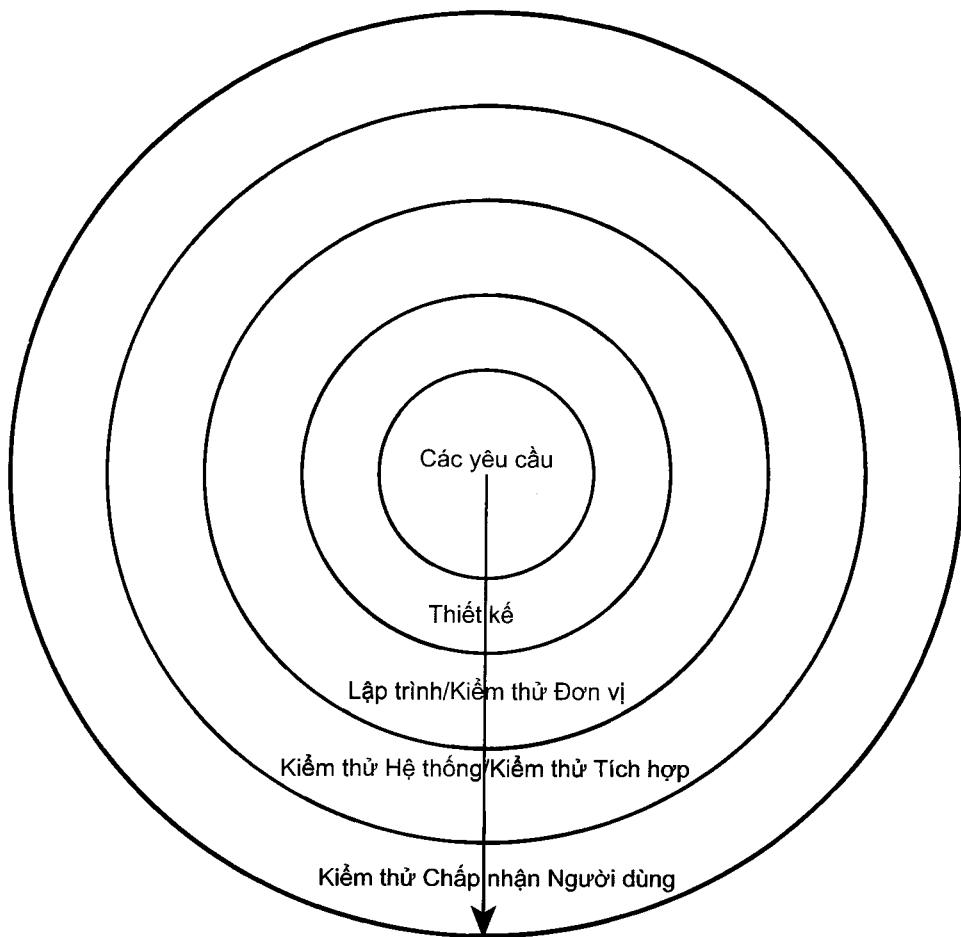
Một định nghĩa hoàn thành khác.

NHỮNG KIỂM THỬ QUAN TRỌNG NHẤT

Do Scrum chủ yếu là một khung quản lý dự án, nên những phương pháp kỹ thuật liên quan tới lập trình và kiểm thử hầu như ít được nhắc tới.

Tuy nhiên, theo kinh nghiệm của chúng tôi, thì tổ chức của bạn nên thực hiện những kiểm thử sau để duy trì những nỗ lực về Scrum:

1. Kiểm thử Tự động
2. Kiểm thử Tích hợp liên tục



Hình 9.6

Một định nghĩa hoàn thành khác.

Kiểm thử Tự động

Lý do mà chúng tôi liệt kê kiểm thử tự động trước là vì ngay khi bạn bắt đầu thấy nhóm định kỳ tung ra các phần tăng trưởng phần mềm, bạn sẽ nhận ra rằng nhóm không thể duy trì được nhịp độ trừ khi có một số thứ được tự động hóa.

Kiểm thử tự động sẽ giúp bạn tiết kiệm thời gian và cho phép nhóm lướt đi một cách êm ái.

Không giống như trong kiểm thử thủ công bạn phải tự mình chạy tất cả mọi test case, kiểm thử tự động có thể tự thực hiện hằng giờ, thậm chí hằng ngày, tùy thuộc vào mã nguồn. Một trong những lợi ích chính của kiểm thử tự động là có một phần mềm sẽ xử lý tất cả kiểm thử cho bạn.

Dĩ nhiên, để thực hiện kiểm thử tự động, bạn phải làm thêm một số việc và phải mua công cụ đồng thời tạo sẵn mọi test case. Nhưng điều này chỉ chiếm một phần nhỏ so với khoảng thời gian mà bạn phải làm khi thực hiện mọi thứ một cách thủ công.

Kiểm thử Tích hợp liên tục

Một loại kiểm thử khác mà chúng tôi nghĩ rằng sẽ cấp thiết với sự thành công của dự án Scrum đó là kiểm thử tích hợp liên tục.

Lý do để thực hiện đều đặn loại kiểm thử quan trọng này là vì bạn muốn đảm bảo rằng sản phẩm của mình luôn chuyển giao được.

TỔ CHỨC CƠ SỞ HẠ TẦNG KIỂM THỬ

Tới đây chúng ta đã tìm hiểu về những loại kiểm thử được xem là căn bản nhất, tiếp theo, chúng ta hãy chuyển sự chú ý sang việc tổ chức kiểm thử.

Nếu bạn may mắn và công ty của bạn hoàn toàn cam kết với Scrum, bạn không cần thực hiện thêm gì nữa ngoài làm việc và giao tiếp chặt chẽ với phòng đảm bảo chất lượng. Ngược lại, bạn có thể nhận thấy rằng công ty chưa triển khai Scrum cho toàn bộ doanh nghiệp, như thế có nghĩa là không có hoặc có rất ít cơ sở hạ tầng về kiểm thử tự động hoặc kiểm thử tích hợp liên tục dành cho Scrum.

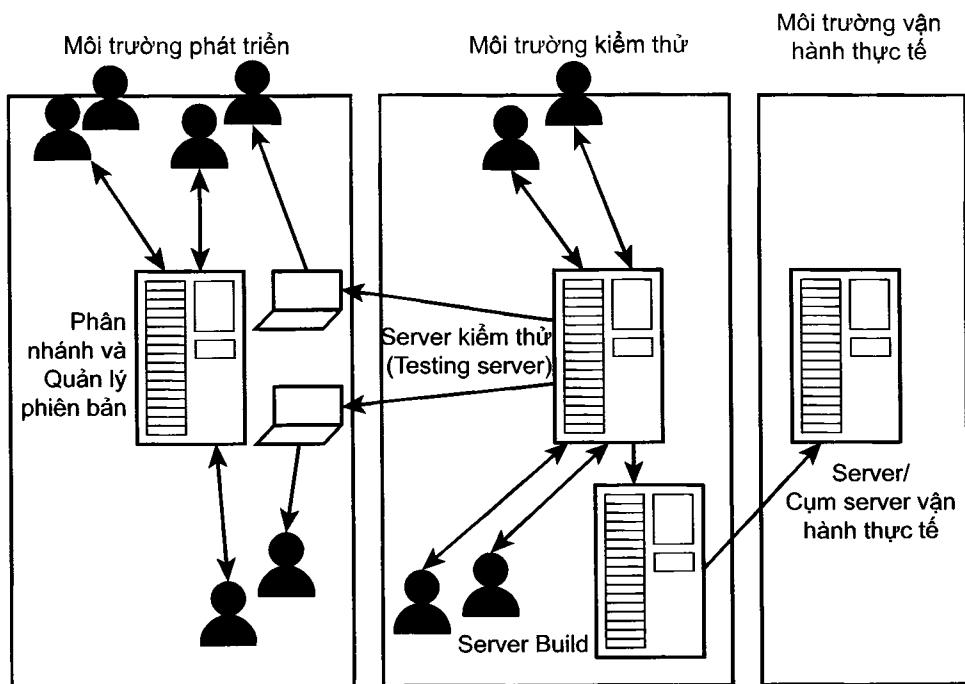
Nếu điều này xảy ra, chương này sẽ cung cấp cho bạn một vài ý tưởng về cơ sở hạ tầng kiểm thử, thứ mà công ty sẽ hỏi bạn cho chính nhu cầu của bạn hoặc công ty.

Hình 9.7 cho chúng ta thấy một môi trường tốt, trong đó phần mềm được tổ chức quanh ba môi trường khác nhau: phát triển, kiểm thử, và vận hành thực tế.

Có một dòng chảy tự nhiên từ phát triển tới kiểm thử và từ kiểm thử tới vận hành thực tế.

Bạn có thể không đủ may mắn để được làm việc ở một công ty mà mọi thứ đã được cài đặt sẵn như thế. Chúng tôi gợi ý rằng bạn nên gấp gô các thành viên của nhóm kỹ thuật ngay từ đầu dự án, và xem xét giải pháp mà họ có thể cài đặt môi trường kiểm thử tạm thời trong môi trường phát triển dự án của bạn, như ở Hình 9.8.

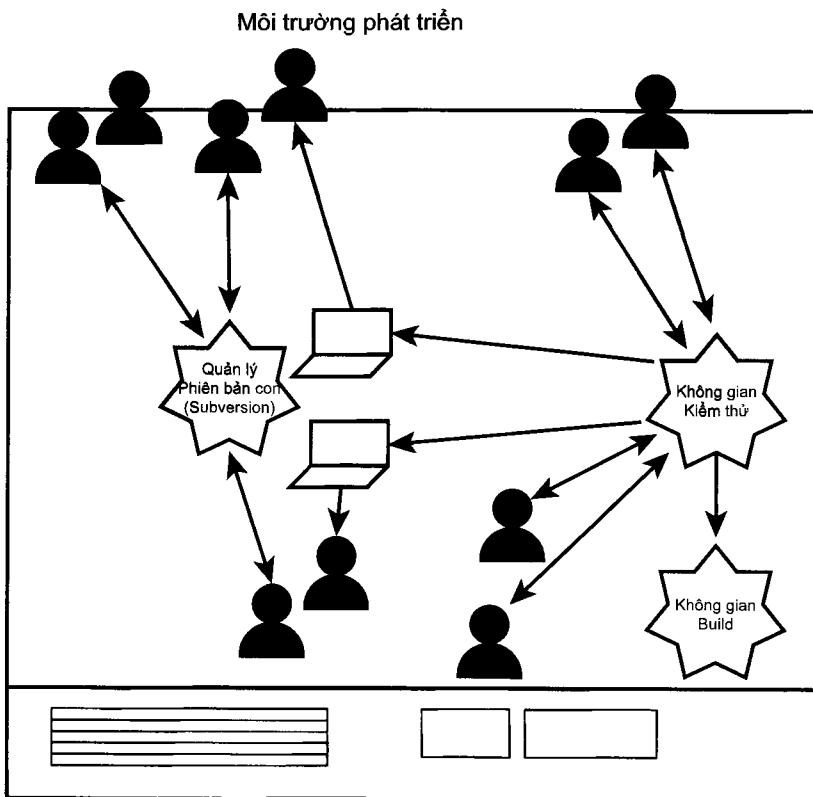
Chương 9 • Tầm quan trọng của Kiểm thử Tự động, Kiểm thử Hồi quy và Kiểm thử Tích hợp



Hình 9.7

Từ phát triển tới kiểm thử và tới môi trường vận hành thực tế.

Từ kinh nghiệm của mình, chúng tôi biết rằng nhiều nhóm không bắt đầu với điều này, nhưng đó là một phần thiết yếu của tất cả những dự án Agile hoặc Scrum thành công mà chúng tôi đã tham gia. Đó là lý do tại sao chúng tôi đưa vào chương này để giúp bạn tiết kiệm thời gian và bớt đau đầu.

**Hình 9.8**

Tổ chức kiểm thử trong môi trường phát triển.

Trừ khi bạn dành thời gian để tổ chức chiến lược kiểm thử của bản thân và tổ chức, nếu không bạn sẽ không thể chuyển giao với Scrum nếu việc kiểm thử không tuân theo hoặc không hỗ trợ bạn.

TÓM TẮT

Kiểm thử không còn nằm ở cuối của vòng đời phát triển phần mềm nữa. Trong chương này chúng tôi chỉ nhấn mạnh một số kiểm thử then chốt nhằm đem lại thành công cho dự án Scrum. Đó là lý do tại sao bạn không thấy chúng tôi đề cập tới việc phát triển hướng kiểm thử (Test-driven develop - TDD), dù phương pháp này mang lại hiệu quả cao để nhóm lập trình trung bình có thể bàn giao được mã nguồn chất lượng. Thay vào đó, chương này tập trung nhiều vào kiểm thử tự động và kiểm thử tích hợp liên tục, hai loại kiểm thử quan trọng nhất mà mỗi nhóm Scrum cần phải thiết lập sớm nhất có thể.

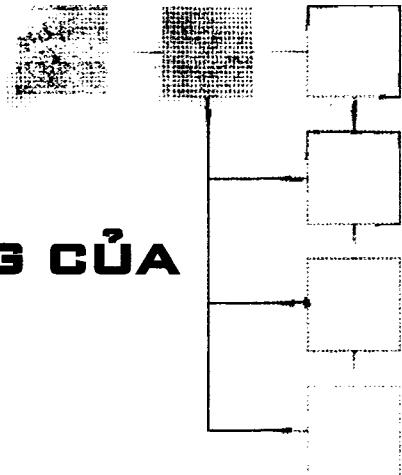
Sau đó, chương này tập trung vào việc tổ chức kiểm thử.

Nếu bạn không làm việc trong một công ty có hạ tầng kiểm thử tốt, thì chúng tôi khuyến nghị bạn hãy làm việc với các thành viên nhóm kỹ thuật ngay từ đầu dự án để thiết lập môi trường kiểm thử tạm thời – nếu điều này cần thiết cho môi trường phát triển dự án. Môi trường kiểm thử này sẽ cung cấp cho nhóm phát triển sự hỗ trợ thiết yếu khi họ tạo ra các phần tăng trưởng mới nối tiếp nhau, hết Sprint này tới Sprint khác.

Bởi định nghĩa *hoàn thành* ảnh hưởng rất lớn tới số lượng các loại kiểm thử cần thực hiện, nên chương này cũng đưa ra một số định nghĩa hoàn thành khác nhau để cung cấp cho bạn một số ý tưởng về điều bạn cần làm trong dự án của mình.

CHƯƠNG 10

TẦM QUAN TRỌNG CỦA LÀM VIỆC NHÓM



Chúng tôi chắc rằng, giống như nhiều người khác bạn đã nghe nhiều về làm việc nhóm (teamwork) và có thể thắc mắc rằng bạn sẽ được chia sẻ những thông tin mới gì trong chương này.

Để tiết kiệm thời gian, chúng tôi sẽ không nhắc lại những điều mà bạn đã nghe hàng trăm lần trước đây. Như đã nói, làm việc nhóm vẫn là yếu tố rất quan trọng trong dự án Scrum.

Cho dù bạn có sự hỗ trợ từ ban quản lý và cùng nhau tạo được một bản Product Backlog tốt nhất có thể, bạn có thể vẫn thất bại nếu các thành viên trong nhóm không cộng tác tốt trong công việc. Từ kinh nghiệm của bản thân, chúng tôi chưa từng thấy bất cứ một dự án nào thành công và nhóm vẫn chuyển giao được trong khi họ không cộng tác tốt với nhau.

Trong một dự án ngắn thì làm việc nhóm có thể không quan trọng do thời gian bị hạn chế. Nhưng khi mọi người cần phải cộng tác với nhau hằng ngày trong một dự án lớn hoặc trung bình, sẽ cần nhiều yếu tố thay vì may mắn để nhóm cộng tác tốt.

Nếu các thành viên trong một nhóm dự án không hòa hợp và không thể làm gì để đạt được điều đó, chúng ta sẽ thấy hầu như dự án đó sẽ kết thúc với các rắc rối. Điều này đúng với những dự án truyền thống và thậm chí với nhiều dự án Agile/Scrum cũng vậy vì hai khung quy trình này phụ thuộc nhiều vào sự tự quản và tinh thần trách nhiệm của nhóm.

Một trong những quy tắc tham gia Scrum đó là đòi hỏi mọi người phải dựa vào nhau như một nhóm để thực hiện công việc. Trong khi chủ đề này nghe có vẻ sáo rỗng, tuy nhiên không có gì quan trọng cho một dự án Scrum hơn là việc các thành viên cộng tác tốt với nhau.

Chương 10 • Tầm quan trọng của làm việc nhóm

Trong những nội dung tiếp theo, chúng tôi sẽ cung cấp cho bạn một số ý tưởng và kỹ thuật mà chúng tôi đã học được trong những năm qua. Điều này sẽ giúp bạn cũng như nhóm của mình tránh được những xung đột, hoặc ít nhất là học được cách để giải quyết xung đột càng nhiều càng tốt, trước khi chúng phá hỏng dự án của bạn.

Tuy nhiên, trước khi nói về nhóm, hãy nói về những cá nhân trong nhóm bởi vì tất cả các nhóm đều là tập hợp của những cá nhân.

CÁ NHÂN

Máy tính là những cỗ máy phức tạp, nhưng không có gì phức tạp hơn con người cả.

May mắn thay, tìm hiểu thêm về con người luôn là một đề tài nghiên cứu, qua đó cung cấp cho chúng ta những hiểu biết sâu sắc về bản thân mỗi con người chúng ta.

Chúng tôi sẽ không biến cuốn sách này thành luận thuyết về tâm lý con người, nhưng sẽ thảo luận về những yếu tố thuộc về bản chất con người có thể giúp nhóm Scrum làm việc tốt cùng nhau.

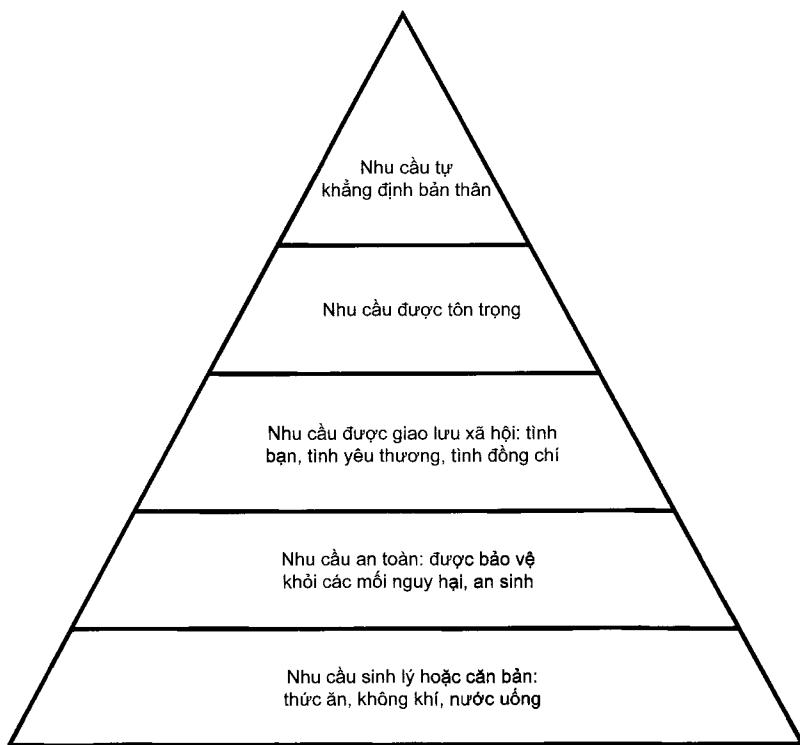
Trước hết, hãy đề cập tới những nghiên cứu được thực hiện bởi Abraham Maslow, được biết đến với tên gọi Tháp Maslow (Maslow's Pyramid) hoặc tháp nhu cầu, nó đưa ra một số khám phá về nhu cầu của bản thân mỗi con người.

Quan sát tháp nhu cầu trong Hình 10.1 từ dưới lên, chúng ta sẽ thấy rằng mỗi cá nhân trước hết phải thỏa mãn một số nhu cầu về thể xác, như thức ăn và nước uống.

Một khi những nhu cầu này được đáp ứng đủ, nhu cầu tiếp theo là cảm giác an toàn. Sự an toàn có thể có được từ một mái nhà để trú ngụ hoặc một nơi để bảo vệ bản thân họ khỏi thời tiết hoặc những nguy hiểm đến từ bên ngoài.

Sau đó, mỗi cá nhân cần có các mối liên hệ xã hội.

Ở những tầng cuối cùng của tháp, là các nhu cầu về được tôn trọng và tự khẳng định bản thân, đây là những gì mà những bậc thầy về Agile và chuyên gia Scrum quan tâm đối với hầu hết các nhóm để trao cho họ quyền thực thi công việc và tự quản. Đây chính xác là lý do tại sao rõ ràng trong Scrum chúng ta lại cho phép sự tự quản và trao quyền để nhóm tự chọn công việc và làm bất cứ thứ gì họ thích để thực hiện cam kết của mình.

**Hình 10.1**

Tháp nhu cầu của Maslow.

Chúng ta đã nói khá nhiều về cá nhân, nhưng trước khi đi vào tìm hiểu chi tiết về nhóm (team) thì hãy nói về tập hợp (group).

TẬP HỢP (GROUP)

Bất cứ lúc nào một cá nhân cảm thấy cần một cảm giác thân thuộc hoặc chia sẻ một nhu cầu hoặc ý tưởng chung thì đều có xu hướng tìm đến những người mà mình có thể kết giao.

Tuy vậy, một tập hợp lại không có mục tiêu chung để gắn kết các cá nhân, không có mục tiêu để họ phải chịu trách nhiệm.

Trở lại với những gì mà chúng ta đã hiểu về làm việc nhóm và năng suất nhóm, bất cứ khi nào một dự án bị thất bại mà không phải vì một lý do nào khác, thì rất có thể là do nguyên nhân nhóm dự án đó đã làm việc như một tập hợp và không cảm thấy có trách nhiệm chung với một thứ gì đó.

NHÓM (TEAM)

Khi một số cá nhân đến với nhau để cùng hướng tới một mục tiêu chung, thì đó chính là điểm khởi đầu của một nhóm. Việc trở thành một nhóm hiệu suất cao hay hiệu suất thấp phụ thuộc vào khả năng làm việc cùng nhau như là một nhóm tự tổ chức hay phụ thuộc vào người lãnh đạo họ trong môi trường mệnh lệnh và kiểm soát truyền thống.

Khi chúng ta nói về một nhóm, hoặc một tập hợp gồm những cá nhân cùng thực hiện một mục tiêu chung, thì không đồng nghĩa với việc ai sẽ làm gì, trong bao lâu, làm bao nhiêu, hay lý do bắt đầu.

Trong diễn giải của Patrick Lencioni, trong cuốn *Five Dysfunctions of Teams* (tạm dịch: Năm điều bất ổn của nhóm), chính làm việc nhóm chứ không phải là vấn đề tài chính hay công nghệ đem lại lợi thế cạnh tranh.

Do vậy, việc tất cả các thành viên của một nhóm cộng tác tốt với nhau là cực kỳ quan trọng.

Trong số những điều mà bạn cần biết về đồng đội của mình thì không có gì quan trọng hơn là hiểu về cá tính của họ. Hiểu được điều này sẽ giúp bạn giao tiếp hiệu quả với đồng đội khi họ hiểu và chấp nhận những gì bạn nói một cách dễ dàng hơn. Hơn nữa, điều này cũng giúp bạn biết lắng nghe và dễ dàng chấp nhận những thứ khác từ họ, đồng thời loại bỏ những hiểu lầm và các xung đột tiềm tàng.

CÁC LOẠI TÍNH CÁCH KEIRSEY

Lần theo ý tưởng về tính cách của thời Hy Lạp cổ đại, David Keirsey đã xây dựng nên thuyết tính cách hiện đại, bao gồm 16 loại tính cách, được gọi là Phân loại Tính cách Keirsey (Hình 10.2).

Phân loại Tính cách Keirsey được thiết kế để giúp mọi người hiểu hơn về bản thân mình, bao gồm tất cả 16 loại tính cách:

1. **Inspector** (Thanh tra) là người cẩn thận và tỉ mỉ. Đây là những cá nhân đặc biệt đáng tin cậy trong việc theo đuổi những thứ mà họ đã hứa.
2. **Protector** (Người bảo vệ) là người rất ám áp và tốt bụng. Họ coi trọng sự hợp tác và rất nhạy cảm với cảm giác của đồng nghiệp.
3. **Counselor** (Tư vấn) là những cá nhân rất nhẹ nhàng, chu đáo và thiên về trực giác. Họ là những người theo thuyết hoàn hảo, cứng đầu và có xu hướng bỏ qua ý kiến của những người khác, họ luôn nghĩ là mình đúng.
4. **Mastermind** (Quân sư hay người vạch ra kế hoạch) là người sống nội tâm, thực tế và chú tâm. Họ quan sát thế giới để tìm kiếm ý tưởng và cơ hội. Đầu óc của họ luôn thu thập thông tin.

IS_T_EJ	IS_F_EJ	IN_F_EJ	IN_T_EJ
Inspector	Protector	Counselor	Mastermind
$IS_ET_I P$	$IS_EF_I P$	$IN_EF_I P$	$IN_ET_I P$
Crafter	Composer	Healer	Architect
$ES_ET_I P$	$ES_EF_I P$	$EN_EF_I P$	$EN_ET_I P$
Promoter	Performer	Champion	Inventor
$ES_IT_E J$	$ES_F_E J$	$ES_F_I J$	$EN_T_E J$
Supervisor	Provider	Teacher	Fieldmarshal

Hình 10.2

16 loại tính cách Keirsey

5. **Crafter** (Thợ thủ công) là người bị lôi cuốn, luôn muốn hiểu cách thức hoạt động của mọi thứ. Họ là người logic và thiên hướng hành động. Họ thường không biết sợ và rất độc lập.
6. **Composer** (Nhạc sĩ) là người trầm tĩnh và kín đáo, rất khó để hiểu họ. Họ giữ các ý tưởng và ý kiến cho riêng mình, ngoại trừ đối với những người rất thân.
7. **Healer** (Người hòa giải) là người sống nội tâm và hợp tác. Họ không thích sự xung đột và sẽ đi đến cùng để loại bỏ chúng. Nếu phải đối mặt thì họ sẽ luôn tiếp cận từ quan điểm cảm xúc của mình.
8. **Architect** (Kiến trúc sư) là người sống nội tâm nhưng thực tế. Quan tâm lớn nhất của họ là xác định xem mọi thứ được cấu trúc, xây dựng và cài đặt như thế nào. Họ sống chủ yếu với những suy nghĩ nội tâm và tận hưởng việc phân tích các vấn đề khó để đi đến những giải pháp hợp lý. Không có gì ngạc nhiên, họ rất khoan dung và linh hoạt.
9. **Promoter** (Người kích động) là kẻ sống với hành động. Cùn, mạo hiểm, họ sẵn sàng nhảy vào mọi thứ làm vầy bắn tay mình.
10. **Performer** (Diễn viên) họ thích trở thành trung tâm của sự chú ý. Trong những tình huống xã hội, họ hoạt bát và biểu cảm. Sống động và vui nhộn, các diễn viên thường muốn thu hút sự chú ý của người khác. Họ sống cho hiện tại và luôn giữ mình cập nhật với các xu hướng mới nhất.
11. **Champion** (Nhà vô địch) là người rất nhiệt tình, thường rất tỏa sáng và đầy tiềm năng. Các nhà vô địch luôn muốn cả thế giới biết đến suy nghĩ của mình, niềm tin và sự nhiệt tình của họ thường truyền cảm hứng và thúc đẩy những người khác.
12. **Inventor** (Nhà sáng chế) thường quan tâm đến việc tạo ra các ý tưởng thay vì phát triển kế hoạch hoặc đưa ra các quyết định. Rất kỳ lạ là họ có phẩm chất về kinh doanh và thường tìm kiếm các dự án mới.

Chương 10 • Tầm quan trọng của làm việc nhóm

13. **Supervisor** (Giám sát viên) là người sống trong hiện tại với đôi mắt của họ luôn quan sát môi trường xung quanh để đảm bảo rằng mọi thứ đang hoạt động trơn tru và có hệ thống.
14. **Provider** (Nhà cung cấp) quan tâm một cách tự nhiên đến những người khác. Họ thích mọi người và có một kỹ năng đặc biệt đó là làm bộc lộ những điểm tốt nhất của người khác.
15. **Teacher** (Giáo viên) là người có những kỹ năng con người đặc biệt. Họ hiểu và quan tâm đến mọi người và có một tài năng đặc biệt trong việc làm bộc lộ những điểm tốt nhất trong mỗi con người.
16. **Field Marshal** (Thống soái) là nhà lãnh đạo bẩm sinh. Họ sống trong thế giới đầy thách thức và muốn là người chịu trách nhiệm chinh phục thế giới. Tài năng về lập kế hoạch dự phòng của họ chỉ đứng sau khả năng thực thi chiến lược hoặc lên kế hoạch hành động.

Biết được loại tính cách của các thành viên trong nhóm sẽ giúp ích rất nhiều trong việc hiểu người khác và làm việc với nhau tốt hơn, nhưng đôi khi điều này vẫn chưa đủ để loại bỏ các vấn đề và xung đột.

Phụ thuộc vào từng giai đoạn khi mà vấn đề hoặc xung đột xảy ra, có nhiều kỹ thuật dùng để giải quyết xung đột, chúng ta sẽ xem xét một số kỹ thuật ở phần tiếp theo.

Nhưng trước hết, chúng ta sẽ xem lại các giai đoạn khác nhau của nhóm và những kỹ thuật giải quyết xung đột, từ đó quyết định kỹ thuật nào thích hợp nhất để sử dụng với từng giai đoạn của nhóm.

NĂM GIAI ĐOẠN CỦA NHÓM

Năm 1965, Bruce Tuckman đã xác định năm giai đoạn mà một nhóm sẽ trải qua cùng nhau:

1. **Hình thành** (Forming): Đây là lúc mà nhóm được tập hợp lần đầu tiên. Vào thời điểm này, mọi người thường hành xử theo cách xã giao và kín đáo.
2. **Sóng gió** (Storming): Trong giai đoạn này, các thành viên của nhóm bắt đầu đặt mình vào vị trí đối nghịch với người khác, thường có vẻ khá đối đầu. Đây là lúc mà vai trò của người quản lý hoặc lãnh đạo là rất hữu ích trong việc xây dựng lòng tin giữa các thành viên trong nhóm. Chúng tôi sẽ bàn chi tiết thêm về điều này ở chương sau.
3. **Quy chuẩn** (Norming): Đây là lúc các thành viên nhóm đổi đầu với nhau khi giải quyết các vấn đề của dự án.
4. **Hoạt động thành công** (Performing): Đây là thời gian mà các thành viên nhóm bắt đầu làm việc cùng nhau hiệu quả và năng suất. Sự tin tưởng giữa các thành viên ở mức khá cao.
5. **Thoái trào** (Adjourning): Giai đoạn cuối cùng trước lúc nhóm được giải phóng khi đã hoàn tất thời kỳ làm việc nhóm.

CÁC KỸ THUẬT ĐỂ GIẢI QUYẾT XUNG ĐỘT TRONG NHÓM

Mặc dù không một ai muốn có xung đột xảy ra, nhưng không may tất cả chúng ta đều đã hoặc sẽ gặp phải trong suốt thời gian làm việc của mình.

Có nhiều kỹ thuật để giải quyết xung đột, nhưng một trong những kỹ thuật nổi tiếng nhất được Kenneth Thomas và Ralph Kilmann đưa ra vào những năm 70 của thế kỷ trước:

1. **Điều tiết** (Accommodating - ACCO): Kỹ thuật này đòi hỏi sự sẵn sàng đáp ứng nhu cầu của những người khác làm ảnh hưởng tới những nhu cầu của chính mình.
2. **Thỏa hiệp** (Compromise - COMP): Điều này xảy ra khi tất cả mọi người trong vụ xung đột từ bỏ một cái gì đó để đạt được một thỏa thuận.
3. **Cạnh tranh** (Competitive - COMPE): Đây là kỹ thuật hữu ích khi có trường hợp khẩn cấp và quyết định cần được đưa ra nhanh chóng hoặc khi quyết định này không phổ biến.
4. **Cộng tác** (Collaborating - COLLA): Tất cả các quan điểm của những thành viên khác nhau trong nhóm đều được xem xét. Kỹ thuật này thường dẫn tới một sự đồng thuận tốt.
5. **Né tránh** (Avoidance - AVOID): Một trong các bên từ chối thảo luận về xung đột. Đây là một ví dụ về kỹ thuật giải quyết xung đột kiểu thua-thua (lose-lose).

Bây giờ chúng ta hãy kết hợp các giai đoạn của nhóm, vòng đời dự án và các kỹ thuật giải quyết xung đột trong một ma trận giải quyết xung đột (Hình 10.3) nhằm giúp hướng dẫn các thành viên khi họ gặp phải những khó khăn.

Theo ma trận này, nếu nhóm vẫn trong giai đoạn hình thành và nếu dự án vẫn ở giai đoạn lập kế hoạch ban đầu, thì hai kỹ thuật phù hợp nhất để sử dụng là điều tiết hoặc cạnh tranh.

Lý giải cho điều này là kỹ thuật điều tiết sẽ giúp mọi người hiểu người khác tốt hơn, kỹ thuật này có thể được thực hiện tốt nhất trong một số nền văn hóa như ở Châu Á. Sau đó, kỹ thuật tốt nhất tiếp theo được sử dụng là cạnh tranh. Thông qua kỹ thuật này, mọi người có thể có cơ hội để thử thách ý tưởng của người khác dựa trên dữ liệu thực tế. Đây là kỹ thuật có thể thực hiện tốt nhất ở các nền văn hóa Phương Tây. Khi xung đột xảy ra sớm trong vòng đời dự án, hậu quả của những đối lập này sinh sôi không ảnh hưởng lớn đến tiến độ của dự án.

Hình thành	ACCO/COMPE			
Sóng gió	COMPE	COMPE		
Quy chuẩn	COMPE, COMP	COMPE, COLLA	COMPE, COLLA	COLLA
Hoạt động thành công	COLLA	COMPE, COLLA	COMPE, COLLA	COLLA

Lập kế hoạch ban đầu Triển khai Sprint Lập kế hoạch liên tục Kết thúc

Hình 10.3

Các kỹ thuật giải quyết phù hợp nhất khi xung đột xảy ra trong nhóm.

Điểm tốt sau khi những kiểu xung đột này xảy ra ở giai đoạn đầu đó là mọi người sẽ hiểu quan điểm của người khác tốt hơn và hướng tới cách tiếp cận mang tính cộng tác hơn. Nếu một nhóm dự án không thể tự giải quyết được các xung đột của mình, tất nhiên là ScrumMaster và Product Owner nên can thiệp gián tiếp, để cố gắng cho mọi thứ lại nằm trong vòng kiểm soát.

ĐIỀU KIỆN TUYỆT VỜI ĐỂ LÀM VIỆC NHÓM

Công việc sẽ tuyệt vời nếu không có những xung đột, nhưng do xung đột thường xảy ra và gây tổn hại tới tiến độ của nhóm, nên chúng ta cần làm gì để phòng tránh?

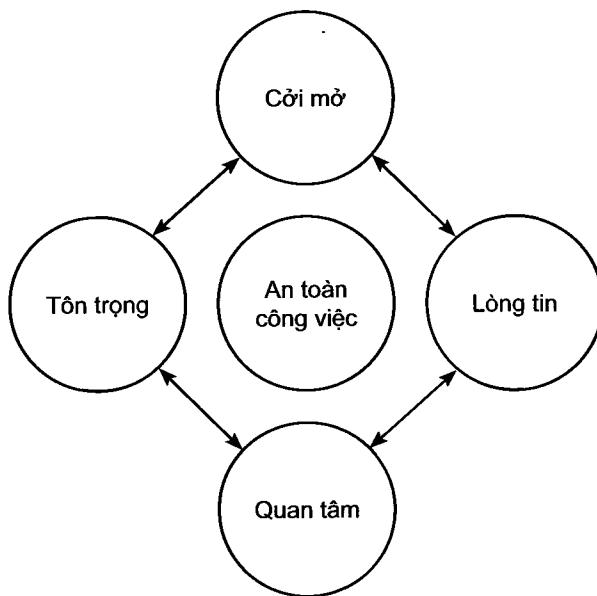
Chúng tôi đã phát hiện ra một số mô hình tương tự xuất hiện nhiều lần. Chúng tôi gọi các mô hình này là (nǎm) điều kiện tuyệt vời để làm việc nhóm và chúng được thể hiện trong Hình 10.4. Những điều kiện này giúp các thành viên trong nhóm cộng tác tốt với nhau.

Với tư cách là một thành viên, bạn nên đến với nhóm với một tư duy mở và nghĩ đến nhóm thay vì cá nhân, cố gắng xem xét mọi việc trên quan điểm của người khác và mong muốn hiểu các thành viên khác trong nhóm.

Chỉ với tư duy mở bạn mới có thể hy vọng hòa nhập thành công vào một nhóm gồm nhiều con người với những tính cách khác nhau.

Thứ hai, bạn nên quan tâm mọi người. Bạn muốn tất cả mọi người đều được lắng nghe và có cơ hội để đóng góp. Càng nhiều người cảm thấy họ được lắng nghe, nhóm càng có nhiều khả năng hoạt động tốt.

Sau đó, tìm cơ hội để thể hiện sự tôn trọng dành cho ai đó trong nhóm.

**Hình 10.4**

Các điều kiện tuyệt vời để làm việc nhóm

Tiếp đến, nhớ rằng lòng tin là chất keo cần có để mọi thứ gắn kết với nhau trong một tinh thần đồng đội tuyệt vời.

Cuối cùng, chúng tôi bổ sung thêm an toàn công việc, như là điều kiện trung tâm của tinh thần làm việc nhóm tuyệt vời bởi vì mọi người không thể làm việc năng suất hoặc hòa hợp với người khác khi mà họ lo sê mất việc.

Đây là điểm mà ScrumMaster và Product Owner có thể trợ giúp bằng cách làm việc chặt chẽ với người quản lý nhóm trực tiếp. Điều này sẽ giúp các thành viên trong nhóm cảm thấy làm việc an toàn bên nhau, như chúng tôi sẽ thảo luận ở chương tiếp theo.

Với kết quả về cộng tác nhóm, dựa trên sự thành công khi họ làm việc cùng nhau, chúng ta có thể xác định ra ba loại nhóm là: (1) nhóm hiệu suất cao, (2) nhóm hiệu suất trung bình và (3) nhóm hiệu suất thấp hoặc tầm thường.

Các nhóm hiệu suất cao sẽ vui vẻ, cởi mở và quan tâm lẫn nhau. Ngược lại, các nhóm có hiệu suất thấp thường đặc trưng bởi sự im lặng trong các cuộc họp, những nụ cười gượng và thái độ che đậy. Các thành viên trong nhóm hiệu suất trung bình chỉ làm việc đều đặn mỗi ngày, chỉ làm những gì cần thiết trong phiên làm việc và được trả tiền, nhưng họ không đem lại giá trị gì cho tổ chức. Khác biệt duy nhất của họ với nhóm hiệu suất thấp hoặc tầm thường là họ không viện tới sự che đậy và các trò mang màu sắc chính trị.

TÓM TẮT

Sự thiếu vắng người quản lý dự án trong Scrum không có nghĩa là trách nhiệm này bị bỏ rơi. Và cũng không phải là chúng ta đã loại bỏ người lãnh đạo dự án.

Không giống như một dự án truyền thống theo kiểu mệnh lệnh và kiểm soát, nhóm trong Scrum là tự tổ chức. Điều này có nghĩa là họ được trao quyền để tự ra quyết định về cách thức mà họ muốn chia sẻ trong công việc. Không có ai hay bất kỳ người quản lý dự án nào chỉ cho nhóm phải làm những gì.

Bởi vì nhóm là tự quản, họ cần học cách để tự giải quyết các xung đột. Đây là lý do chúng tôi đã đề cập trong chương này, không chỉ các kiểu tính cách Keirsey khác nhau để giúp các thành viên nhóm hiểu nhau, mà còn có những kỹ thuật khác nhau để xử lý xung đột.

Việc xem xét các giai đoạn xảy ra xung đột rất quan trọng. Một xung đột xảy ra trong giai đoạn hình thành nhóm cần những kỹ thuật xử lý khác với khi xung đột xảy ra ở giữa các Sprint.

Do các xung đột gây tổn hại tới tiến độ của nhóm, chúng ta cần tự vấn bản thân làm sao để tránh chúng. Một trong những câu trả lời cho câu hỏi này được gọi là năm điều kiện tuyệt vời để làm việc nhóm. Bốn trong số năm điều kiện này là Cởi mở, Lòng tin, Quan tâm và Tôn trọng. Điều kiện cuối cùng là An toàn công việc. Trong khi các thành viên nhóm ít có khả năng kiểm soát sự an toàn công việc của họ, Product Owner và ScrumMaster có ảnh hưởng nhiều hơn đến điều này bằng cách làm việc chặt chẽ với người quản lý trực tiếp của nhóm.

CHƯƠNG 11

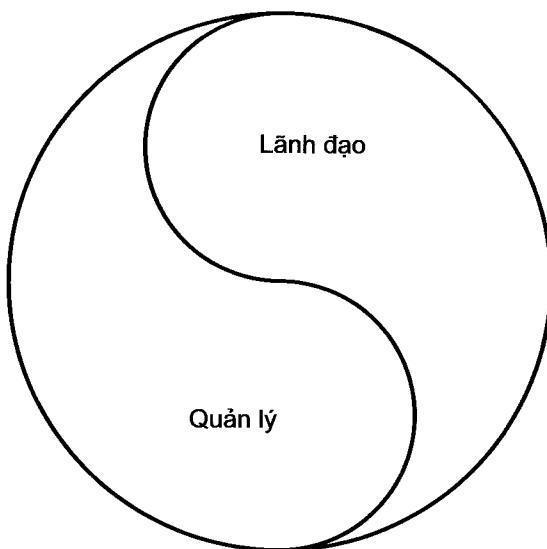
BẢN CHẤT MỚI CỦA QUẢN LÝ VÀ LÃNH ĐẠO TRONG DỰ ÁN SCRUM

Mặc dù bạn đã học được rằng không có nhà quản lý dự án trong Scrum, nhưng điều này không có nghĩa là không có sự quản lý hay lãnh đạo trong một dự án Scrum. Quan niệm sai lầm này đã được chỉnh sửa lại trong một bài thảo luận của Mike Cohn vào ngày 25/8/2009 trên InformIT. Bạn có thể tìm thấy bài viết này tại www.informit.com/articles/article.aspx?p=1382538.

Không giống như trong môi trường mệnh lệnh và kiểm soát truyền thống, trách nhiệm quản lý bây giờ được phân chia cho ba thành phần trong nhóm Scrum là: ScrumMaster, Nhóm Phát triển và Product Owner. Mặc dù ScrumMaster và Product Owner không trực tiếp quản lý nhóm, nhưng họ chịu trách nhiệm báo cáo dự án lên cấp quản lý kinh doanh. Điều này đúng với quan điểm của Ken Schwaber trong cuốn *Agile Project Management with Scrum* (tạm dịch: Quản trị Dự án linh hoạt với Scrum), cho dù trong dự án Scrum việc báo cáo không còn là nhiệm vụ nữa mà chỉ là theo nhu cầu.

Bất cứ ai cũng có thể được xem là lãnh đạo trong một dự án Scrum, miễn là người đó có một số ảnh hưởng lên nhóm. Trước sự ngạc nhiên của một số những người theo thuyết thuần túy cùng sự thích thú của chúng tôi, Mike Cohn đã đề cập trên blog của mình về Product Owner, ScrumMaster và thậm chí cả nhà quản lý chức năng như là những điển hình về lãnh đạo có ảnh hưởng đến hướng đi và thành công của một dự án Scrum, chúng tôi hoàn toàn đồng ý với ông ấy. Bạn có thể ghé thăm blog của Mike Cohn tại địa chỉ

<http://blog.mountaingoatsoftware.com/the-role-of-leaders-on-a-self-organizing-team>.



Hình 11.1

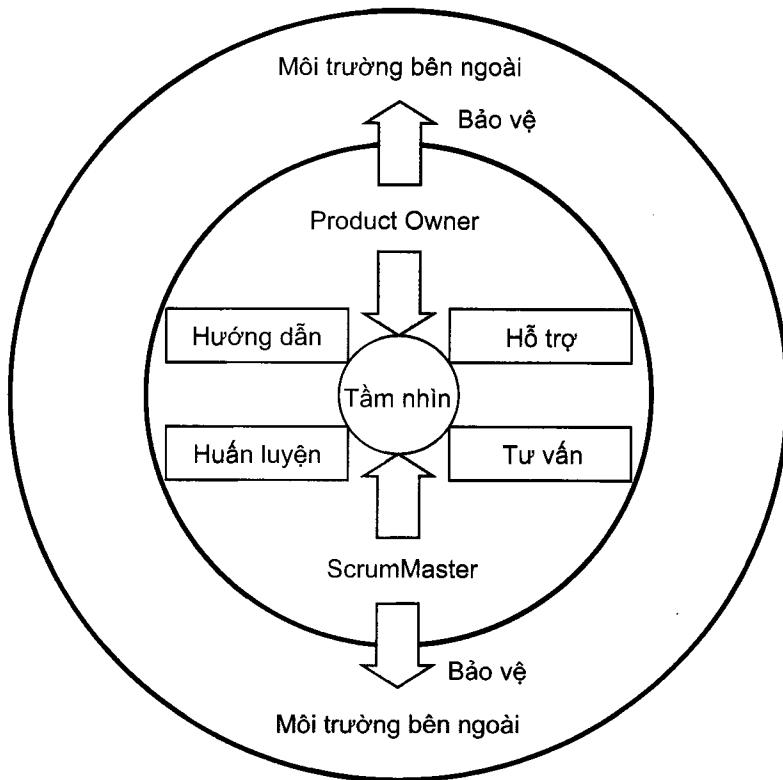
Quản lý và lãnh đạo như là hai nửa của vòng tròn.

Nhưng để diễn giải những gì Mike Cohn đã viết trên blog rằng lãnh đạo một nhóm tự quản lý cần phải làm nhiều việc hơn là những việc như mua thức ăn hay là để nhóm tự do làm những gì họ muốn. Các lãnh đạo trong một nhóm dự án Scrum ảnh hưởng đến nhóm một cách gián tiếp, đây chính là những gì mà chúng tôi đề cập trong chương này.

Điều cuối cùng, nhưng không kém quan trọng, mọi người thường so sánh quản lý với lãnh đạo như là hai việc đối lập nhau. Chúng tôi thì nghĩ rằng chúng bổ sung cho nhau thay vì chống đối nhau và đây chính là lý do tại sao chúng tôi cho rằng chúng là hai nửa của hình tròn như đã minh họa trong Hình 11.1. Không nửa nào hơn nửa nào cả, các vai trò này thường hòa quyện vào nhau.

Như những gì mà bậc thầy quản lý Peter Ducker đã nói: “chiến lược mà không được thực thi thì chỉ là một tầm nhìn”. Chúng tôi cho rằng lãnh đạo mà không quản lý thì giống như là có một chiến lược nhưng không có khả năng thực thi nó.

Áp dụng điều này vào Scrum, có nghĩa là cho dù với một nhóm tự tổ chức, ScrumMaster và Product Owner cũng không nên ngần ngại sử dụng các kỹ năng lãnh đạo và quản lý để nói chuyện với nhóm phát triển khi họ nhận thấy dự án không tiến triển tốt hay nếu nhóm không thực hiện đúng như những gì họ phải làm.



Hình 11.2

Hai phương diện của quản lý và lãnh đạo.

Mọi thứ trở nên thú vị hơn khi bạn nhận ra rằng họ phải thực hiện những kỹ năng này với sự tinh tế và giàn tiếp, cả với nhóm hay với những người ngoài nhóm có ảnh hưởng đến năng suất của nhóm (Hình 11.2).

Những gì mà Hình 11.2 muốn truyền tải là tồn tại hai phương diện với hai vai trò quản lý dự án và lãnh đạo nhóm của Product Owner và ScrumMaster: một mặt là đối với nội bộ nhóm và mặt khác là với thế giới bên ngoài.

Theo truyền thống, người ta đã quyết định rằng chức danh ScrumMaster phải bảo vệ nhóm khỏi những xáo trộn từ bên ngoài trong suốt Sprint, kể cả từ Product Owner (có thể là người của nhóm) khi họ yêu cầu nhóm phải quan tâm đến một số yêu cầu mới.

Tuy nhiên, trong thực tế, rõ ràng là có nhiều trường hợp mà Product Owner không phải là người quản lý của một đơn vị kinh doanh, nhưng lại là người phải làm việc với các đơn vị kinh doanh khác và đôi khi phải giải quyết xung đột về độ ưu tiên giữa các đơn vị này.

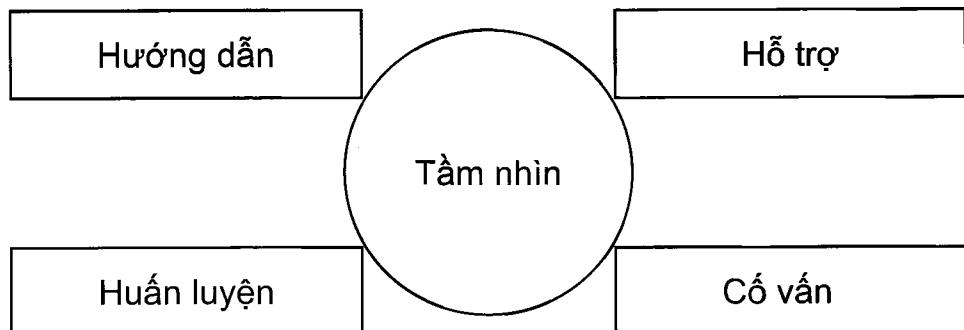
Điều này để nói lên rằng, trong một số trường hợp thì Product Owner cần phải đẩy lùi các đơn vị kinh doanh khác để có thể bảo vệ nhóm, khi có những yêu cầu mới mà các đơn vị kinh doanh này muốn đưa vào giữa một Sprint. Đây chính là thời

Chương 11 • Bản chất mới của Quản lý và Lãnh đạo trong dự án Scrum

điểm mà Product Owner thực sự thể hiện là một người lãnh đạo kiểu phục vụ, được ScrumMaster hỗ trợ nhằm cố gắng bảo vệ nhóm khỏi những xáo trộn từ bên ngoài.

Cho dù thảo luận của chúng tôi dưới đây sẽ hơi khác với thảo luận CDE (Container, Difference, Exchange) về việc làm cách nào để lãnh đạo một nhóm tự tổ chức trong cuốn sách tuyệt vời của Mike Cohn, “Succeeding with Agile Using Scrum” (tạm dịch: Thành công với Agile sử dụng Scrum), phần thảo luận này có chung tiền đề với cuốn sách của Mike Cohn, vì trong cuốn sách này thì nhóm dự án Scrum cần khá nhiều sự quản lý và lãnh đạo để có thể hoạt động tốt được, trái ngược với những niềm tin phổ biến.

Để hiểu rõ hơn về vai trò quản lý và lãnh đạo của cả Product Owner và ScrumMaster, hãy xem xét phương diện nội bộ nhóm như được trình bày ở trong Hình 11.3, để xem đâu là những khía cạnh khác nhau của họ và Product Owner cùng ScrumMaster có thể và nên làm gì để cải thiện tiến độ của nhóm hướng tới mục tiêu dự án.



Hình 11.3

Thước đo về khả năng lãnh đạo kiểu phục vụ của ScrumMaster và Product Owner đối với nội bộ nhóm.

Tầm nhìn: Product Owner là vai trò có liên quan nhiều nhất trong việc chịu trách nhiệm cho tầm nhìn, các mục tiêu và yêu cầu nghiệp vụ của sản phẩm. Bằng cách đặt những câu hỏi thăm dò, ScrumMaster có thể giúp Product Owner xây dựng tầm nhìn và xác định các mục tiêu của sản phẩm một cách rõ ràng hơn. Việc có một tầm nhìn sản phẩm rõ ràng sẽ giúp Product Owner trả lời bất cứ câu hỏi nào mà nhóm có thể đưa ra liên quan đến yêu cầu nghiệp vụ và định hướng của sản phẩm.

Hỗ trợ: Đây là điều mà cả Product Owner và ScrumMaster sẵn sàng cung cấp. Thông thường, vai trò được biết đến phổ biến của ScrumMaster là giúp loại bỏ những trở ngại và hỗ trợ nhóm hiểu về quy trình Scrum. Nhưng loại bỏ các trở ngại là việc mà Product Owner có thể và nên làm một khi chúng xuất hiện, chẳng hạn với trường hợp đã được đưa ra ở trên, nhằm chống lại những cá nhân kinh doanh gây sức ép lên các thành viên của nhóm để họ làm việc với những hạng mục mới không có trong kế hoạch của Sprint hiện tại.

Bản chất mới của Quản lý và Lãnh đạo trong dự án Scrum

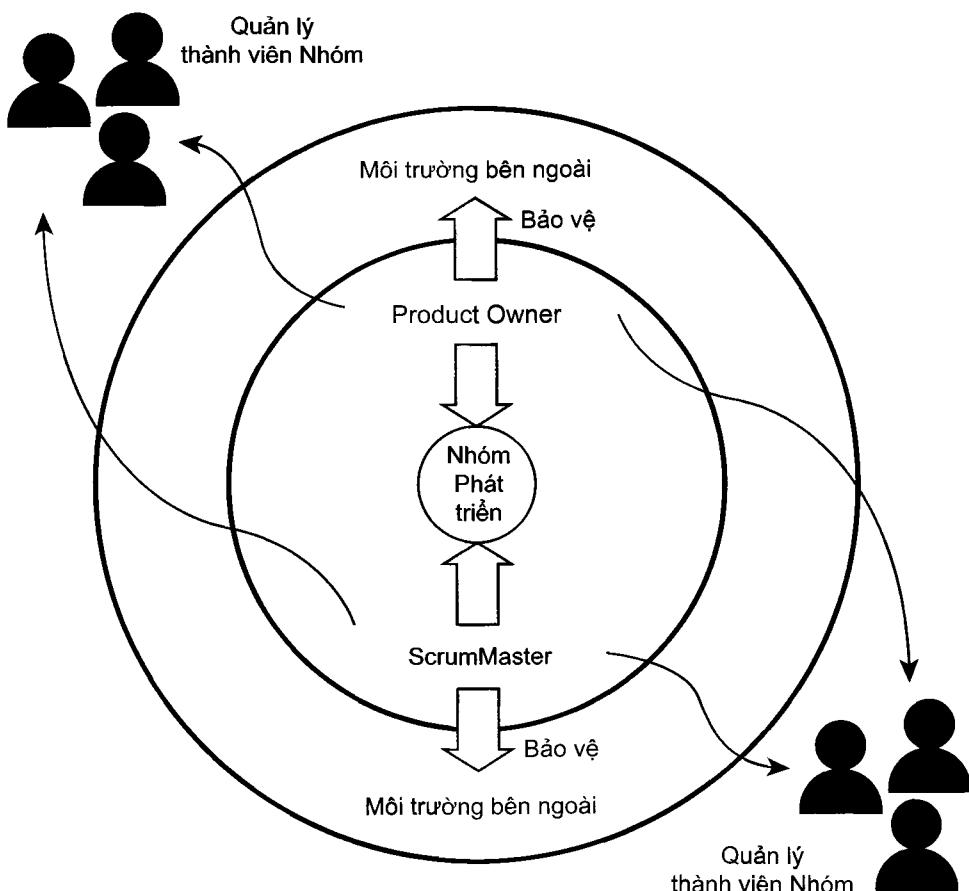
Hướng dẫn: Ở đây chúng tôi muốn nói về năng lực của ScrumMaster và Product Owner, hai trong ba kiểu lãnh đạo mà Mike Cohn nhắc đến trong cuốn sách của Ông, nhằm phát huy những ảnh hưởng tích cực lên nhóm, dù là để thúc đẩy sự cộng tác nhiều hơn hay là hiệu suất tốt hơn.

Cố vấn: Bất cứ lúc nào có thể, ScrumMaster và Product Owner không nên ngần ngại chia sẻ với nhóm những gì mà họ biết nhằm giúp các thành viên nhóm làm công việc của họ tốt hơn, đặc biệt là khi Product Owner và ScrumMaster là những chuyên gia trong các lĩnh vực đó.

Huấn luyện: Đây là một trong những khía cạnh ít được thực hành nhất của quản lý và lãnh đạo trong môi trường mệnh lệnh và kiểm soát truyền thống. Huấn luyện là một kỹ thuật mà Product Owner và ScrumMaster có thể và nên sử dụng thường xuyên để khai phóng tiềm năng của các thành viên trong nhóm.

Sau khi đã xem lại những khía cạnh khác nhau ở phương diện nội bộ nhóm của lãnh đạo nhóm, hãy chuyển sự chú ý đến trách nhiệm bên ngoài của ScrumMaster và Product Owner để xem họ nên làm và có thể làm gì để giúp nhóm với môi trường bên ngoài ví dụ với quản lý trực tiếp của nhóm và các bên liên quan khác. Có một số việc khác cũng rất quan trọng mà Product Owner và ScrumMaster nên làm nhằm cung cấp cho nhóm điều kiện thiết yếu nhất để làm việc nhóm tốt ngoài việc hỗ trợ, hướng dẫn và huấn luyện nhóm, đó là: yêu tố an toàn công việc như đã được nói đến trong Chương 10.

Hình 11.4 chỉ ra các dạng can thiệp mà ScrumMaster và Product Owner nên làm. Chẳng hạn, với các nhà quản lý chức năng của nhóm, ScrumMaster và Product Owner có thể giải thích với họ Scrum là gì và tại sao các thành viên nhóm không nên bị trừng phạt bởi vì từ chối làm việc với những vấn đề không dự tính trước, những thứ có thể cản trở công việc của nhóm trong Sprint.



Hình 11.4

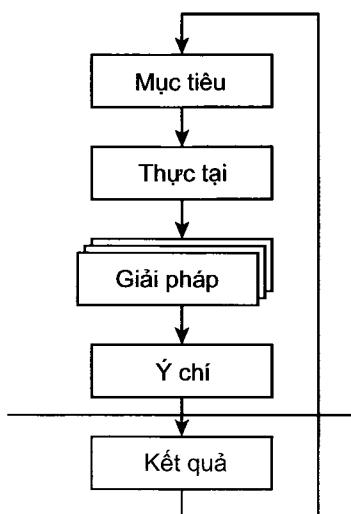
Sự can thiệp bên ngoài khi có sự bảo vệ thành viên nhóm.

HUẤN LUYỆN ĐỂ ĐẠT HIỆU QUẢ CAO NHẤT: MÔ HÌNH GROW

Mô hình GROW (viết tắt của Goal - Mục tiêu, Reality - Thực tại, Options - Giải pháp và Will - Ý chí) là một kỹ thuật nổi tiếng dành cho huấn luyện, đây là một trong những kỹ thuật được các lãnh đạo kiều phục vụ thích sử dụng nhằm nâng cao hiệu suất của một nhóm. Kỹ thuật này có thể được sử dụng bởi bất cứ ai và không đòi hỏi bất cứ sự đào tạo đặc biệt nào. GROW cung cấp hướng tiếp cận có cấu trúc và hiệu quả cho người lao động và quản lý nhằm xây dựng các mục tiêu và xác định cách để đạt được chúng.

Mỗi huấn luyện viên đều có cách huấn luyện riêng của họ. Nhưng phần tiếp theo đây sẽ đưa ra một ví dụ về việc cấu trúc một phiên huấn luyện theo mô hình GROW. Mô hình này đã được chúng tôi mở rộng thêm một giai đoạn mới (Kết quả) nhằm cung cấp một cơ hội cho sự cải tiến liên tục.

- Thiết lập mục tiêu:** Trước tiên, hãy giúp các thành viên trong nhóm của mình định nghĩa được mục tiêu của họ. Để làm được điều này, bạn có thể sử dụng kỹ thuật SMART được thảo luận trong Chương 4, “Thu thập các yêu cầu trực quan cho Product Backlog”. Với SMART (Rõ ràng, Đo được, Có thể đạt được, Thực tế, Dựa trên thời gian) thì mục tiêu có thể đạt được trong một khung thời gian cho trước.



Hình 11.5

Kỹ thuật GROW mở rộng.

- Kiểm tra Thực tại (hiện tại):** Tiếp đó, hãy yêu cầu thành viên nhóm mô tả Thực tại của họ, nghĩa là thực tại mà họ đang sống. Đây là một bước quan trọng. Mọi người thường có gắng giải quyết một vấn đề mà không để ý đến điểm khởi đầu của họ và do vậy thường thiếu những thông tin cần thiết để giải quyết vấn đề một cách hiệu quả. Như một câu ngạn ngữ Trung Quốc đã nói “Nếu bạn không biết mình đang ở đâu thì không có bẩn đồ nào có thể giúp bạn đến được nơi mình muốn”. Khi thành viên trong nhóm nói cho bạn biết về Thực tại của họ, giải pháp cho vấn đề sẽ bắt đầu xuất hiện.
- Khám phá các Giải pháp:** Một khi bạn và các thành viên trong nhóm đã xây dựng được các Mục tiêu và tìm hiểu về Thực tại (Hiện tại), đây là lúc để khám phá tất cả những giải pháp có thể để giải quyết vấn đề. Hãy chống lại sự cám dỗ muôn tự mình thực hiện bước này. Thay vào đó, hãy giúp các thành viên trong nhóm tự tìm ra những giải pháp tốt nhất có thể, sau đó hãy thảo luận với nhau về chúng.

Trước tiên hãy để cho các thành viên nhóm được trình bày và đưa ra các ý tưởng. Sau đó hãy đưa ra gợi ý của bạn.

4. **Xây dựng Ý chí:** Thông qua việc khảo sát Thực tại và khám phá các Giải pháp, thành viên nhóm sẽ có được một ý tưởng tốt về việc làm cách nào để đạt được Mục tiêu của mình.

Điều này là rất tuyệt vời, thế nhưng có thể vẫn chưa đủ. Do vậy, bước cuối cùng của bạn với vai trò huấn luyện viên là đảm bảo các thành viên nhóm cam kết thực hiện một số hành động cụ thể. Làm như vậy thì bạn sẽ giúp họ hình thành ý chí và động lực để cải tiến.

5. **Kiểm tra Kết quả** (giai đoạn do chúng tôi mở rộng): Thông qua việc kiểm tra các kết quả đạt được so với những mục tiêu đã thiết lập ban đầu, bạn và các thành viên trong nhóm có thể nhận ra khiếm khuyết và những việc cần phải hoàn thành để cải thiện hiệu suất của mình.

Sau đây là một ví dụ hữu ích về mô hình GROW, hãy tưởng tượng bạn đang lên kế hoạch cho một chuyến đi.

Trước tiên, bạn bắt đầu với một bản đồ để hình dung địa điểm nhóm muốn đến (Mục tiêu) và xác định vị trí mà nhóm đang ở (Thực tại). Sau đó bạn sẽ tìm hiểu một vài con đường (Lựa chọn hay Giải pháp) mà nhóm có thể đi. Cuối cùng, bạn sẽ giúp đảm bảo rằng nhóm sẽ thực hiện chuyên đi (Ý chí) và chuẩn bị sẵn sàng cho những điều kiện và trở ngại mà họ có thể gặp trên đường. Trong khi họ đang đi trên đường hướng tới đích (mục tiêu đã thiết lập) thì bạn phải nhắc họ định kỳ kiểm tra lại những gì đã đạt được (Kết quả) nhằm xác định sự sai lệch và những hành động mà họ có thể cần phải thực hiện để đưa mình trở lại đúng quỹ đạo cho đến khi tới đích (Mục tiêu).

ĐẶC ĐIỂM CỦA MỘT NHÀ LÃNH ĐẠO VÀ QUẢN LÝ CHU ĐÁO

Trước khi tiếp tục, chúng tôi nghĩ rằng để trở thành một lãnh đạo tốt hoặc một nhà quản lý chân chính bạn nên:

1. **Chân thật:** Điều này có nghĩa rằng bạn cần nói đi đôi với làm. Đây cũng có nghĩa là khi mắc lỗi, bạn không ngần ngại thừa nhận nó. Bởi vì không có ai là hoàn hảo, mọi người sẽ không giữ sự oán giận một khi bạn thừa nhận những lỗi lầm của mình; ngược lại họ sẽ tôn trọng bạn hơn nhờ sự trung thực. Và nếu mọi người tôn trọng bạn, nhiều khả năng họ sẽ đi theo bạn thay vì những người không được họ tôn trọng.
2. **Cởi mở:** Một trong những đặc điểm quan trọng nhất của một nhà lãnh đạo là sự cởi mở với các ý tưởng và ý kiến của mọi người. Không dập tắt bất cứ một gợi ý nào đến từ nhóm hoặc những người quanh bạn mà không xem xét giá trị của nó.
3. **Thành thật:** Điều này có nghĩa rằng bạn không nên cố gắng cư xử theo cách không đúng với giá trị và niềm tin của mình. Nói cách khác, bạn không nên cố tham dự trò chơi bằng cách cố là một người nào đó không phải bạn.

Đặc điểm của một Nhà lãnh đạo và Quản lý chu đáo

4. **Sự sẵn sàng:** Chúng ta luôn bận rộn trong môi trường doanh nghiệp, nhưng khi là một lãnh đạo, mọi người kỳ vọng bạn luôn sẵn sàng cho họ lời khuyên, lắng nghe những quan ngại của họ và phản hồi hoặc làm sáng tỏ một vấn đề gì đó mà họ chưa hiểu rõ. Do đó, hãy điều chỉnh lại lịch làm việc nếu cần thiết, nhưng cố gắng để bạn luôn sẵn sàng ngồi cùng thành viên nhóm ít nhất là một cuộc trò chuyện ngắn trước khi bạn tìm hiểu nếu các vấn đề hoặc mối bận tâm của họ cần phải được làm sáng tỏ hơn hoặc họp bàn thêm. Là một lãnh lãnh đạo kiểu phục vụ, bạn nên giữ cánh cửa của mình mở càng nhiều càng tốt để báo hiệu rằng dù bạn là một lãnh đạo bận rộn nhưng vẫn luôn sẵn sàng hỗ trợ nhóm.
5. **Quan tâm tới người khác:** Trong một xã hội mà thành công và sự thăng tiến là những từ khóa, có thể sẽ khó khăn khi đòi hỏi bạn quan tâm tới người khác. Nhưng nếu suy nghĩ thêm, cuối cùng sự thành công mà bạn đang quan tâm là việc bạn giúp các thành viên trong nhóm đạt được thành công trong công việc của họ. Vì vậy, khi mà vấn đề đang chuyển sang mức tệ hại nhất, hãy cố gắng nghĩ tới việc chăm lo những đồng đội của bạn. Điều này chỉ có thể giúp bạn thành công hơn mà thôi. Nhưng nếu bạn thực sự là một nhà lãnh lãnh đạo kiểu phục vụ, người quan tâm đến người khác mà không nghĩ tới lợi ích của bản thân, thì điều đó có thể giúp nâng hình ảnh của bạn trở thành một nhà lãnh đạo có tư cách tốt.

TÓM TẮT

Thực tế, việc nhóm phát triển Scrum là một nhóm tự tổ chức không có nghĩa là không có quản lý dự án hay vai trò lãnh đạo trong một dự án Scrum giống như người ta vẫn thường quan niệm sai.

Không giống với môi trường mệnh lệnh và kiểm soát truyền thống, trách nhiệm quản lý bây giờ được phân chia cho ba thành phần trong nhóm Scrum: ScrumMaster, Nhóm Phát triển và Product Owner.

Đề cập về lãnh đạo, bất cứ ai cũng có thể trở thành lãnh đạo miễn là người này có ảnh hưởng tới nhóm, nhưng những người mà chúng tôi nghĩ có thể gây nhiều ảnh hưởng tích cực tới nhóm trong vai trò lãnh đạo là Product Owner và ScrumMaster. Đây là hai trong số ba kiểu mẫu lãnh đạo chính được Mike Cohn đề cập tới trong cuốn sách có tên là *Succeeding with Agile using Scrum* (tạm dịch: Thành công với Agile sử dụng Scrum).

Nhằm tổng hợp các khía cạnh khác nhau về lãnh đạo cùng với các kỹ thuật mà Product Owner và ScrumMaster có thể sử dụng, nội dung chương đã bao hàm một thước đo về lãnh đạo kiểu phục vụ. Thước đo này gồm năm yếu tố là tầm nhìn, hỗ trợ, hướng dẫn, tư vấn và huấn luyện.

Trong khi hầu hết các khía cạnh này đã phần nào được biết đến, thì huấn luyện vẫn còn tương đối mới, ít nhất là trong phát triển phần mềm. Đó là lý do chúng tôi đã trình bày chi tiết hơn về huấn luyện trong chương này và đặc biệt là trong mô hình GROW. Ở đó chúng tôi bổ sung một giai đoạn mới (Kết quả) với mục đích cung cấp một cơ hội để cải tiến liên tục nhằm hỗ trợ cho các ý tưởng cải tiến liên tục trong Agile.

Để kết thúc cuộc thảo luận về quản lý và lãnh đạo, chương này cũng nhắc nhở thêm về những phẩm chất mà Product Owner và ScrumMaster nên có để trở thành lãnh đạo và quản lý tốt, đó là: (1) chân thật, (2) cởi mở, (3) thành thật, (4) sự sẵn sàng và (5) quan tâm (tới người khác).

CHƯƠNG 12

LÀM THẾ NÀO ĐỂ THÍCH ỦNG VỚI SCRUM MÀ KHÔNG PHÁ BỎ TÍNH LƯNHOẠT CƠ BẢN HOẶC TRIỀN KHAI SCRUMBUT TIÊU CỰC

Trừ khi bạn đủ may mắn để làm việc trong một công ty mà các quản lý cấp cao và cấp trung đều ủng hộ Scrum và toàn bộ công ty hoặc phòng CNTT đã hoàn tất việc tái tổ chức để dùng Scrum. Nếu không dự án hoặc môi trường công ty có thể sẽ không được hoàn hảo như Scrum quy định.

Thông thường, bạn sẽ thấy mình ở trong một công ty mà người quản lý có thể đã nghe nói về Scrum hoặc thậm chí đã từng thử nó cho một vài dự án, nhưng họ không sa thải tất cả những chuyên gia và tổng hợp gia (người phụ trách đa ngành) đã thuê hoặc thay thế các quản trị dự án của họ bằng các ScrumMaster. Ngoại trừ một số công ty phần mềm thương mại, hầu hết các tập đoàn ở Hoa Kỳ, đại diện cho khoảng 99% các công ty, vẫn tổ chức theo lối cũ, có nghĩa là phòng CNTT được phân chia thành các bộ phận chức năng. Để thấy được sự thực này, bạn hãy vào trang Carrier (nghề nghiệp) trên website của Capital One, một trong những công ty được xem là đã chấp nhận Scrum và quan sát xem có bao nhiêu vị trí như chuyên gia phân tích nghiệp vụ (business analyst - BA), phân tích hệ thống (business system analyst) và quản trị dự án (project manager) vẫn đang được đăng tuyển. Quan sát cũng cho thấy kết quả tương tự với những công ty lớn khác được biết là đã chuyển sang Agile/Scrum, chúng tôi chỉ đề cập một số ở đây như Verizon, Sabre, NBC Universal, General Dynamics, Texas Instruments và American Airlines.

Do đó, thông thường bạn và nhóm Scrum sẽ phải tiến hành đào tạo và thương thuyết rất nhiều để làm cho mọi việc xảy ra. Tất cả những nội dung sẽ trình bày trong chương là những kiến thức bạn cần để thích ứng Scrum (mà không phá vỡ gốc rễ Agile) và làm cho Scrum hoạt động tốt trong công ty đúng như những thiết lập hiện tại. Sau hết, Scrum chỉ là một phương tiện để đi tới đích và vì lý do này bạn không nên máy

Chương 12 ▪ Làm thế nào để thích ứng với Scrum mà không phá bỏ tính linh hoạt cơ bản hoặc triển khai ScrumBut tiêu cực

móc theo xu hướng của một số chuyên gia, mà nên thực tế như một người đang thực hành Scrum.

Để tăng độ thuyết phục về việc bạn có thể và phải thích ứng Scrum với môi trường của mình, hãy tham khảo bài viết “Scrum in Church” (tạm dịch: Scrum trong Nhà thờ) của một trong những cha đẻ Scrum, Jeff Sutherland và vợ ông ấy, Arline C. Sutherland. Trong bài viết này, họ cho rằng sự thích ứng là cần thiết khi môi trường thay đổi.

Trong bài viết này, Arline nói rằng, các tình nguyện viên của bà ấy thay vì tiến hành các buổi họp đúng hằng ngày, họ chỉ thực hiện một lần vào mỗi Thứ Sáu khi họ tới nhà thờ, bởi họ không đến nhà thờ hằng ngày.

Vì vậy, nếu bạn đã bị thuyết phục rằng chúng ta có thể thích ứng Scrum cho môi trường của mình nếu cần thiết, thì câu hỏi lúc này là chúng ta cần thích ứng Scrum như thế nào mà không phá bỏ tính linh hoạt cơ bản hoặc triển khai ScrumBut tiêu cực.

LÀM THẾ NÀO ĐỂ THÍCH ỦNG SCRUM MÀ KHÔNG TRIỂN KHAI “SCRUMBUT” TIÊU CỰC VỚI NHỮNG NGUY BIỆN

Trước đây chúng tôi đã tự hỏi mình câu hỏi này khi cảm thấy thất bại hoàn toàn trong lần đầu tiên sử dụng Scrum. May mắn thay, sau đó một vài năm, khi mọi thứ bắt đầu lắng xuống và chúng tôi đã bắt đầu hiểu nhiều hơn về cách thức để thích ứng Scrum mà không phản lại các nguyên tắc Agile.

Sự việc xảy ra xoay quanh hai loại “ScrumBut” đó là: ScrumBut tốt và ScrumBut xấu. ScrumBut được coi là xấu là loại giống với những gì mà Ken Schwaber đã gọi là “ScrumBut,” hoặc những nguy biện tệ hại hay việc áp dụng sai Scrum.

Hãy xem một ví dụ về “ScrumBut” tiêu cực: “Chúng tôi triển khai Scrum, nhưng không có bất cứ ai sẵn sàng đảm nhiệm vai trò Product Owner.”

ScrumBut được coi là tốt giống như Jurgen Appelo, CIO và blogger nổi tiếng về Agile, viết trên blog của ông ấy (“ScrumBut là một phần tốt nhất của Scrum”) năm 2009, bài viết đưa ra một số ví dụ về “ScrumBut” tích cực, được ông ấy và chúng tôi gọi là những thích ứng Scrum.

Chúng tôi đồng ý với ý kiến của Jurgen Appelo khi ông tiếp tục cho rằng sự thích ứng là những gì tốt nhất mà các nhóm có thể học được từ các Cải tiến Scrum (Scrum Retrospective). Cũng trên trang blog của Jurgen Appelo, Mike Cottmeyer, sáng lập viên của APLN tại Atlanta (Hoa Kỳ), cũng đồng ý rằng bệnh giáo điều của nhiều nhà thực hành Scrum là khá phản tác dụng. Như chúng tôi, ông ấy tin rằng việc tạo ra những thích ứng đó cần được khuyến khích trừ khi nó tiềm ẩn những trở ngại chính đáng.

Chúng tôi tin rằng điểm khác biệt tinh tế giữa hai loại ScrumBut chủ yếu dựa vào lý do của bạn là tốt hay xấu trong trường hợp bạn không làm một điều gì đó theo quy định thông thường của Scrum. Để nhận biết một “ScrumBut” là tích cực hay tiêu cực sẽ đòi hỏi bạn mất nhiều công sức thực hành và đặc biệt cần hiểu rõ về tinh thần Agile mà chúng tôi đã chia sẻ trong Chương 1, “Chuẩn bị cho Agile và Scrum”.

VÍ DỤ VỀ CÁC TÌNH HUỐNG THÍCH ỨNG SCRUM

Để hiểu một cách thấu đáo, dưới đây là một số ví dụ về những tình huống mà chúng tôi đã thấy Scrum được điều chỉnh một phần hoặc thậm chí nhóm đã sử dụng Scrum hơi khác so với nguyên bản. Điều này cho phép chúng ta vẫn sử dụng Scrum bất chấp một số điều kiện của Scrum không được đáp ứng.

Khía cạnh Tổ chức

Phải làm gì nếu tổ chức của bạn vẫn yêu cầu báo cáo trạng thái dự án trong các cuộc họp PMO hằng tuần, sử dụng một mẫu báo cáo khác để thay thế cho biểu đồ Burndown?

Bên cạnh sự cần thiết phải thích ứng yêu cầu này, chúng ta cũng nên đề cập đến tính hữu hiệu của việc sử dụng biểu đồ Burndown như là một cách để báo cáo tiến độ của nhóm dự án.

Chúng ta phải thừa nhận rằng với những quản trị dự án dày dạn kinh nghiệm, dù có PMP¹ hay không, dù các biểu đồ Burndown có thể có chất lượng tốt, nhưng bạn nghĩ biểu đồ Burndown thực sự thể hiện loại tiến độ nào cho quản trị dự án? Câu trả lời là chỉ một số nhà quản trị dự án (nhưng không nhiều) chấp nhận biểu đồ Burndown, do các Sprint tương đối ngắn (khoảng 30 ngày).

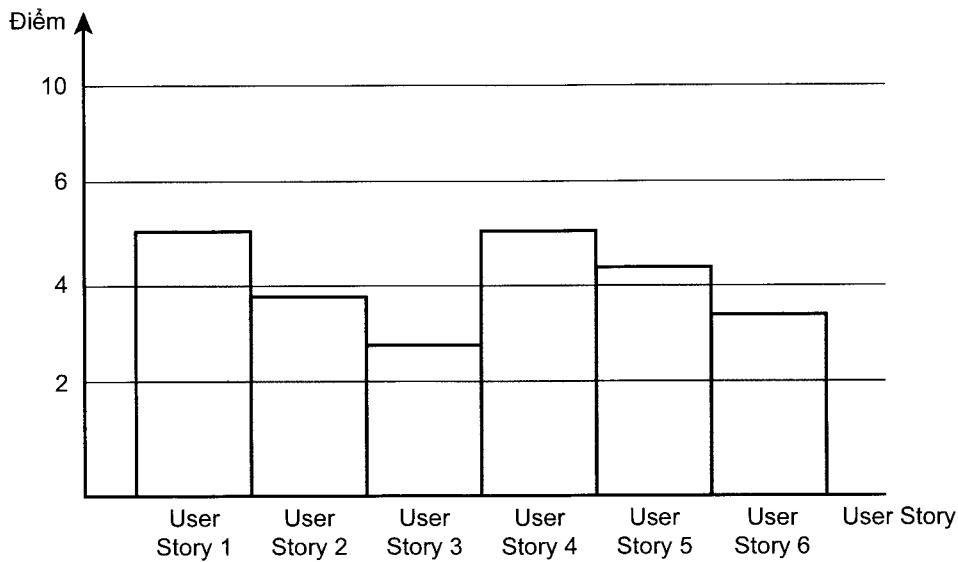
Khi suy ngẫm thêm, bạn có thể dễ dàng thấy rằng việc biết lượng thời gian (số giờ) để thực hiện công việc không thực sự cho bạn thấy chi tiết về những gì đã hoàn thành.

Số giờ còn lại có thể nghe thực tế hơn một chút so với phần trăm hoàn thành theo cách tính truyền thống nhưng thực sự là không hơn nhiều lắm.

Do Scrum dựa trên sự minh bạch, chúng tôi nghĩ rằng báo cáo tiến độ các User story là cách đo đạc tốt hơn, như trình bày trong Hình 12.1 và 12.2, tương ứng giữa cột Story và cập nhật tiến độ của nó trong biểu đồ.

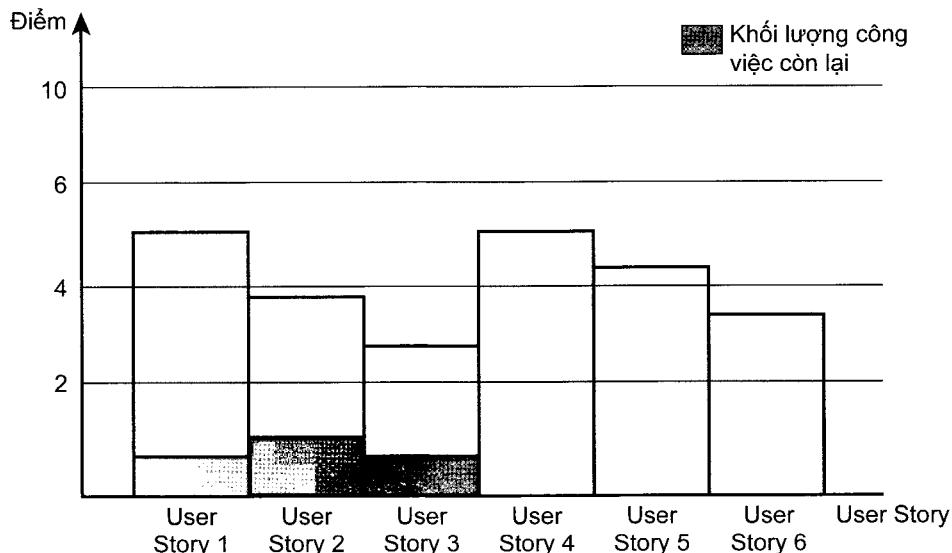
¹ PMP: Project Management Professional - Chứng chỉ Chuyên gia Quản trị Dự án

Chương 12 ▪ Làm thế nào để thích ứng với Scrum mà không phá bỏ tính linh hoạt cơ bản hoặc triển khai Scrumbut tiêu cực



Hình 12.1

Biểu đồ Burndown với các cột Story.



Hình 12.2

Cập nhật cột Story.

Phải làm gì nếu nhà quản lý muốn xem tổng số tiền đã chi cho dự án Scrum của bạn?

Chúng tôi không tìm thấy nhiều tài liệu phổ biến về Scrum để giải quyết câu hỏi chính đáng này và đây là lý do trước đó trong Chương 2, “Bàn luận về tài chính” đã thảo luận về vấn đề phân tích Giá trị thu được.

Nếu các chuyên gia Scrum không thể hiện đủ sự quan tâm về các vấn đề quản lý kinh doanh liên quan đến các phân tích tài chính dự án, chúng tôi lo ngại rằng Scrum sẽ không trở thành một phần của xu hướng quản lý dự án chính trong thời gian dài sắp tới.

Phải làm gì nếu tổ chức của bạn vẫn yêu cầu bổ nhiệm một người nào đó làm người quản lý dự án Scrum của bạn?

Trong trường hợp này, bạn nên cố gắng làm rõ với bộ phận quản lý rằng một trong những nguyên tắc của Scrum là nhóm tự tổ chức và tác động đó sẽ có trong các điều khoản về quản trị dự án.

Trong thực tế, trở thành người quản lý trong dự án Agile không có nghĩa là chỉ đạo công việc của mọi người mà là động viên và gỡ bỏ các trở ngại để bảo vệ tinh thần làm việc nhóm trong suốt Sprint. Điều đó liệu có đồng nghĩa rằng nhà quản lý dự án không thể đề xuất cho nhóm những gì họ cần làm để đạt được năng suất cao? Nhà quản lý dự án có thể làm điều này nhưng giàn tiếp.

Phải làm gì nếu không ai sẵn sàng hoặc có đủ kinh nghiệm để đảm nhận vai trò ScrumMaster?

Có hai kịch bản khả thi dưới đây:

Một là, nếu nhóm mới tiếp cận Scrum, bạn có thể gặp vấn đề trừ khi bạn tìm ra ai đó trong nhóm khá thành thạo Scrum và mời người đó đảm nhận vai trò này.

Hai là, nếu nhóm đã có một chút kinh nghiệm làm việc cùng nhau với Scrum, giải pháp là bạn chỉ có thể từ bỏ vai trò ScrumMaster. Nếu gặp phải trường hợp này, các bạn chỉ cần chia sẻ nhiệm vụ của ScrumMaster với nhau, như trong bài viết trên blog của Jurgen Appelo đã đề cập ở trên.

Phải làm gì nếu nhà quản lý khăng khăng rằng một trong những người quản lý chức năng sẽ đảm nhận vai trò ScrumMaster?

Chúng tôi đã gặp điều này trước đây dù bạn có tin hay không. Điều nguy hiểm ở đây là nhà quản lý vẫn sử dụng phong cách cũ với triết lý quản lý theo kiểu mệnh lệnh và kiểm soát mà không phải Scrum. Đơn giản bạn không thể là ScrumMaster nếu không chuyển từ kiểu mệnh lệnh và kiểm soát sang lãnh đạo kiểu phục vụ và điều đó càng khó khăn khi bạn không biết và không hiểu cẩn bản về lãnh đạo kiểu đầy tờ trong Scrum. Trong tất cả các tình huống mà chúng tôi đã gặp từ trước tới nay, đây có lẽ là tình huống có khả năng gây tổn hại nặng nhất.

Vì vậy, trước khi điều gì đó vượt ra ngoài tầm kiểm soát, bạn nên nói chuyện nhiều với nhà quản lý về những thay đổi trong trách nhiệm quản lý dự án với Scrum. Thay vì từ chối, bạn nên đồng ý để mang lại cho mọi người một cơ hội đảm đương một vai

Chương 12 • Làm thế nào để thích ứng với Scrum mà không phá bỏ tính linh hoạt cơ bản hoặc triển khai Scrumbut tiêu cực

trò mới, nhưng nếu bạn là huấn luyện viên của họ, sẽ có rất nhiều thứ về đào tạo và tổ chức mà bạn cần phải thực hiện. Nhưng đó là những gì mà một huấn luyện viên cần phải làm đúng không?

Phải làm gì nếu các thành viên trong nhóm không có khả năng gặp nhau trực tiếp hoặc tổ chức được các cuộc họp đúng hằng ngày?

Có thể sẽ không sao ngay cả khi bạn không thể họp mặt trực tiếp lần tổ chức được các buổi họp đúng hằng ngày. Trong tình huống này bạn phải sáng tạo để tìm ra cách thức nhằm giữ cho các thành viên trong nhóm bám sát tiến độ hướng tới mục tiêu Sprint. Sử dụng Skype hoặc/và task board điện tử sẽ hữu ích trong trường hợp này.

Nếu bạn thấy cần bằng chứng tin cậy để thuyết phục rằng bạn không thể tiến hành họp đúng hằng ngày, hãy kiểm một bản sao của bài viết “Scrum in Church” về Agile năm 2009, được viết bởi Arline Conan Sutherland và chồng của bà - Tiến sĩ Jeff Sutherland – một trong những tác giả Scrum. Qua việc đọc bài viết này, bạn sẽ nhận thấy rằng đây chính là những trở ngại mà vợ chồng Tiến sĩ Sutherland đã trải qua trong khi cố gắng triển khai họp Scrum Hằng ngày trong môi trường nhà thờ.

Phải làm gì nếu bộ phận quản lý của công ty giao cho bạn một nhóm lớn, nhiều hơn chín hoặc mười người, để thực hiện công việc?

Chừng nào người quản lý không yêu cầu bạn phải duy trì các nhóm lớn, hãy chia thành hai nhóm để có thể đạt được kích thước nhóm tối ưu nằm trong khoảng từ bảy tới chín thành viên. Tiếp theo, hãy phân chia công việc xoay quanh khái niệm thành phần dữ liệu chung như chúng tôi đã trình bày trong Chương 6, “Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm”. Điều này sẽ giúp bạn tổ chức làm việc nhóm trong dự án theo khái niệm nhóm tính năng, việc đó sẽ cho phép các nhóm nhỏ hoạt động với khả năng độc lập nhất có thể, trong khi việc tích hợp công việc của các nhóm trong một kiến trúc sản phẩm lớn lại suôn sẻ nhất có thể.

Khía cạnh Hạ tầng

Phải làm gì nếu hạ tầng kiểm thử cần thiết không được thiết lập để giúp nhóm tiến hành kiểm thử tự động, hồi quy hoặc tích hợp?

Điều này có thể là một vấn đề, bởi vì kiểm thử tự động và kiểm thử tích hợp liên tục rất quan trọng đối với sự thành công của Scrum. Một điều mà bạn có thể làm trong trường hợp này hoặc là (1) thử nói chuyện với phòng Đảm bảo Chất lượng (QA) để nhờ họ giúp đỡ sớm nhất có thể, hoặc là (2) thử tự mình triển khai môi trường kiểm thử bằng cách tải về an toàn một số phần mềm kiểm thử tự động và kiểm thử tích hợp liên tục nguồn mở, thậm chí trong không gian phát triển của riêng mình, như đã đề xuất ở Chương 9, “Tầm quan trọng của kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp.”

Phải làm gì nếu nhóm phát triển không muốn thực hành TDD (Test-Driven Development – Phát triển hướng kiểm thử)?

Chỉ trong trường hợp bạn không thể quen được với TDD, bạn nên biết rằng đây là một kỹ nghệ thực hành (engineering practice) bởi nó đòi hỏi các nhà phát triển phải viết các unit test case – trường hợp kiểm thử đơn vị (cố gắng để tạo ra kết quả thất bại) trước khi họ viết mã nguồn (cố gắng để đạt các kiểm thử đã viết).

Chúng ta biết rằng nhiều chuyên gia Scrum đã tiến tới việc ủng hộ TDD và điều đó thật tuyệt vời. Nhưng nếu nhóm của bạn không biết, hoặc không muốn thực hành TDD, có thể vẫn ổn (miễn là bạn có người dùng để thực hiện rất nhiều các kiểm thử chấp nhận).

Chúng tôi đã có cơ hội để làm việc với nhiều nhà phát triển tuyệt vời, họ không triển khai TDD nhưng vẫn chuyển giao mã nguồn chạy tốt ngay từ đầu.

Chúng tôi vẫn hết sức khuyến nghị bạn, nếu có thời gian và tiền bạc, bạn hãy để cho nhóm của mình học và thực hành TDD bởi vì nó đã được chứng minh là một kỹ nghệ thực hành rất hiệu quả.

Phải làm gì nếu nhóm kiểm thử không có bất cứ ai để phân công làm việc cùng với dự án Scrum mới của bạn?

Đây sẽ là một vấn đề khi mà kiểm thử liên tục và từ sớm là một phần chính yếu trong Agile. Vì vậy, điều đầu tiên phải làm là đưa một số BA (Business Analyst – Chuyên gia phân tích Nghiệp vụ) hoặc nhà phát triển đảm nhận vai trò này. Tốc độ nhóm có thể giảm một chút ở thời kỳ đầu do họ phải cố gắng làm quen với nhiệm vụ mới, nhưng mọi thứ sẽ tiến triển như bạn hành động sớm.

Khía cạnh Nhóm

Phải làm gì nếu hầu hết các thành viên nhóm đều biết Scrum và bạn không có sẵn một ScrumMaster để hướng dẫn họ?

Trừ khi bạn có một vài thành viên trong nhóm có kinh nghiệm với Scrum nếu không điều này sẽ là vấn đề. Nếu có thành viên có kinh nghiệm, hãy mời người đó làm ScrumMaster để thực hiện Scrum.

Nếu nhóm của bạn không có ai đủ hiểu biết về Scrum, đây có lẽ là thời điểm để yêu cầu một huấn luyện viên.

Phải làm gì nếu nhà quản lý nghĩ rằng nên để một người đảm nhận cả hai vai trò Product Owner và ScrumMaster do hạn chế về ngân sách?

Chúng tôi không nghĩ đây là lựa chọn tối ưu, nhưng với chúng tôi tình huống này đã diễn ra tốt đẹp, mặc dù có một số rủi ro. Thậm chí một số chuyên gia đã cho rằng việc kết hợp này là không được phép, nhưng bạn cứ thử làm như vậy nếu đây là cách duy nhất để bạn thử sức với Scrum. Nhưng hãy chuẩn bị biện pháp phòng ngừa nhằm đảm bảo người đó cam kết làm bất cứ điều gì để đảm nhận tốt vai trò tổng hợp mới này. Những cam kết cần thiết là:

Chương 12 ▪ Làm thế nào để thích ứng với Scrum mà không phá bỏ tính linh hoạt cơ bản hoặc triển khai ScrumBut tiêu cực

1. Cam kết để nhóm biết rõ khi nào anh ta đóng vai là ScrumMaster và khi nào là Product Owner.
2. Không bao giờ được tận dụng lợi thế có thể chuyển qua chuyển lại giữa hai vai trò nhằm đem lại lợi ích cho bản thân.

Nhìn chung, đây có lẽ là một trong những tình huống “ScrumBut” rủi ro nhất, nhưng nếu đó là lựa chọn duy nhất để triển khai Scrum thì cứ thử đi. Trong khi tìm kiếm cơ hội đầu tiên, bắt cứ khi nào có thể hãy tách biệt hai vai trò này.

Khía cạnh Công nghệ

Phải làm gì nếu nhóm phát triển mới tiếp cận một số công nghệ hoặc hạ tầng hỗ trợ (ví dụ như các công cụ kiểm thử tự động và kiểm thử tích hợp liên tục), mà không được đào tạo về chúng?

Trong khả năng chúng tôi có thể chia sẻ, không sớm thì muộn đây sẽ là một vấn đề khiến nhóm gặp khó khăn bởi vì họ thiếu hiểu biết về những công cụ hỗ trợ trong công việc. Vì vậy, thậm chí nếu bạn đang gấp rút thử nghiệm Scrum, hãy chắc chắn đã tính tới một khoản tiền đào tạo trong ngân sách của mình để đảm bảo các thành viên được đào tạo đúng cách trước khi triển khai các công cụ đó.

Khía cạnh Quy trình

Phải làm gì nếu các nhóm không định nghĩa được thế nào là hoàn thành?

Thậm chí khi một nhóm đã tự làm việc tốt, một điều rất quan trọng đó là nhóm cần định nghĩa những gì được coi là *hoàn thành*, vì có nhiều hoạt động sẽ phụ thuộc vào điều này. Nhưng khi bạn có nhiều hơn hai nhóm Scrum làm việc cùng nhau và không có nhóm nào muốn định nghĩa hoàn thành, thì bạn đang đối đầu với những vấn đề. Bạn phải định nghĩa hoàn thành dù thích hay không thích điều này.

Phải làm gì nếu nhà quản lý đề nghị tổ chức một số giai đoạn chuyên biệt trong Scrum để thực hiện những hoạt động mà nhóm sẽ không có thời gian để quan tâm?

Một trong những nguyên tắc của Scrum đó là mọi thứ nên hoàn thành trong Sprint. Do đó bạn không bao giờ thực sự cần có những giai đoạn Scrum chuyên biệt, ví dụ như phân đoạn kiểm thử hay giai đoạn xác định yêu cầu.

Bạn nên tránh những giai đoạn chuyên biệt nhiều nhất có thể, bởi vì chúng sẽ đưa bạn trở lại với mô hình thác nước trước khi bạn kịp nhận ra điều này.

Phải làm gì nếu một số người đề xuất rằng một Sprint nên kéo dài hơn 30 ngày?

Bạn phải thận trọng khi đối phó với vấn đề này ngay cả khi bạn bị chậm trong việc quyết định xem liệu độ dài của Sprint có lớn hơn 30 ngày hay không.

Chúng tôi đã từng nghe thấy những Sprint 6 tuần, nhưng chẳng có lý do gì để không duy trì Sprint 4 tuần nếu bạn triển khai Scrum. Chúng tôi khuyến nghị rằng bạn cần giữ Sprint 4 tuần và giảm số thời điểm chuyển giao, nhưng không nên mở rộng độ dài

Sprint chỉ vì bạn nghĩ sản phẩm của mình phức tạp hơn và do đó nhóm sẽ cần nhiều thời gian để triển khai các thiết kế hơn.

Phải làm gì nếu công ty nói rằng họ vẫn muốn thu thập nhiều yêu cầu nhất có thể tại thời điểm bắt đầu dự án?

Dù bạn có tin hay không thì đây cũng chính là vấn đề chúng tôi thường gặp trong nhiều công ty. Đặc biệt là ở những công ty không sẵn sàng triển khai Scrum. Nhưng việc này sẽ tiêu diệt nhiều mục đích của Agile và Scrum. Cả Agile và Scrum đều nói về các tương tác hằng ngày (hoặc ít nhất là thường xuyên) với những người dùng cuối nhằm giúp đảm bảo các yêu cầu của họ được hiểu rõ ràng. Vì vậy, chúng ta không thu thập nhiều yêu cầu nhất có thể tại thời điểm bắt đầu dự án mà bắt đầu với cơ sở đúng đó là chỉ thu thập các yêu cầu ở mức khái quát để lập kế hoạch phát hành và sau đó thu thập thêm các yêu cầu bằng cách duy trì các cuộc thảo luận với người dùng nghiệp vụ trong suốt quá trình triển khai dự án.

Phải làm gì nếu nhà quản lý vẫn cứ nhất quyết buộc bạn sử dụng Scrum cùng với Vòng đời Phát triển Phần mềm (SDLC) hiện nay của công ty?

Nếu SDLC của công ty sử dụng vòng đời thác nước thì bạn sẽ không thể triển khai được bởi vì Scrum dựa trên một khung làm việc tăng trưởng. Vì vậy, hãy quay lại giải thích với quản lý rằng Agile khác mô hình thác nước, đây là một quy trình tiếp cận theo hướng lặp và tăng trưởng.

Phải làm gì nếu QA hoặc nhóm hạ tầng nói với bạn rằng họ không có nền tảng tốt để triển khai việc kiểm thử tích hợp liên tục hoặc build hằng ngày?

Đây chắc chắn là tình huống nguy hiểm vì đó là nền tảng căn bản của Agile và Scrum. Không khởi động dự án Scrum cho tới khi một số thứ như vậy đã bắt đầu được thiết lập và bạn sẽ không hối tiếc vì điều này.

Khía cạnh Nghiệp vụ

Phải làm gì nếu cấp quản lý kinh doanh không có ai làm Product Owner?

Trong trường hợp này, hãy hỏi ai đó đảm nhận vai trò này nếu bạn có thể, ví dụ như nhà quản lý kinh doanh hoặc ít nhất là những người hiểu biết tốt và có khả năng phân tích nghiệp vụ. Người này sẽ có đủ quyền lực và sự hỗ trợ từ cấp quản lý kinh doanh để đưa ra mọi quyết định.

Chúng ta không thể bao quát được toàn bộ tình huống có thể gặp phải trong thực tế ở chương này. Mục đích ở đây chỉ là cung cấp cho bạn những ý tưởng về cách bạn có thể thích ứng Scrum với những tình huống trong thực tế trong khi vẫn đảm bảo sự thích ứng này phù hợp với tinh thần của Agile và Scrum.

TÓM TẮT

Một thời điểm nào đó trong sự nghiệp phát triển phần mềm, tất cả chúng ta đều mong ước rằng mình có thể khám phá một số quy trình hoặc phương pháp luận áp dụng cho mọi dự án và đảm bảo rằng mọi thứ sẽ hoạt động tốt nhất.

Tiếc rằng, công việc và những con người mà chúng ta gặp trong thời gian dài đó không đơn giản như vậy, nên không một quy trình hay phương pháp luận nào đã từng tồn tại hoặc sẽ xuất hiện có thể đáp ứng được mong muốn của chúng ta. Sự thực, Scrum cũng như vậy mà thôi.

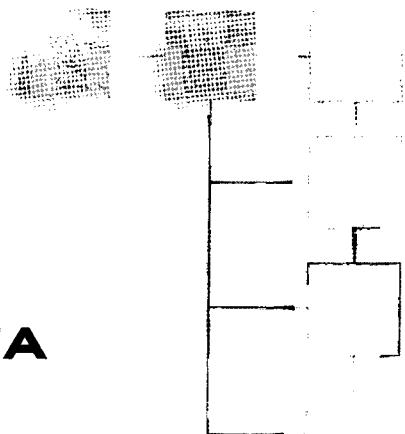
Không nản lòng với những gì mà một số chuyên gia gọi là “ScrumBut” tiêu cực, chúng ta đã xem xét trong chương này một số ví dụ về những tình huống mà bạn có thể thích ứng Scrum mà vẫn giữ được tinh thần Agile trong đó. Đây là lý do chúng tôi bắt đầu cuốn sách này với những thảo luận về Tuyên ngôn Agile trong Chương 1.

Một số người khác, như Giáo sĩ Arline Conan Sutherland, vợ của Tiến sĩ Jeff Sutherland và Jurgen Appelo, đã viết về những kinh nghiệm của họ trong việc thích ứng Scrum, những thích ứng được Jurgen xem là điều tốt nhất trong việc cải tiến Scrum.

Theo sau những ví dụ của họ, chúng tôi cũng liên hệ tới một số kinh nghiệm của mình về “ScrumBut” tích cực để bạn có thể lấy cảm hứng từ đó. Chúng tôi chắc chắn rằng bạn sẽ có những thích ứng riêng và viết lại trong tương lai không xa.

CHƯƠNG 13

TỰ ĐÁNH GIÁ ĐỘ SẴN SÀNG CỦA DỰ ÁN SCRUM



Sẽ là rất tuyệt vời nếu biết được cơ hội thành công trước khi bắt đầu “chuyến hành trình Scrum”, bạn sẽ biết được mình phải đặt trọng tâm vào đâu để cải thiện lợi thế của bản thân. Trong chương này, bạn sẽ học một phương pháp tự đánh giá đơn giản cho dự án để dự đoán những cơ hội thành công trước khi bắt đầu.

MỘT CÔNG CỤ ĐƠN GIẢN ĐỂ ĐÁNH GIÁ ĐỘ SẴN SÀNG VỚI SCRUM

Trong Hình 13.1 bạn có thể thấy rằng bản đánh giá này dựa trên việc tính điểm theo sáu khía cạnh đã được giới thiệu trong suốt cuốn sách.

1. Khía cạnh Tổ chức

Khía cạnh này chủ yếu là để đánh giá xem liệu những phòng ban và các nhóm khác nhau đã quen thuộc với các giá trị và kỹ thuật của Scrum hay chưa. Họ càng thành thạo thì sẽ càng tốt hơn cho bạn sau này.

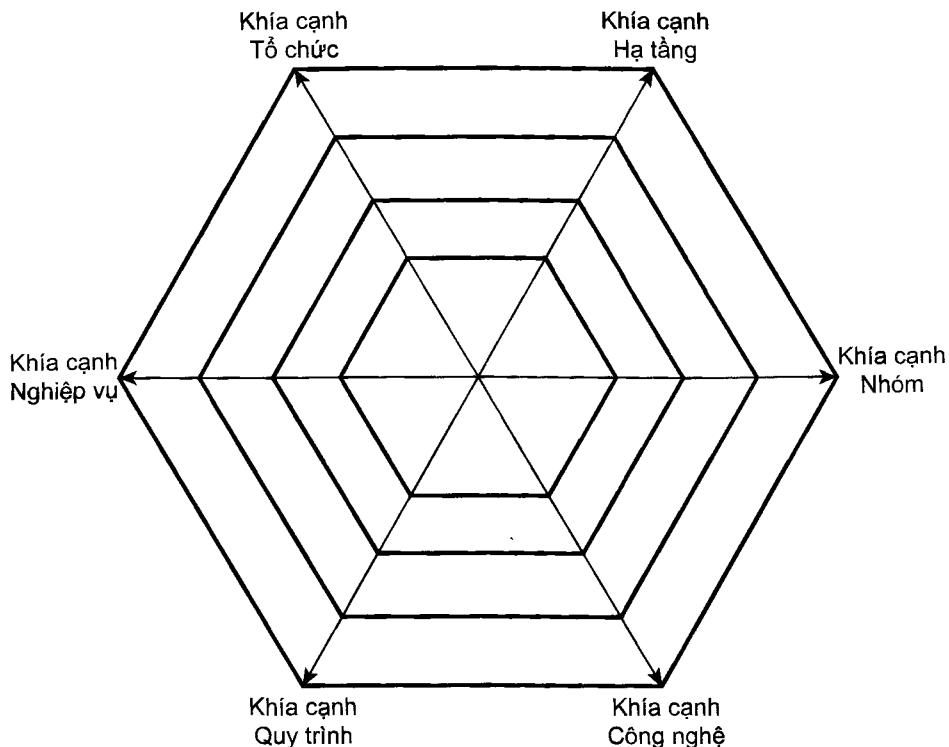
2. Khía cạnh Hạ tầng

Khía cạnh này chủ yếu là để đánh giá xem liệu hạ tầng kiểm thử đã sẵn sàng để cho phép nhóm triển khai tất cả những kiểm thử cần thiết hay chưa. Điều này đã được đề cập trong Chương 6, “Ảnh hưởng của tầm nhìn kiến trúc tới tốc độ nhóm và chất lượng phần mềm”.

3. Khía cạnh Nhóm

Khía cạnh này chủ yếu để đánh giá mức độ quan hệ giữa các thành viên trong nhóm dự án - một nhân tố mấu chốt để tinh thần làm việc nhóm tốt.

Chương 13 ▪ Tự đánh giá Độ sẵn sàng của Dự án Scrum



Hình 13.1

Ma trận tự đánh giá độ sẵn sàng với Scrum.

4. Khía cạnh Công nghệ

Khía cạnh này chủ yếu để đánh giá xem liệu nhóm đã am hiểu về công nghệ sẽ được sử dụng hay chưa. Mặc dù chính bản thân khía cạnh này không phải là vấn đề cho mỗi dự án Scrum, nhưng nó lại giúp bạn biết được tình trạng của nhóm để đưa vào bản ước tính của mình.

5. Khía cạnh Quy trình

Khía cạnh này chủ yếu để đánh giá xem liệu công ty của bạn đã sẵn có những kiến thức tốt và kinh nghiệm thực tế với Scrum hay chưa.

6. Khía cạnh Nghiệp vụ

Khía cạnh này chủ yếu để đánh giá xem liệu đối tác kinh doanh của bạn đã am hiểu hoặc quen thuộc với những yêu cầu và kỹ thuật của Scrum hay chưa. Đúng như những gì bạn nghĩ, họ càng am hiểu thì sẽ càng tốt cho bạn. Lợi ích việc họ biết Scrum sẽ giúp bạn dễ dàng có được một Product Owner được trao quyền, toàn tâm toàn ý tham gia và rất am hiểu.

Một công cụ đơn giản để đánh giá độ sẵn sàng với Scrum

Tùy theo các câu trả lời và điểm số mà bạn thu được cho những khía cạnh này bạn sẽ biết được chúng sẽ hỗ trợ hay khiêm cho việc triển khai Scrum trở nên khó khăn hơn.

Bản khảo sát hoạt động theo cách sau, câu trả lời cho từng câu hỏi càng tốt hoặc tích cực thì điểm số của bạn càng cao; điểm tối đa (tốt nhất) là +2 còn điểm tối thiểu (tệ nhất) là 0.

Khi cộng tất cả các câu trả lời lại bạn sẽ có tổng điểm nằm trong khoảng từ 0 đến 36.

Nếu bạn có tổng điểm bằng 0 (tối thiểu), điều đó có nghĩa là môi trường dự án sẽ khiến bạn trải qua một thời gian khó khăn trong việc chuyển giao các sản phẩm cam kết của dự án.

Nếu bạn có tổng điểm số bằng 36 (tối đa) thì điều đó có nghĩa là môi trường dự án sẽ mang lại cho bạn cơ hội thành công tối đa trong việc chuyển giao các sản phẩm cam kết của dự án.

Nếu bạn có trên 18 điểm (+18 là điểm trung bình) thì bạn đang ở trên mức trung bình và có thể rất thành công với Scrum, nhưng điểm số đó cũng cho bạn thấy một số vấn đề cần giải quyết nhằm cải thiện khả năng chuyển giao của nhóm. Đó có thể là việc giúp các thành viên nhóm làm việc với nhau tốt hơn nếu họ chỉ mới làm quen với nhau hoặc đã gặp phải một số vấn đề không tốt trong dự án trước đó.

Nếu bạn có tổng điểm dưới 18 thì có nghĩa là cơ hội thành công của bạn đang ở thấp hơn mức trung bình. Bạn vẫn có thể thành công, nếu cố gắng hết sức để cải thiện các vấn đề trong môi trường xung quanh dự án nhằm nâng cao khả năng chuyển giao của nhóm.

Hãy in bản khảo sát trong Hình 13.2 - 13.7 và hoàn thành chúng tốt nhất trong khả năng có thể của bạn, bạn sẽ sớm ở trên con đường trở thành nhà thực hành Scrum.

Khía cạnh Tổ chức

Yếu tố	Khoảng giá trị (0/2)
1. Đã có những phòng ban khác nhau cùng làm việc thành công trong một dự án Scrum?	
2. Có sự chống đối mạnh mẽ với Scrum trong tổ chức?	
3. Có tồn tại sự hỗ trợ lớn về Scrum giữa những phòng ban khác nhau trong công ty?	

Hình 13.2

Khía cạnh Tổ chức.

Khía cạnh Hạ tầng

Yếu tố	Khoảng giá trị (0/2)
1. Kiểm thử tự động đã được áp dụng và trở thành một kỹ thuật phổ biến hay chưa?	
2. Kiểm thử tích hợp liên tục đã được áp dụng và trở thành một kỹ thuật phổ biến hay chưa?	
3. Môi trường build (build environment) hằng ngày đã được áp dụng và trở thành một kỹ thuật phổ biến hay chưa?	

Hình 13.3

Khía cạnh Hạ tầng.

Khía cạnh Nhóm

Yếu tố	Khoảng giá trị (0/2)
1. Scrum là hoàn toàn mới đối với nhóm?	
2. Có phải trước đó các thành viên trong nhóm đã từng làm việc thành công với nhau?	
3. Các thành viên trong nhóm hiểu và tôn trọng lẫn nhau?	

Hình 13.4

Khía cạnh Nhóm.

Khía cạnh Công nghệ

Yếu tố	Khoảng giá trị (0/2)
1. Nhóm phát triển có nhiều kinh nghiệm với ngôn ngữ lập trình?	
2. Các thành viên trong nhóm phát triển có nhiều kinh nghiệm với công nghệ được sử dụng?	
3. Môi trường chạy ứng dụng thực tế với Scrum đã sẵn sàng?	

Hình 13.5

Khía cạnh Công nghệ.

Khía cạnh Quy trình

Yếu tố	Khoảng giá trị (0/2)
1. Scrum có phải là khung quy trình được chấp thuận trong công ty hay không?	
2. Trong công ty có sự hỗ trợ tốt cho Scrum hay không?	
3. Trong công ty có sự phản đối đáng kể nào đối với Scrum hay không?	

Hình 13.6

Khía cạnh Quy trình.

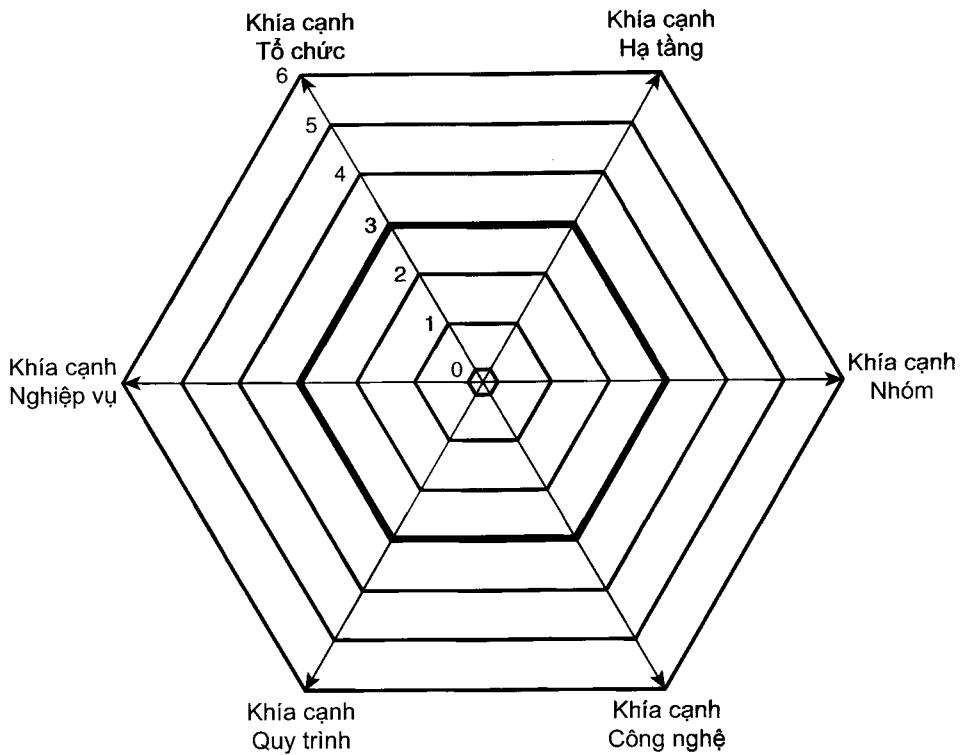
Khía cạnh Nghiệp vụ

Yếu tố	Khoảng giá trị (0/2)
1. Có một Product Owner nào hoàn toàn sẵn sàng và gắn bó với nhóm hay không?	
2. Có phải Product Owner đã quen với Scrum nhưng vẫn thiếu kinh nghiệm thực tế hay không?	
3. Product Owner đã từng thành công với Scrum trước đây hay chưa?	

Hình 13.7

Khía cạnh Nghiệp vụ.

Chương 13 • Tự đánh giá Độ sẵn sàng của Dự án Scrum



Hình 13.8

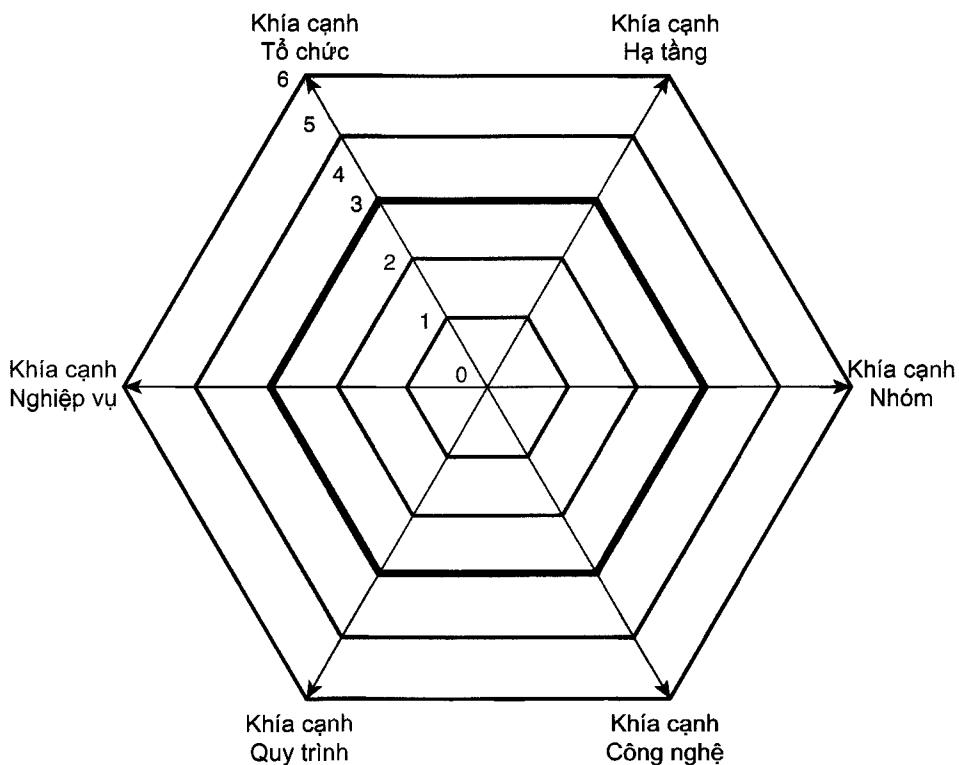
Điểm tối thiểu trong ma trận tự đánh giá độ sẵn sàng của Scrum.

Đến đây thì bạn đã kết thúc bản khảo sát.

Tùy vào toàn bộ câu trả lời, bạn có thể đạt tối thiểu là 0 điểm, được biểu diễn bằng một chấm nhỏ ở chính giữa Hình 13.8.

Tuy nhiên, nếu may mắn, bạn có thể đạt tối đa là 36 điểm, được biểu diễn bằng vòng tròn ngoài cùng bao toàn bộ Hình 13.9. Trong hình này vòng tròn thứ hai biểu diễn giá trị trung bình.

Nếu bạn giống với một vài nhóm và một số công ty khác, bạn có thể đạt được tổng điểm dao động xung quanh 18 điểm, với một vài điểm cao hoặc thấp hơn ở những khía cạnh khác nhau. Hình 13.10 minh họa cho trường hợp này.

**Hình 13.9**

Điểm tối đa (36) trong ma trận tự đánh giá độ sẵn sàng của Scrum.

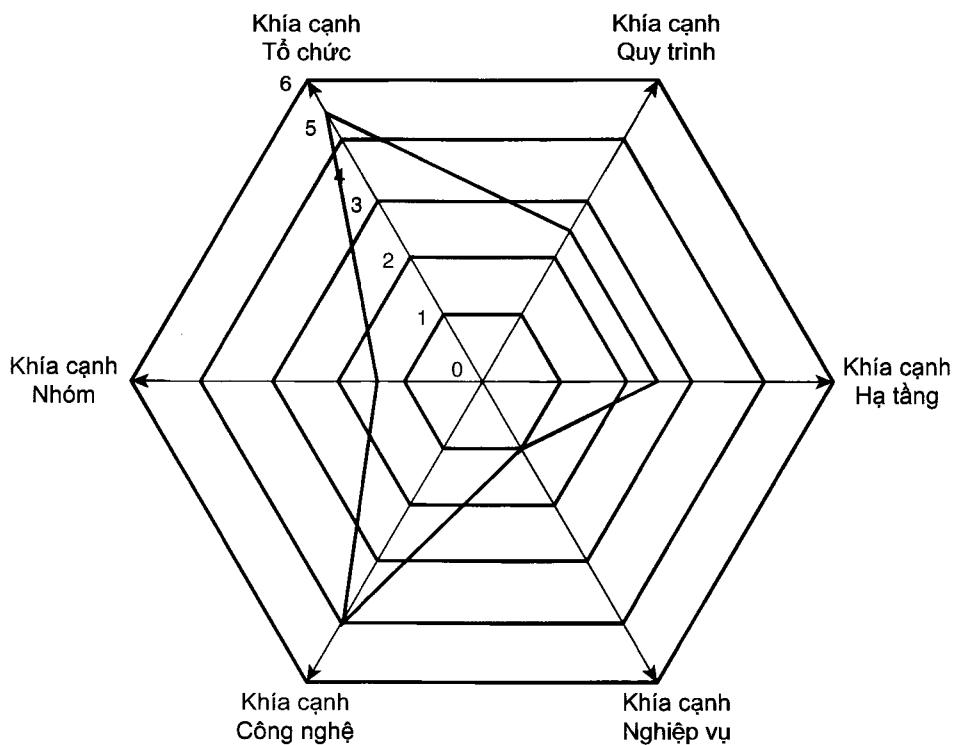
Nếu gặp trường hợp này thì bạn phải có trách nhiệm nâng điểm số thấp lên cao hơn khi bắt đầu (hoặc trước khi bắt đầu) nhằm cải thiện cơ hội thành công với Scrum của nhóm dự án.

VÍ DỤ

Để minh họa việc đánh giá này, hãy cùng xem xét lần lượt sáu khía cạnh và tính điểm cho dự án đầu tiên chúng tôi sử dụng Scrum cho doanh nghiệp cách đây vài năm (từ Hình 13.11 đến Hình 13.15).

Hãy nhìn vào số điểm cho mỗi khía cạnh trong Hình 13.16. Bạn sẽ thấy rằng trong khi đã làm khá tốt với các khía cạnh tổ chức, nghiệp vụ, công nghệ, nhóm và quy trình, thì chúng tôi cần phải nỗ lực để cải thiện khía cạnh hạ tầng (kiểm thử), khía cạnh chỉ đạt được điểm số là 1.

Chương 13 • Tự đánh giá Độ sẵn sàng của Dự án Scrum



Hình 13.10

Điểm số nằm xung quanh giá trị trung bình (18) trên ma trận tự đánh giá về Scrum.

Yếu tố	Khoảng giá trị (0/2)
1. Đã có những phòng ban khác nhau cùng làm việc thành công trong một dự án Scrum?	1
2. Có sự chống đối mạnh mẽ với Scrum trong tổ chức?	1
3. Có tồn tại sự hỗ trợ lớn về Scrum giữa những phòng ban khác nhau trong công ty?	1

Hình 13.11

Khía cạnh Tổ chức.

Yếu tố	Khoảng giá trị (0/2)
1. Kiểm thử tự động đã được áp dụng và trở thành một kỹ thuật phổ biến hay chưa?	0
2. Kiểm thử tích hợp liên tục đã được áp dụng và trở thành một kỹ thuật phổ biến hay chưa?	0
3. Môi trường build (build environment) hàng ngày đã được áp dụng và trở thành một kỹ thuật phổ biến hay chưa?	1

Hình 13.12

Khía cạnh Hạ tầng Phát triển.

Yếu tố	Khoảng giá trị (0/2)
1. Scrum là hoàn toàn mới đối với nhóm?	1
2. Có phải trước đó các thành viên trong nhóm đã từng làm việc thành công với nhau?	1
3. Các thành viên trong nhóm hiểu và tôn trọng lẫn nhau?	2

Hình 13.13

Khía cạnh Nhóm.

Cộng điểm cho sáu khía cạnh về môi trường để xác định tổng giá trị thu được sau quá trình tự đánh giá. Trong trường hợp này, tổng điểm là 20 ($3 + 1 + 4 + 4 + 3 + 5$). Điều này có nghĩa đối với chúng tôi (1) môi trường không làm cho công việc của nhóm dự án khó khăn hoặc dễ dàng và (2) do đó, hệ số nhân chúng tôi nên sử dụng để ước tính Story là 1.

Yếu tố	Khoảng giá trị (0/2)
1. Nhóm phát triển có nhiều kinh nghiệm với ngôn ngữ lập trình?	2
2. Các thành viên trong nhóm phát triển có nhiều kinh nghiệm với công nghệ được sử dụng?	2
3. Môi trường chạy ứng dụng thực tế với Scrum đã sẵn sàng?	0

Hình 13.14

Khía cạnh Công nghệ.

Chương 13 • Tự đánh giá Độ sẵn sàng của Dự án Scrum

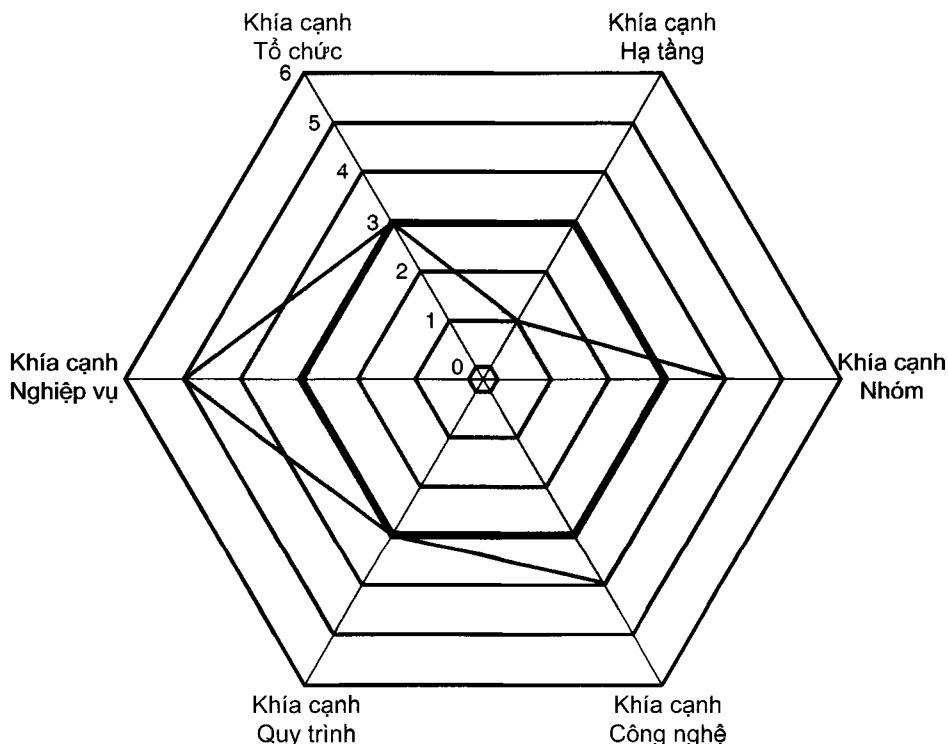
Yếu tố	Khoảng giá trị (0/2)
1. Scrum có phải là khung quy trình được chấp thuận trong công ty hay không?	1
2. Trong công ty có sự hỗ trợ tốt cho Scrum hay không?	1
3. Trong công ty có sự phản đối đáng kể nào đối với Scrum hay không?	1

Hình 13.15

Khía cạnh Quy trình.

Với những kết quả này và việc tiến hành cuộc họp lập kế hoạch Sprint đầu tiên, chúng tôi có thể tính toán điểm story và thu được bảng kết quả như trong Hình 13.17 cho ba story đầu tiên, với UP (Unadjusted Point - Điểm chưa Hiệu chỉnh) của chúng lần lượt là 7, 7 và 8.

Sau những lỗ lực của chúng tôi trong việc cải tiến hạ tầng kiểm thử, điểm số cho khía cạnh kiểm thử đã được cải thiện và tổng điểm tự đánh giá là 25 như bạn thấy trong Hình 13.18.



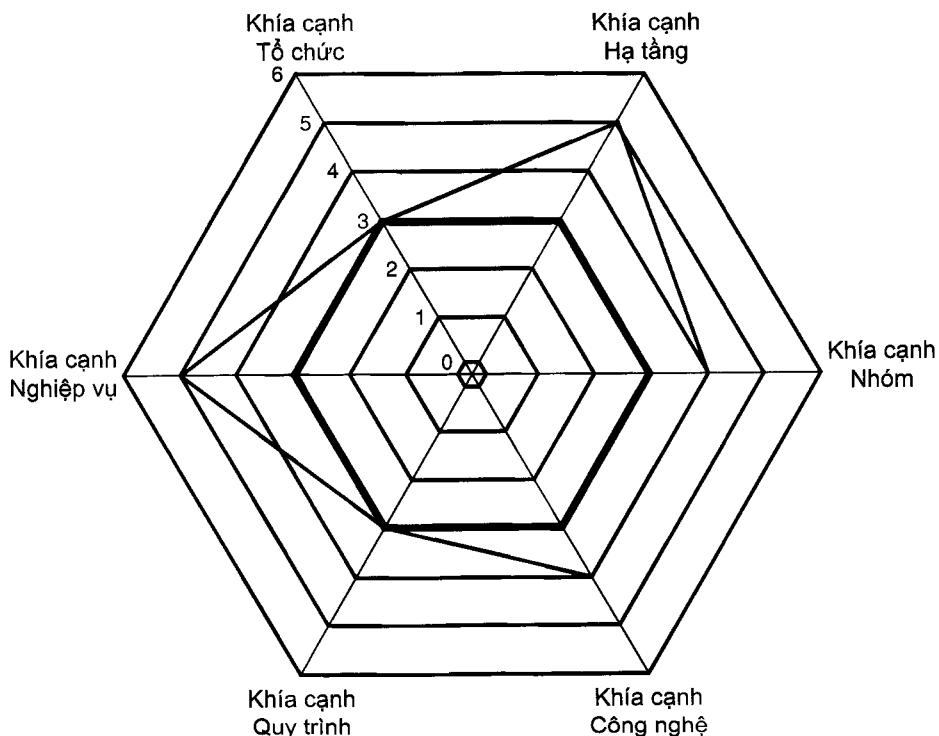
Hình 13.16

Điểm số của một công ty mới khởi nghiệp khi họ bắt đầu dự án Scrum đầu tiên.

	Đặc điểm				Tổng UP (Điểm chưa Hiệu chỉnh)	Hệ số nhân	AP (Điểm Hiệu chỉnh)	ED (Yếu tố Môi trường)	PPS (=AP x ED)/36)
Hạng mục Product Backlog (Story)	Loại tương tác	Quy tắc Nghiệp vụ	Thực thể	Loại Thao tác dữ liệu					
Sprint 1									
Đăng ký	3	1	1	2	7	1	7	20	4
Đăng nhập	3	1	1	2	7	1	7	20	4
Đăng xuất	3	1	1	1	6	1	6	20	3
Thêm phòng	3	1	1	2	7	1	7	20	4
Tổng									15

Hình 13.17

Ma trận ước tính cho Sprint 1. Chú thích: Hãy nhớ rằng AP (Điểm Hiệu chỉnh) = UP (Điểm chưa Hiệu chỉnh) x C (Hệ số nhân) và PPS (Điểm/Story) = (AP x ED)/36.

**Hình 13.18**

Điểm số sau khi hạ tầng kiểm thử đã được cải thiện.

Chương 13 • Tự đánh giá Độ sẵn sàng của Dự án Scrum

Với kết quả này, chúng tôi biết rằng mình có thể cải thiện hệ số nhân và do đó hệ số sẽ sử dụng là 0.5 (thay vì 1 cho Sprint 1), điều này mang lại bảng kết quả như trong Hình 13.19.

DIỄN ĐẠT LẠI CÁC ĐÁNH GIÁ

Kết quả đánh giá này sẽ giúp bạn hiểu rằng một khi biết một khía cạnh nào đó đang ở đâu, bạn có thể cải thiện điểm số cho khía cạnh đó tới khi đạt được giá trị nhất định (chẳng hạn là 4) điều này mang lại cho bạn nhiều hy vọng để thành công hơn với Scrum.

Hạng mục Product Backlog (Story)	Đặc điểm				Tổng UP (Điểm chưa Hiệu chỉnh)	Hệ số nhân	AP (Điểm Hiệu chỉnh)	ED (Yếu tố Môi trường)	PPS (=AP x ED)/36)
	Loại tương tác	Quy tắc Nghiệp vụ	Thực thi	Loại Thao tác dữ liệu					
Sprint 2									
Chỉnh sửa Hồ sơ	3	1	1	3	8	0.5	4	24	3
Hủy tài khoản	3	1	1	1	6	0.5	3	24	2
Xóa phòng	3	1	1	1	6	0.5	3	24	2
Duyệt phòng	3	1	1	1	6	0.5	3	24	2
Duyệt các bản ghi	3	1	1	1	6	0.5	3	24	2
Tổng									11

Hình 13.19

Mã trận ước tính cho Sprint 2 sau khi hạ tầng kiểm thử được cải thiện. Chú thích: Hãy nhớ rằng AP (Điểm Hiệu chỉnh) = UP (Điểm chưa Hiệu chỉnh) x C (Hệ số nhân) và PPS (Điểm/Story) = (AP x ED)/36).

Nói chung, mục tiêu của bạn sẽ bám sát và gần những tiêu chí sau đây hơn:

1. Thường xuyên chuyển giao phần mềm.
2. Cộng tác thường xuyên giữa nhóm làm phần mềm và nghiệp vụ.
3. Đóng khung thời gian mọi hoạt động hoặc mọi cuộc họp để tránh kéo dài quá thời gian.
4. Thực hiện thường xuyên chu trình thanh tra và thích nghi
5. Nhóm tự-quản và được trao quyền
6. Và mọi hoạt động duy trì ở tốc độ ổn định (không được hối thúc nhóm)

TÓM TẮT

Một trong những phương pháp để bạn biết cơ hội thành công của dự án với Scrum đó là tự tiến hành đánh giá một cách trung thực về vị trí của mình bằng cách sử dụng ma trận tự đánh giá độ sẵn sàng của dự án Scrum. Trong chương này chúng tôi cung cấp thông tin để bạn dễ dàng sử dụng và hiểu được sáu khía cạnh của ma trận này.

1. Khía cạnh Tổ chức
2. Khía cạnh Hạ tầng
3. Khía cạnh Nhóm
4. Khía cạnh Công nghệ
5. Khía cạnh Quy trình
6. Khía cạnh Nghiệp vụ

Câu trả lời và điểm số của các khía cạnh sẽ hỗ trợ hoặc cản trở việc triển khai Scrum.

Nếu bạn có tổng điểm là 36 (mức tối đa), điều này có nghĩa rằng môi trường dự án giúp bạn có cơ hội tối đa để đạt được sự thành công.

Nếu bạn có điểm tổng lớn hơn 18 (điểm trung bình là 18), điều này có nghĩa rằng cơ hội thành công của bạn trên mức trung bình và mặc dù có một số vấn đề thì bạn vẫn có thể rất thành công với Scrum.

Nếu tổng điểm nhỏ hơn 18, điều đó có nghĩa rằng cơ hội thành công của bạn dưới mức trung bình.

Nếu bạn có tổng điểm là 0 (mức thấp nhất), nghĩa là với môi trường triển khai dự án này bạn sẽ có quãng thời gian khó khăn để chuyển giao những sản phẩm đã cam kết.

Một ví dụ minh họa đã được trình bày trong chương về điểm số của 6 khía cạnh môi trường ở một công ty mới khởi nghiệp khi họ khởi động dự án Scrum đầu tiên. Công ty này đạt tổng điểm là 20.

Chúng tôi cũng cung cấp một số gợi ý trong chương này về cách thức để bạn cải thiện cơ hội thành công của mình.

CHƯƠNG 14

KHI NÀO BẠN CẦN MỘT SCRUMMASTER?

Cho tới thời điểm này chúng ta chưa thảo luận sâu về ScrumMaster, bởi vì về phương diện nào đó, cuốn sách này sẽ trợ giúp như là một ScrumMaster của bạn. Tuy nhiên, với mục đích mô tả đầy đủ vai trò của ScrumMaster, chúng tôi đưa vào trong chương những phẩm chất mà bạn cần tìm kiếm cho vai trò này.

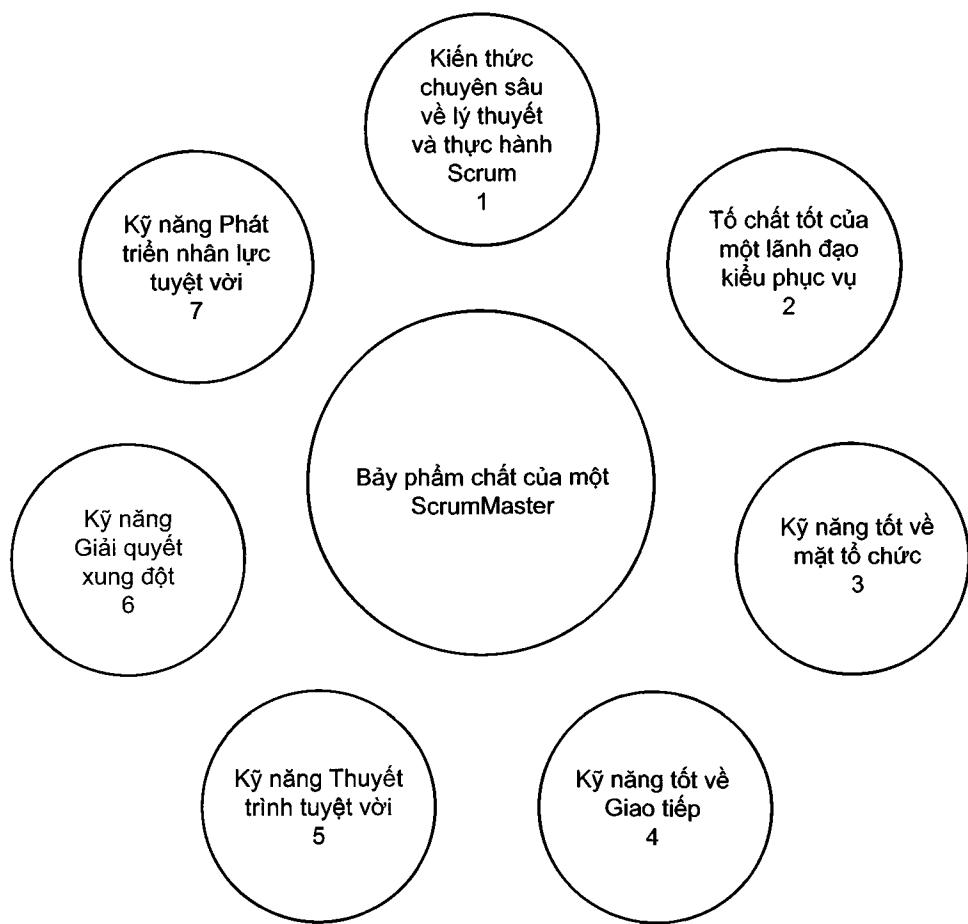
Kinh nghiệm đã dạy cho chúng tôi rằng một ScrumMaster cần trước hết và quan trọng nhất là 7 phẩm chất và kỹ năng như trình bày trong Hình 14.1.

Bảy phẩm chất mà một ScrumMaster cần có là:

1. Có kiến thức chuyên sâu về cả lý thuyết lẫn thực hành Scrum
2. Có tố chất tốt của một lãnh đạo kiểu phục vụ
3. Có kỹ năng tốt về mặt tổ chức
4. Có kỹ năng tốt về giao tiếp
5. Có kỹ năng thuyết trình tuyệt vời
6. Có kỹ năng giải quyết xung đột
7. Có kỹ năng phát triển nhân lực tuyệt vời

Hãy lần lượt xem xét từng phẩm chất này:

Chương 14 - Khi nào bạn cần một ScrumMaster?



Hình 14.1

Bảy phẩm chất của một ScrumMaster.

KIẾN THỨC CHUYÊN SÂU VỀ LÝ THUYẾT VÀ THỰC HÀNH SCRUM

Trong số bảy phẩm chất, một điều khá dễ hiểu là kiến thức về Scrum được coi là phẩm chất đầu tiên mà Scrum Master cần có.

Ngoài kiến thức lý thuyết về Scrum, thậm chí ngay cả khi có chứng chỉ xác nhận, việc có kiến thức thực hành Scrum thu được từ trải nghiệm là tốt nhất cho vai trò Scrum Master.

Như bạn đã biết, mặc dù Scrum có vẻ đơn giản về mặt lý thuyết, nhưng việc triển khai trong thực tế lại khá thách thức, đặc biệt nếu công ty nơi mà bạn đang làm việc không tổ chức lại hoạt động theo hướng Scrum. Đây là lý do tại sao ngay từ đầu cuốn sách đã đưa ra cách thức tiếp cận các nhóm khác nhau để có được sự đồng lòng và cộng tác của họ.

TỐ CHẤT LÃNH ĐẠO KIỂU PHỤC VỤ TUYỆT VỜI

Do Scrum là một phần của phong trào Agile và dựa trên ý tưởng một nhóm sẽ đạt hiệu quả hơn nếu các thành viên được phép tự tổ chức và trao quyền để làm những công việc của họ theo cách phù hợp nhất. Vì vậy, ScrumMaster cần là người hiểu và tin tưởng rằng vai trò lãnh đạo kiểu phục vụ là tố chất rất quan trọng.

Nói cách khác, một trong những vai trò quan trọng nhất của ScrumMaster là phục vụ nhóm phát triển trong suốt Sprint thông qua việc gỡ bỏ các trở ngại nhiều nhất có thể để bảo vệ họ càng nhiều càng tốt trước những phiền toái đến từ bên ngoài.

KỸ NĂNG TỐT VỀ MẶT TỔ CHỨC

Nếu Scrum là hoàn toàn mới với nhóm phát triển và Product Owner (hoặc thậm chí họ đã quen với Scrum), thì họ sẽ dựa vào ScrumMaster để được trợ giúp tổ chức các cuộc họp theo yêu cầu của Scrum nhằm thu được những lợi ích từ Scrum và để triển khai các công việc của họ.

Những cuộc họp mà rõ ràng ScrumMaster cần giúp tổ chức đó là họp Lập kế hoạch Phát hành, họp Lập kế hoạch Sprint, Họp Scrum Hàng ngày, Sơ kết Sprint và Họp cải tiến Sprint.

Nếu không có sự trợ giúp từ ScrumMaster, nhóm phát triển và Product Owner sẽ khó khăn trong việc ghi chép lại toàn bộ các cuộc họp trong khi họ đang cố gắng tập trung vào công việc của mình.

KỸ NĂNG TỐT VỀ GIAO TIẾP

Với việc ScrumMaster là người được cho là người hiểu biết nhất về Scrum, thì không còn nghi ngờ gì khi kỹ năng giao tiếp cần được xếp hạng cao trong số những kỹ năng và phẩm chất mà ScrumMaster cần có.

Tại sao vậy? Bởi vì ScrumMaster của một dự án được kỳ vọng sẽ giao tiếp với nhiều người, bao gồm thành viên trong nhóm phát triển, các nhóm khác, các nhà quản lý kinh doanh, quản lý kỹ thuật và Product Owner để giúp họ hiểu những lợi ích và yêu cầu của Scrum.

Một khía cạnh ít được biết đến trong vai trò ScrumMaster đó là trợ giúp Product Owner trong việc chuẩn bị và tham dự các cuộc họp để báo cáo với quản lý, điều này rõ ràng đòi hỏi ScrumMaster cần giao tiếp giỏi.

KỸ NĂNG THUYẾT TRÌNH TUYỆT VỜI

Do giao tiếp là một trong những kỹ năng quan trọng nhất của ScrumMaster, nên khả năng thuyết trình là một kỹ năng khác mà vai trò này cần có để thành công. Cho dù

Chương 14 • Khi nào bạn cần một ScrumMaster?

sử dụng PowerPoint hay bất cứ công cụ thuyết trình nào để truyền thông cho những thành viên còn lại của tổ chức, thì việc ScrumMaster cần trau dồi kỹ năng thuyết trình càng nhiều càng tốt là rất quan trọng.

KỸ NĂNG GIẢI QUYẾT XUNG ĐỘT

Trừ khi bạn đủ may mắn để làm việc trong một công ty mà mọi người đều yêu quý nhau hoặc nơi mà Scrum đã chứng minh được sự thành công bằng việc không có sự xung đột giữa các cá nhân, nếu không nhóm của bạn có thể trải nghiệm với những xung đột. Một trong những kỹ năng mà ScrumMaster nên làm chủ hoàn toàn là biết cách giúp các thành viên giải quyết các xung đột của họ để không gây ảnh hưởng tới khả năng chuyển giao của nhóm. Đây là một khía cạnh khác ít được biết tới trong vai trò lãnh đạo của ScrumMaster. Khía cạnh này thường không được thảo luận trong một lớp học, nhưng trong thực tiễn lại là thứ mà ScrumMaster được kỳ vọng sẽ trợ giúp để giữ cho nhóm tiến về phía trước.

KỸ NĂNG TỐT VỀ PHÁT TRIỂN NHÂN LỰC

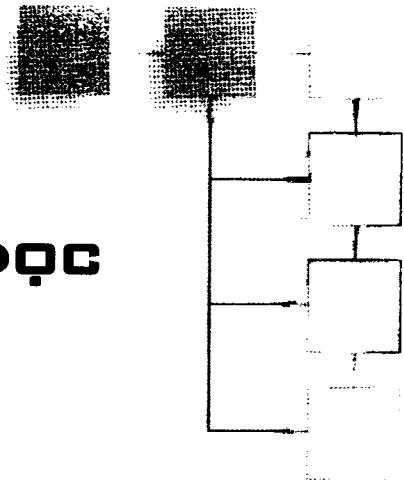
Đây là một kỹ năng khác mà ScrumMaster nên có để giúp hướng dẫn và phát triển nhóm thành nhóm có hiệu suất cao. Điều này hầu như được thực hiện thông qua việc khích lệ và thử thách liên tiếp.

TÓM TẮT

ScrumMaster là một trong những vai trò quan trọng nhất của dự án Scrum, cho dù ai đó thích hay không thích thì một số người vẫn kỳ vọng ScrumMaster không chỉ trợ giúp nhóm gỡ bỏ các trở ngại mà còn hỗ trợ họ giải quyết các xung đột và đưa nhóm trở thành một nhóm có hiệu suất cao. Do đó, ScrumMaster cần có ít nhất là bảy phẩm chất mà chúng ta đã xem xét trong chương này.

CHƯƠNG 15

NHẮN GỬI BẠN ĐỌC



Đến đây là bạn đã đi tới phần cuối của cuốn sách. Cảm ơn và xin chúc mừng bạn! Thay vì để bạn đơn độc trong hành trình triển khai Agile, trong cuốn sách này chúng tôi dành một vài lời khuyên về những cách thức để bạn áp dụng với các dự án thực tế.

Trước tiên, chúng tôi giả định là bạn đã đọc Chương 1 trước đó, hoặc là những giới thiệu về Agile và Scrum hay những điều mới mẻ về khóa học Scrum mà bạn vừa thực hiện với nhóm của mình. Thậm chí là nếu bạn đã từng đọc một cuốn sách khác hay sách trắng (whitepaper) về Agile và Scrum thì Chương 1 vẫn sẽ trợ giúp bạn hiểu cẩn bản về Agile/Scrum và thích ứng với Scrum thành công cho môi trường có những ràng buộc, những thứ không cho phép bạn sử dụng Scrum "ngoài khuôn khổ", giống như hầu hết những tình huống trong dự án mà chúng tôi đã gặp phải.

Để thu được nhiều lợi ích từ những kiến thức đã học trong cuốn sách này, bạn nên:

- Sử dụng hoặc tùy chỉnh những câu hỏi trong các bản khảo sát Chương 13 để tự đánh giá độ sẵn sàng của Scrum trong môi trường dự án của bạn sớm nhất có thể khi bạn có ý tưởng triển khai Scrum. Tốt nhất là hãy làm điều này trước khi bạn bắt đầu hành trình làm dự án hoặc ngay sau khi bạn nhận được đèn xanh từ quản lý cho phép thực hiện dự án. Hãy cố gắng trung thực trong việc trả lời các câu hỏi.
- Do các nhà điều hành kinh doanh đặt nặng vấn đề tài chính, nên bạn hãy cố gắng kiềm chế sự hưng thú đối với Agile hoặc Scrum, hãy nói ít về việc bạn có thể chuyển giao phần mềm tốt hơn, nhanh hơn như thế nào và tập trung nhiều hơn về lợi nhuận tài chính. Nhà quản lý sẽ thích lắng nghe những ý kiến của bạn hơn nếu bạn lượng hóa những thứ mà bạn nghĩ rằng nhà quản lý sẽ nhận được từ dự án theo thuật ngữ về lợi nhuận và chi phí. Chúng tôi đã đề cập vấn đề này trong Chương 2.
- Mặc dù điều quan trọng nhất là có được sự hỗ trợ từ ban điều hành và quản lý cấp cao, nhưng hãy nhớ kiểm tra sự cam kết và cộng tác tốt từ quản lý cấp trung; điểm mấu chốt với dự án của bạn đó là sự tương tác giữa quản lý

cấp trung với các nhóm Scrum trong suốt thời gian triển khai. Như chúng tôi đã nói trong Chương 3, mối quan hệ với quản lý cấp trung sẽ xây dựng hoặc phá vỡ dự án của bạn, bắt kể bạn nhận được sự ủng hộ từ những nhà quản lý kinh doanh cấp cao.

- Dù bạn có tin hay không, nhưng chúng tôi đã thấy nhiều dự án Scrum bị chậm tiến độ hoặc chỉ đơn giản là thất bại do thiếu những yêu cầu tốt hoặc đúng cho Scrum chứ không phải do thiếu kinh phí hay hiểu biết về kỹ thuật. Vì vậy, trừ khi bạn là một chuyên gia trong việc thu thập yêu cầu cho Agile hoặc Scrum, nếu không Chương 4 sẽ có những kiến thức mà bạn muốn đọc để tìm hiểu về một kỹ thuật hữu ích cho tất cả mọi người. Phương pháp viết yêu cầu cũng ảnh hưởng đến sự thành công của bạn trong việc lập kế hoạch phát hành và lập kế hoạch Sprint và do đó chúng tôi đã trình bày với các bạn phương pháp để viết các yêu cầu hiệu quả trong Agile.
- Khi Agile và Scrum trở nên phổ biến hơn, một trong những trở ngại đối với việc triển khai chúng trên toàn doanh nghiệp là vấn đề về tốc độ nhóm, hoặc số điểm User story mà nhóm có thể chuyển giao ở mỗi phân đoạn (Sprint) và số điểm đó không thể so sánh được giữa các nhóm khác nhau. Vì vậy, nếu bạn muốn biết kỹ thuật giúp ước tính điểm Story mà có thể dễ dàng lý giải và so sánh được giữa các nhóm khác nhau, thì hãy tìm hiểu kỹ thuật này trong Chương 5.
- Mặc dù trên thực tế một số người mong rằng mình sẽ không phải làm gì với kiến trúc phần mềm nữa, nhưng do kiến trúc là điều cần thiết cho sự ổn định và khả năng mở rộng của một tòa nhà, nên kiến trúc phần mềm cũng rất quan trọng và nó ảnh hưởng tới chất lượng phần mềm, do đó quyết định sự thành công của dự án. Dựa trên những dự án Agile của chúng tôi, thì mấu chốt vấn đề ở đây là tìm hiểu cách để tìm ra kiến trúc hoặc ít nhất là ý đồ kiến trúc mà không cần dành hết một khoảng thời gian để xác định từ đầu và từ đó sử dụng kiến trúc này để cải thiện tốc độ nhóm và chất lượng phần mềm. Đây là chủ đề chính của Chương 6.
- Những lợi ích của một tầm nhìn kiến trúc tốt được thảo luận trong Chương 7. Bạn nên tận dụng những gì được trình bày trong chương này để hiểu cách áp dụng tầm nhìn kiến trúc trong việc lập kế hoạch phát hành, điều có thể mang lại giá trị kinh doanh nhiều nhất đồng thời tránh được việc có những User story dẫn tới sự phân đoạn và cuối cùng không phù hợp với thiết kế hệ thống.
- Mọi cá nhân đều quan trọng trong dự án Scrum, nhưng không nghi ngờ gì việc Product Owner chính là người làm cho doanh nghiệp yêu hay ghét nhóm của bạn. Bạn có thể thường xuyên muốn tham khảo Chương 8 để thấy những phẩm chất cá nhân và tính chuyên nghiệp mà một Product Owner cần phải có. Nếu vì một lý do nào đó, Product Owner của bạn không giống như người mà chúng tôi đã mô tả trong Chương 8, thì khi đó bạn sẽ có những khó khăn trong công việc của mình.

- Một trong số tất cả những điều quan trọng mà bạn nên kiểm tra đó là môi trường kiểm thử dành cho dự án đã sẵn sàng để sử dụng hay chưa. Vì chúng tôi đoán rằng bạn bị giới hạn về thời gian hoặc ngân sách, nên chúng tôi sẽ khuyến nghị bạn nên tiến hành ba loại kiểm thử quan trọng nhất đó là: kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp liên tục. Để biết cách tổ chức hạ tầng kiểm thử bạn hãy xem xét Chương 9.
- Mặc dù ngân sách, các yêu cầu, hạ tầng kiểm thử cùng một số thứ khác là những yếu tố quan trọng mà bạn cần quan tâm, nhưng chúng tôi cũng khuyến nghị bạn nên chú ý đến động lực giữa tất cả các thành viên trong nhóm, những người đã được xếp vào dự án. Trong tất cả những dự án mà chúng tôi đã tham gia thì làm việc nhóm luôn là thứ xây dựng hoặc phá vỡ dự án. Đây là điều mà Patrick Lencioni đã thảo luận trong cuốn sách *Five Dysfunctions of Teams* (tạm dịch: Năm bất ổn của nhóm). Hãy tham khảo Chương 10 thường xuyên khi cần bởi vì làm việc nhóm luôn là chìa khóa đi tới thành công của bạn.
- Dù bạn tin hay không thì thực tế cho thấy rằng nhóm tự quản trong dự án Scrum không giúp cho công việc quản lý dự án và lãnh đạo nhóm dễ dàng hơn, mà chỉ đơn giản là có đôi chút khác biệt. Trong một số trường hợp, thì thậm chí lãnh đạo và quản lý dự án còn gặp những thách thức phức tạp hơn. Vì vậy, chúng tôi khuyến nghị rằng bạn hãy xem xét động lực trong nhóm của mình càng sớm càng tốt, chúng tôi cũng khuyến nghị rằng bạn nên xem xét khía cạnh mềm trong quản lý dự án và lãnh đạo nhóm ngay khi có thể. Bạn sẽ tìm thấy những hướng dẫn cho điều này trong Chương 11.
- Ngay sau khi hoàn thành bản tự đánh giá mức độ sẵn sàng cho dự án Scrum của mình, bạn nên nhìn vào kết quả và cân nhắc xem khía cạnh nào của Scrum cần phải điều chỉnh để Scrum hoạt động tốt trong môi trường hiện tại. Bạn hãy quan tâm đến điều này càng sớm càng tốt, để có nhiều thời gian hơn cho việc nhận ra thích ứng nào tốt, thích ứng nào không và thậm chí tiến hành một chu kỳ thích ứng khác nếu cần. Hãy tham khảo tất cả kiến thức này trong Chương 12.
- Cuối cùng nhưng không kém phần quan trọng, Chương 14 sẽ cung cấp cho bạn một bản tổng hợp về các phẩm chất mà mỗi ScrumMaster nên có để thành công.

Chúc các bạn may mắn trong chuyến hành trình Agile/Scrum của mình!

Andrew Pham

Phuong-Van Pham

PHỤ LỤC A

TÌM HIỂU HAI CASE STUDY VỀ PHÁT TRIỂN SẢN PHẨM PHẦN MỀM

GIỚI THIỆU

Trong phụ lục này, chúng tôi sẽ giới thiệu cho bạn hai ví dụ về việc xây dựng và triển khai sản phẩm phần mềm thành công nhờ áp dụng những lời khuyên trong cuốn sách này, với các nhóm Scrum nhỏ gồm năm thành viên và các Sprint ngắn có độ dài một tuần.

RUBY VÀ RUBY ON RAILS (ROR)

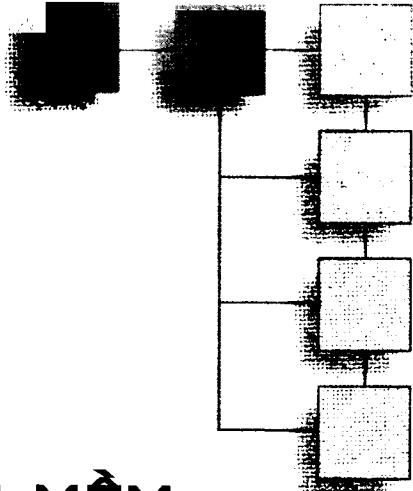
Để hiểu rõ hơn về hai ví dụ này (những ứng dụng thực tế), chúng tôi khuyến nghị bạn nên biết một chút về Ruby on Rails. Nếu bạn chưa từng có kinh nghiệm với Ruby on Rails nhưng đã sử dụng một số khung làm việc web (web framework) như Django hay CakePHP, bạn có thể thoải mái bỏ qua và chuyển đến tìm hiểu hai case study trong phụ lục này. Sẽ không có vấn đề gì nếu bạn chưa có một chút kinh nghiệm nào! Chúng tôi sẽ giải thích về Ruby và Ruby on Rails và tất cả những đoạn mã ví dụ khi trình bày về chúng.

Ngôn ngữ Ruby

Nếu bạn không phải là nhà phát triển Ruby, chúng tôi sẽ khái quát cú pháp căn bản của Ruby trong những trang tiếp theo nhằm cung cấp cho bạn một số kiến thức về ngôn ngữ này. Mặc dù đó không phải là những giới thiệu đầy đủ, nhưng sẽ giúp bạn hiểu các ví dụ được viết bằng Ruby trong phụ lục này.

Cú pháp và Cấu trúc

Cú pháp của Ruby cực kỳ đơn giản nhưng linh hoạt – lập trình viên có thể dễ dàng tiếp cận theo cách họ muốn. Ví dụ, trong khi các ngôn ngữ khác bạn phải luôn sử dụng dấu chấm phẩy (;) để kết thúc một dòng lệnh thì trong Ruby bạn không cần dùng nữa.



Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Tuy nhiên, nếu muốn, Ruby vẫn cho phép bạn sử dụng dấu chấm phẩy (;) để kết thúc mỗi dòng lệnh. Tất cả điều này phụ thuộc vào phong cách lập trình của bạn. Chúng tôi sẽ đồng thời trình bày về cú pháp và cấu trúc bởi vì chúng là những nội dung không tách biệt nhau.

Do Ruby là ngôn ngữ lập trình hướng đối tượng, nên chúng tôi sẽ đi sâu vào những nội dung về đối tượng và cách để tạo ra đối tượng trong Ruby. Các ngôn ngữ lập trình hướng đối tượng cho phép chúng ta cấu trúc chương trình theo những đối tượng trong thế giới thực, điều mà nói chung nhiều người cho rằng sẽ khiến cho việc tổ chức mã và phát triển chương trình dễ dàng hơn.

Lớp và Đối tượng

Trước hết, không chỉ với Ruby mà với tất cả các ngôn ngữ hướng đối tượng, việc hiểu được mối quan hệ giữa lớp và đối tượng là rất quan trọng. Một lớp đóng vai trò như một bản thiết kế để tạo ra một đối tượng. Một ví von thường được sử dụng đó là lớp như là bản thiết kế cho một ngôi nhà và mỗi đối tượng là một ngôi nhà thực được tạo ra từ bản thiết kế này. Các lớp rất quan trọng bởi chúng cho phép chúng ta thiết lập tính chất (thuộc tính) và hành vi của mỗi đối tượng được khởi tạo (tạo ra) từ nó.

Do đó, như ở ví dụ sau, chúng ta sẽ tạo một lớp rỗng để làm mô hình cho một đối tượng con người và sẽ hoàn chỉnh nó ở những bước tiếp theo trong chương này.

```
class Person  
end
```

Đã hoàn tất! Bây giờ chúng ta đã có lớp Person và có thể bổ sung các thuộc tính, hành vi để lớp này trở thành một mô hình thực sự về một con người. Sẽ rất đơn giản nếu chúng ta muốn tạo ra một đối tượng thực sự từ lớp này trong Ruby. Tất cả những gì bạn cần làm là sử dụng phương thức new như sau:

```
Person.new
```

Bây giờ, chúng ta có thể bắt đầu bổ sung các thuộc tính của một con người. Tuy nhiên, để làm điều đó, trước tiên chúng ta tìm hiểu về biến trong Ruby.

Biến

Các chương trình lưu trữ dữ liệu vào các biến và thao tác với chúng trong suốt thời gian tồn tại của chương trình. Tuy nhiên, biến trong Ruby khác với những ngôn ngữ định kiểu khác như Java, bởi vì bạn không cần phải xác định rõ ràng ngay từ đầu loại dữ liệu mà biến sẽ lưu trữ. Ví dụ, để tạo một biến lưu trữ một con số, một chuỗi hay một mảng trong Java bạn phải làm như sau:

```
int variableInJava = 10;  
String variableInJava = "Đây là một chuỗi";  
Array variableInJava = {1,2,3,4,5};
```

Trong khi, với Ruby tất cả những gì bạn cần làm là:

```
variableInRuby = 10;
variableInRuby = "Đây là một chuỗi"
variableInRuby = [1,2,3,4,5];
```

Mặc dù đây là một ví dụ cực kỳ đơn giản, nhưng với khả năng này, Ruby cho phép bạn dễ dàng tạo hoặc thay đổi giá trị của các biến trong suốt thời gian thực thi (khi chương trình đang chạy), điều này rất hữu ích khi độ phức tạp của chương trình gia tăng.

Bây giờ, hãy bổ sung thêm một số thuộc tính cho lớp Person.

```
Class Person
```

```
  # Đây là chú thích - dòng này sẽ bị trình thông dịch bỏ qua
  attr_accessor :name, :age, :weight, :height
end
```

Lúc này lớp Person đã có khả năng để tạo ra các đối tượng Person, mỗi đối tượng sẽ có tên (name), tuổi (age), cân nặng (weight) và chiều cao (height). Có hai thứ mới được giới thiệu trong ví dụ trên, đó là phương thức attr_accessor và một cú pháp lạ với ký hiệu ":" (dấu hai chấm).

Phương thức attr_accessor (sẽ được giải thích kỹ hơn sau) chủ yếu được sử dụng để tạo các biến thể hiện (thuộc tính) với tên tương ứng trong danh sách mà chúng tôi đã đưa ra (trong ví dụ này là các biến name, age, weight và height). Với mô tả này của lớp Person thì mỗi thể hiện (đối tượng) của lớp sẽ có các biến thể hiện. Ký hiệu ":" là một phương thức khởi tạo sẵn có (literal constructor) của lớp Symbol trong Ruby. Đây là cách cơ bản để xác định một thứ gì đó là biến. Với kiến thức này, chúng ta sẽ tiếp tục với các kiểu dữ liệu trong Ruby.

Kiểu dữ liệu

Ruby có nhiều kiểu dữ liệu khác nhau được thể hiện bởi các lớp String, Array, Hash, Fixnum, Symbol, ...

Trong Ruby, kiểu String (chuỗi) là cách thường được sử dụng để biểu diễn các ký tự, từ, câu hoặc bất cứ thứ gì tương tự như vậy. Để tạo một chuỗi trong Ruby, bạn có thể làm như sau:

```
@string = "Đây là một chuỗi"
```

Dòng lệnh trên sẽ tạo một biến thể hiện để lưu trữ chuỗi "Đây là một chuỗi".

Kiểu Array (mảng) được sử dụng để lưu trữ tập hợp các đối tượng trong Ruby. Ví dụ, mảng sẽ tương tự như khái niệm về một gia đình – một gia đình đơn thuần là một tập những con người có quan hệ với nhau. Để tạo ra một mảng trong Ruby, bạn có thể làm như sau:

```
@family = ['John', 'Mary', 'Adam', 'Susan']
```

Để lấy chuỗi 'John' từ mảng này, bạn sử dụng chỉ số như sau:

```
@family[0] # Lệnh này sẽ trả về chuỗi 'John'
```

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Kiểu Hash (bảng băm) tương tự như mảng, ngoại trừ việc bạn có thể xác định một khóa duy nhất cho mỗi giá trị chứa trong hash để giúp cho việc truy xuất sau này. Trở lại với ví dụ ở trên, thay vì dùng chỉ số là một con số để lấy chuỗi 'John', bạn có thể tạo một hash giống như sau:

```
@family = { 'Dad'=>'John', 'Mom'=>'Mary', 'Son'=>'Adam', 'Daughter'=>'Susan'}
```

Và sau đó bạn có thể lấy chuỗi 'John' từ hash bằng cách như sau:

```
@family['Dad']
```

Các số nguyên trong Ruby được biểu diễn chung bởi lớp Fixnums. Bạn có thể tạo một số đơn giản bằng cách như sau:

```
@number = 1
```

Lệnh này sẽ tạo một biến chứa giá trị là 1.

Kiểu Symbol tương tự với kiểu String nhưng chúng thường được sử dụng để xác định một số thứ khác - chẳng hạn định nghĩa về chính biểu tượng đó. Chúng thường được sử dụng để làm khóa cho các giá trị trong hash. Do đó, giống như ví dụ trước, bạn có thể tạo một hash về một gia đình như sau:

```
@family = { :dad => 'John', :mom => 'Mary', :son => 'Adam', :daughter => 'Susan'}
```

Phương thức

Phương thức là cách để mô tả một hành vi trong các chương trình. Thông thường, bạn có thể gọi một phương thức của một đối tượng nào đó để thực hiện một việc gì đó. Ví dụ, hãy bổ sung phương thức greet cho lớp Person.

```
class Person
  # Đây là chú thích - dòng này sẽ bị trình thông dịch bỏ qua
  attr_accessor :name, :age, :weight, :height

  def greet
    puts "Xin chào, tôi là #{self.name}."
  end
end
```

Nếu chúng tôi tạo một đối tượng Person và gán tên là 'John', sau đó gọi phương thức greet, sau đây là những gì xảy ra:

```
@person = Person.new
@person.name = "John"
@person.greet
=> Xin chào, tôi là John.
```

Như bạn đã thấy trong các giới thiệu ở trên, Ruby nhấn mạnh tới sự đơn giản và dễ dàng; do đó, nhiều chương trình được viết bằng Ruby thường giúp cả những người

không phải lập trình viên có thể được đọc và hiểu được, vì cú pháp của Ruby hướng tới việc phản ánh ngôn ngữ tự nhiên nhiều nhất có thể. Trong khi được coi là ngôn ngữ đơn giản, Ruby cũng là ngôn ngữ rất linh hoạt với khả năng kết hợp nhiều cấu trúc đơn giản để tạo ra những chương trình giải quyết được các mục đích phức tạp.

Bây giờ, hãy xem xét khung làm việc Ruby on Rails đồng thời giải thích mã Ruby được sử dụng trong mỗi ví dụ.

Ruby on Rails (RoR), khung làm việc Web

Ruby on Rails là một khung làm việc web (web framework) của David Heinemeier Hansson, được sử dụng kết hợp với Ruby. Thuật ngữ "khung làm việc web" trong thực tế thường được đề cập đến phần mềm cho phép bạn viết các chương trình tốt hơn bằng cách đưa ra những cách thức cụ thể hoặc đề xuất những cách để tổ chức mã nguồn. Khung làm việc web Ruby on Rails (RoR) cũng cung cấp cho bạn nhiều mã có thể tái sử dụng, điều này sẽ giúp bạn tránh lãng phí thời gian vào việc xây dựng lại những thứ đã có.

Hai nguyên tắc thường được nói đến của Ruby on Rails đó là *Don't repeat yourself* (tạm dịch: đừng lặp lại chính mình) và *Convention over configuration* (tạm dịch: quy ước về cấu hình). *Đừng lặp lại chính mình (DRY)* hiểu đơn giản là bạn có thể cấu trúc chương trình của mình theo cách để có thể tái sử dụng mã nguồn càng nhiều càng tốt, không lãng phí thời gian để phát triển những thứ giống nhau hết lần này đến lần khác. *Quy ước về cấu hình* có nghĩa là Ruby on Rails đi kèm với một cấu hình đã định nghĩa sẵn và thường rất phù hợp cho việc tạo ra hầu hết các ứng dụng web. Bạn không phải lãng phí thời gian để tinh chỉnh phần mềm cho phù hợp với nhu cầu. Tất nhiên, nếu cần bạn hoàn toàn vẫn có thể tùy chọn tinh chỉnh phần mềm. Những nguyên tắc này làm cho Ruby on Rails trở thành một sự chọn lựa khung làm việc web tốt cho những nhóm Scrum mong muốn tạo ra ứng dụng web.

MVC

Một trong những lợi thế lớn nhất của Ruby on Rails là hình thức gọi ý (bản chất là sự bắt buộc) bạn về cách tổ chức mã nguồn. Kiến trúc ba tầng, thực tế được biết đến với tên gọi MVC hay Model-View-Controller. Kiến trúc ba tầng hiểu đơn giản nghĩa là có ba tầng mã khác nhau có liên quan cùng làm việc với nhau để tạo ra những hiệu quả mong muốn cho một ứng dụng. Trong phần sau, chúng tôi sẽ giải thích từng phần của kiến trúc MVC, nhưng theo một thứ tự khác với thứ tự thông thường đã được trình bày ở những cuốn sách khác.

View – Khung nhìn. Khung nhìn nghĩa là những gì mà người dùng nhìn thấy được. Khi bạn vào một website thì trang hiển thị trên màn hình được gọi là Khung nhìn. Thông thường, khung nhìn là sự kết hợp HTML, CSS và nhúng thêm mã Ruby, chịu trách nhiệm (theo thứ tự tương ứng) cấu trúc, định kiểu trình bày (style) và đưa dữ liệu đến Khung nhìn.

Vậy những gì thường được hiển thị trên Khung nhìn? Đó là dữ liệu, thứ có liên quan nhất tới người dùng – nhưng nó đến từ đâu? Đó chính là Model.

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Model – Mô hình. Mô hình là cách mà dữ liệu trong ứng dụng được lưu trữ, lấy ra và thao tác. Thông thường, Mô hình được mô tả như một phần có chứa logic nghiệp vụ và dữ liệu.

Trong Rails, Mô hình là một lớp thường sử dụng ActiveRecord¹ để truy xuất dữ liệu trong cơ sở dữ liệu. Sử dụng ActiveRecord để tạo ra một ánh xạ đối tượng-quan hệ giữa các đối tượng với dữ liệu được lưu trữ trong cơ sở dữ liệu. Điều đó có nghĩa là mỗi dòng trong cơ sở dữ liệu có khả năng được tương tác như một đối tượng (thay vì dùng câu lệnh SQL phức tạp và dài dòng), do đó việc thao tác với dữ liệu trong ứng dụng đơn giản và dễ hiểu hơn. Ví dụ, bạn có một mô hình Computer (máy tính). Mô hình Computer có một vài thuộc tính như tốc độ, bộ nhớ và ổ đĩa cứng. Với ActiveRecord, bạn sẽ có một bảng trong cơ sở dữ liệu tên là Computer (máy tính) với các cột Speed (tốc độ), Memory (bộ nhớ), Hard Drive (ổ đĩa cứng). Bây giờ, khi muốn thêm một dòng mới vào bảng Computer, bạn sẽ khai báo và khởi tạo một đối tượng computer mới ở đâu đó trong ứng dụng (thường trong tầng Controller hoặc Model) sử dụng dòng mã sau:

```
@computer = Computer.new
```

Sau đó, bạn sẽ thiết lập các thuộc tính của đối tượng computer theo ý muốn (ActiveRecord tự động ánh xạ tên các thuộc tính với cột trong cơ sở dữ liệu mà không cần phải khai báo):

```
@computer.speed = "2"  
@computer.memory = "1024"  
@computer.hard_drive = "320"
```

Đây là chính là điểm mang tới sự thú vị của ActiveRecord trong Rails – nếu muốn lưu đối tượng computer mới này vào cơ sở dữ liệu, tất cả những gì bạn phải làm đó là:

```
@computer.save
```

Vậy là xong! Bây giờ đối tượng computer đã được lưu vào cơ sở dữ liệu. Nhưng nếu sau này bạn muốn tìm lại nó để kiểm tra thuộc tính thì phải làm sao? Tất cả những gì bạn phải làm là (giả sử chiếc máy tính này là máy tính đầu tiên trong bảng Computer):

```
@computer = Computer.find(1)
```

Bây giờ, bạn có thể kiểm tra thuộc tính của nó với đoạn mã sau:

```
# Đây là chú thích  
# lệnh puts thực hiện việc in ra màn hình console những gì bạn yêu cầu -  
# trong trường hợp này là: "speed", "memory" và "hard_drive"  
puts @computer.speed  
puts @computer.memory  
puts @computer.hard_drive
```

Trong trường hợp bạn muốn xóa một máy tính, tất cả những gì bạn phải làm là (nếu bạn biết ID của chiếc máy tính này trong cơ sở dữ liệu):

```
@computer.destroy
```

¹ ActiveRecord: Là một lớp được xây dựng sẵn làm nhiệm vụ thao tác với Cơ sở dữ liệu

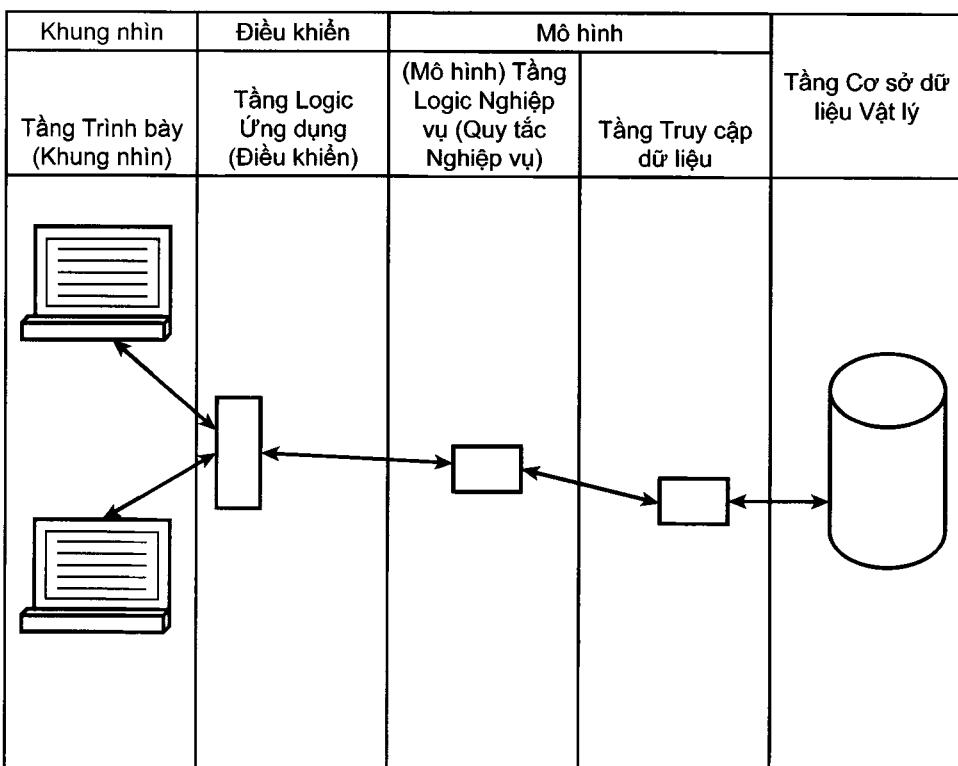
ActiveRecord chắc chắn sẽ làm cho các nhà phát triển dễ dàng truy cập, thao tác, hoặc xóa dữ liệu trong cơ sở dữ liệu hơn bằng cách hoạt động như là một tầng trung gian, thông minh giữa các lớp của bạn và cơ sở dữ liệu.

Câu hỏi cuối cùng là, làm thế nào để ứng dụng biết những gì sẽ hiển thị trên Khung nhìn và do đó biết cần phải tìm dữ liệu nào trong Mô hình để hiển thị lên Khung nhìn?

Controller – Điều khiển. Điều khiển đóng vai trò như người đứng giữa Khung nhìn và Mô hình. Khi bạn gõ một URL trên trình duyệt, tức là bạn đang gửi đi một yêu cầu tới server (nơi mà website được cài đặt hoặc lưu trữ trên đó). Một cấu trúc giống như bộ điều khiển sẽ nhận các yêu cầu và hiển thị Khung nhìn mà bạn mong muốn, trong khi đó bộ điều khiển này cũng lấy dữ liệu từ Mô hình để hiển thị lên Khung nhìn.

Hình A.1 dưới đây sẽ trình bày tóm tắt về sự phù hợp giữa RoR và khung làm việc MVC, hình ảnh này có được chủ yếu từ những gì bạn đã thấy trong Chương 6 và 7.

Vì sao MVC lại tốt? Kiến trúc MVC cho phép bạn tổ chức mã theo cách mà, trong đó bạn chỉ thay đổi một thành phần nào đó để có được điều mà người dùng yêu cầu thay vì phải mất hàng giờ hoặc vài ngày để lục tìm toàn bộ ứng dụng và thay đổi các thành phần khác nhau.



Hình A.1

Sơ đồ Khung làm việc MVC.

Phụ lục A - Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Điều này có nghĩa rằng với những ràng buộc và điều kiện thực tế thì thay đổi là sẽ xảy ra và không thể tránh khỏi, do đó bạn có thể dễ dàng tùy biến và tiếp tục luồng công việc mà không bị phát sinh nhiều về chi phí phát triển.

QUẢN LÝ PHIÊN BẢN VÀ KIỂM THỬ ĐỐI VỚI VIỆC PHÁT TRIỂN WEB BẰNG ROR

Để phát triển hai case study trình bày trong phần phụ lục này, chúng tôi sử dụng Ruby on Rails kết hợp với Git để quản lý phiên bản và sử dụng Test::Unit để kiểm thử. Chúng tôi sẽ sớm giải thích lý do tại sao hai công cụ này giúp có được một môi trường thuận lợi cho việc phát triển tốt hơn.

Hệ thống Quản lý Phiên bản Git

Git là một hệ thống quản lý phiên bản mã nguồn mở và miễn phí được tạo ra như là một sự thay thế cho Subversion, Mercurial và CVS. Về cơ bản, một hệ thống quản lý phiên bản cho phép bạn lưu trữ toàn bộ mã nguồn dưới dạng các bản chụp (snapshot) ở một nơi tập trung. Điều này có nghĩa là nếu bạn bắt đầu làm việc trên một phần của ứng dụng và nhận ra rằng mình đã làm cho toàn bộ ứng dụng bị rối tung cả lên, bạn có thể dễ dàng lấy lại mã nguồn hoạt động trước đó từ hệ thống quản lý phiên bản và bắt đầu làm lại.

Tuy nhiên, lợi ích thực sự của một hệ thống quản lý phiên bản như Git chính là khả năng cộng tác. Với việc toàn bộ mã nguồn được lưu trữ ở một nơi tập trung, nhiều nhà phát triển có thể làm việc song song cùng nhau trên những phần khác nhau hay cùng trên một phần của ứng dụng (đây là một nguyên lý quan trọng của cuốn sách này). Sau đó họ “đẩy” phần mã nguồn mà mình đã làm việc ngược trở lại kho lưu trữ để Git hợp nhất tất cả lại với nhau và tạo ra một bản sao mã nguồn. Bản sao này tích hợp tất cả các thay đổi được tạo ra song song bởi các lập trình viên khác nhau (đây cũng là một nguyên lý quan trọng nữa của cuốn sách này).

Việc quyết định sử dụng hệ thống quản lý phiên bản nào phụ thuộc vào nhu cầu của ứng dụng, chúng tôi chọn Git bởi vì Git dễ sử dụng, cài đặt đơn giản và miễn phí (bởi vì là phần mềm mã nguồn mở).

Kiểm thử và Khung làm việc Kiểm thử

Kiểm thử quan trọng không những vì kiểm thử giúp đảm bảo các chức năng của ứng dụng hoạt động như mong muốn trong khi không cần nhiều nỗ lực (và do đó cho phép sự phát triển linh hoạt hơn bởi việc thay đổi ứng dụng sẽ tốn ít chi phí vì bạn đã có các kiểm thử để đảm bảo ứng dụng vẫn hoạt động tốt) mà còn bởi vì kiểm thử đảm bảo ứng dụng sẽ đáp ứng đầy đủ mọi nhu cầu của khách hàng, đặc biệt là với các kiểm thử chấp nhận của người dùng.

Test::Unit là một khung làm việc dành cho kiểm thử trên Ruby và Ruby on Rails. Mặc dù tên của khung làm việc này như vậy, nhưng nó có thể dùng cho cả kiểm thử đơn vị, kiểm thử chức năng và kiểm thử tích hợp. Đây là một khung làm việc kiểm thử rất đơn giản và dễ hiểu với phần lớn các kiểm thử được thực hiện thông qua các khẳng định (assertion). Chương trình sử dụng các khẳng định này là cách để nói rằng “kết quả do đoạn mã sinh ra khớp với kết quả mong muốn”.

Có ba loại kiểm thử chính mà bạn có thể thực hiện bằng cách sử dụng công cụ kiểm thử tự động như Test::Unit trong một dự án RoR:

1. **Kiểm thử đơn vị:** Kiểm thử độc lập từng thành phần nguyên tử (không thể tách ra thành các phần nhỏ hơn nữa, thường là các phương thức) của ứng dụng, thông thường là dành để kiểm tra lỗi logic với sự chặt chẽ của dữ liệu đầu vào và đầu ra.
2. **Kiểm thử chức năng:** Kiểm thử sự phối hợp của từng thành phần nguyên tử độc lập của ứng dụng mà không có giao diện (thường là khung nhìn).
3. **Kiểm thử tích hợp:** Kiểm thử sự phối hợp của từng thành phần nguyên tử độc lập của ứng dụng kèm với giao diện (thường là khung nhìn).

Kiểm thử Tự động

Như chúng tôi đã giải thích trong Chương 9, kiểm thử tự động khác với kiểm thử thủ công ở chỗ là có một ứng dụng phần mềm thực thi tất cả các kiểm thử mã nguồn thay vì bạn phải tự mình kiểm thử thủ công từng phần mã nguồn (công việc có thể tốn hàng giờ, thậm chí hàng ngày tùy thuộc vào mã nguồn). Nhưng để thực hiện kiểm thử tự động thì bạn phải làm thêm một ít công việc và tạo các kiểm thử sử dụng những khung làm việc kiểm thử có sẵn, chẳng hạn như Test::Unit. Tuy nhiên, phần việc làm thêm này sẽ giúp bạn hạn chế tối đa chi phí khi thay đổi mã nguồn và thêm các tính năng mới, bởi vì nếu bạn lo ngại về việc làm hỏng các phần khác của ứng dụng sau khi đã thay đổi một phần nào đó, thì bạn chỉ cần đơn giản chạy một câu lệnh và tất cả các kiểm thử sẽ được thực thi để đảm bảo không có gì bị hỏng cả.

Kiểm thử Hồi quy

Kiểm thử hồi quy nhằm đảm bảo rằng mọi thay đổi được thực hiện với mã nguồn sẽ không làm hỏng bất cứ phần nào khác mà có thể dẫn đến việc mất các chức năng.

Có thể thực hiện kiểm thử hồi quy một cách dễ dàng bằng cách sử dụng kiểm thử tự động để tự động chạy tất cả các kiểm thử. Việc có kiểm thử tự động tốt (nghĩa là nó sẽ kiểm thử tất cả mã nguồn trong những tình huống khác nhau) cũng đồng nghĩa với việc có kiểm thử hồi quy tốt. Điều này là rất quan trọng, bởi vì nó giảm thiểu chi phí do thay đổi những phần đã hoàn thiện của ứng dụng – điều mà chúng ta không thể tránh khỏi. Lý do mà kiểm thử tự động có thể giúp giảm thiểu chi phí khi thay đổi những phần đã viết mã là vì nếu bạn thay đổi một phần của ứng dụng, thì bạn chỉ cần chạy các kiểm thử tự động để chắc chắn rằng không có gì bị hỏng cả. Do tất cả những bất ổn của mã nguồn mới đều đã bị loại bỏ nên bạn sẽ ít phải e dè về những thay đổi đối với các thành phần của ứng dụng.

Kiểm thử Chấp nhận Người dùng

Kiểm thử chấp nhận người dùng là một trong những quy trình quan trọng nhất thường được triển khai ở cuối dự án. Quy trình này bao gồm việc chỉ định một người dùng để thiết kế các tình huống được khách hàng thực hiện trong thực tế, người này phải khác với người đã tạo ra các kiểm thử tự động. Việc này giúp bạn đảm bảo ứng dụng đáp ứng tốt tất cả các nhu cầu của người dùng và làm họ hài lòng. Kiểm thử chấp nhận người dùng là rất quan trọng, bởi vì ứng dụng vẫn có thể hoạt động như mong muốn nhưng vẫn chưa thỏa mãn đầy đủ các nhu cầu của người dùng. Việc triển khai kiểm thử chấp nhận người dùng đảm bảo có được điều đó.

Khung làm việc Kiểm thử Test::Unit

Test::Unit là một khung làm việc dành cho kiểm thử trên Ruby và Ruby on Rails. Mặc dù tên của khung làm việc này như vậy, nhưng nó có thể dùng cho cả kiểm thử đơn vị, kiểm thử chức năng và kiểm thử tích hợp. Đây là một khung làm việc kiểm thử rất đơn giản và dễ hiểu với phần lớn các kiểm thử được thực hiện thông qua các khẳng định (assertion). Chương trình sử dụng các khẳng định này là cách để nói rằng “kết quả do đoạn mã sinh ra khớp với kết quả mong muốn”.

CASE STUDY 1 (NOSHSTER)

Noshster là một mạng xã hội trên toàn thế giới dành cho những người sành ăn viết blog về những món mà họ đã ăn, đồng thời cũng là nơi để họ chia sẻ và đánh giá về các nhà hàng mà họ đã tới.

Tầm nhìn và Mục tiêu của Sản phẩm

Tầm nhìn sản phẩm của Noshster đó là tạo ra mạng xã hội cho những người sành ăn, để họ có thể theo dõi các món mà họ đã ăn một cách trực quan và đơn giản, đồng thời khám phá những loại thực phẩm mới bằng cách dõi theo (follow) các thành viên khác của Noshster - những người mà họ thấy thú vị và chia sẻ đánh giá của mình về các nhà hàng nơi mà họ đã ăn những món khác nhau.

Mục tiêu chung của Noshster là mang lại cho người dùng khả năng thỏa mãn sự khát khao bằng cách giúp họ tìm thấy những nhà hàng tốt nhất nơi phục vụ các món ăn đặc biệt.

Sử dụng phương pháp đã được giới thiệu trong Chương 8 để giúp Product Owner xác định tầm nhìn sản phẩm, tầm nhìn của Noshster có thể được tóm tắt như sau:

- **AI:** Người sành ăn
- **Tại sao:** Tìm kiếm những món ăn ngon nhất thế giới
- **Cái gì:** Những món ăn được xếp hạng, chứ không phải các nhà hàng
- **Ở đâu:** Món ăn ngon nhất bằng cách nhìn vào đánh giá các món ăn của các vùng khác nhau trên khắp thế giới.

- Khi nào: 24 x 7

Thu thập yêu cầu bằng cách sử dụng kỹ thuật trực quan trong cuốn sách

Sử dụng các kỹ thuật thu thập yêu cầu trực quan trong Chương 4, nhóm có thể xác định các yêu cầu dưới đây bằng cách sử dụng phép loại suy rừng và cây (Hình A.2).

Tầm nhìn Kiến trúc và Lập Kế hoạch Phát hành/kế hoạch Sprint

Với Noshster, như một ví dụ trong cuốn sách này, chúng tôi quyết định triển khai ứng dụng theo lát cắt dọc. Tức là thay vì phát triển các tính năng nằm trên cùng một “cây” trong các Sprint khác nhau, chúng tôi thực hiện và hoàn thành các tính năng trên cùng một cây trong một Sprint. Chúng ta sẽ xem xét một ví dụ khác về ứng dụng được phát triển theo lát cắt ngang trong case study tiếp theo.



Hình A.2

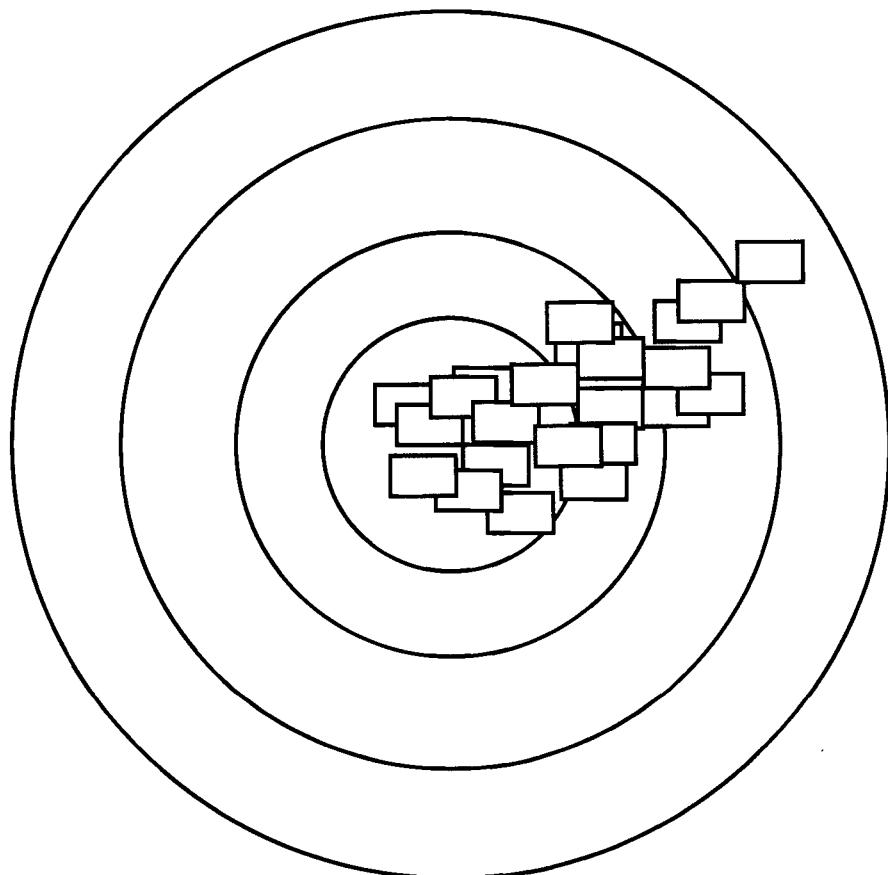
Phân cấp theo “rừng và cây” các yêu cầu của Mạng xã hội ẩm thực Noshster.

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Việc bạn phát triển ứng dụng theo lát cắt dọc hay lát cắt ngang phụ thuộc vào nhiều yếu tố như:

1. Bạn cần các tính năng nền tảng nào để phát triển phần còn lại của ứng dụng một cách hiệu quả mà không cần phải thiết kế lại là gì? Bạn cần mỗi "cây" tính năng được phát triển hoàn chỉnh trước khi chuyển sang giai đoạn tiếp theo, hay chỉ cần một tập hợp nhỏ các tính năng trên mỗi "cây" đó?
2. Cách tổ chức các tính năng theo kiểu nào trong suốt tiến trình phát triển cho phép bạn cung cấp nhiều giá trị kinh doanh nhất cho Product Owner hoặc doanh nghiệp?

Nhưng trước tiên chúng ta hãy quay trở lại quy trình và xem cách sử dụng các khuyến nghị đã được đưa ra trong cuốn sách để giúp nhìn thấy mọi thứ rõ ràng hơn. Trước tiên, hãy tập hợp các yêu cầu lại với nhau, hình ảnh trông khá nhiều giống như Hình A.3 dưới đây.

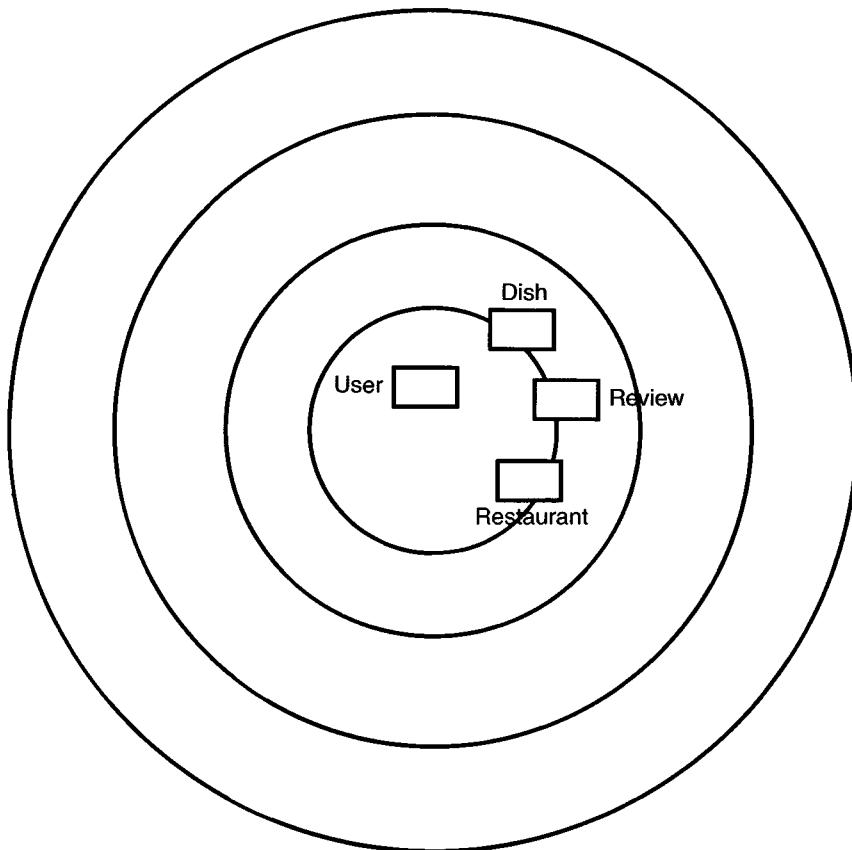


Hình A.3

Hình ảnh khi lần đầu tiên danh sách các yêu cầu được tập hợp lại.

Tâm nhìn Kiến trúc

Sử dụng phương pháp đã học trong Chương 6, chúng tôi có thể nhanh chóng nhìn thấy bức tranh rõ ràng hơn, như là trong Hình A.4 dưới đây.



User	Người dùng
Dish	Món ăn
Review	Đánh giá
Restaurant	Nhà hàng

Hình A.4

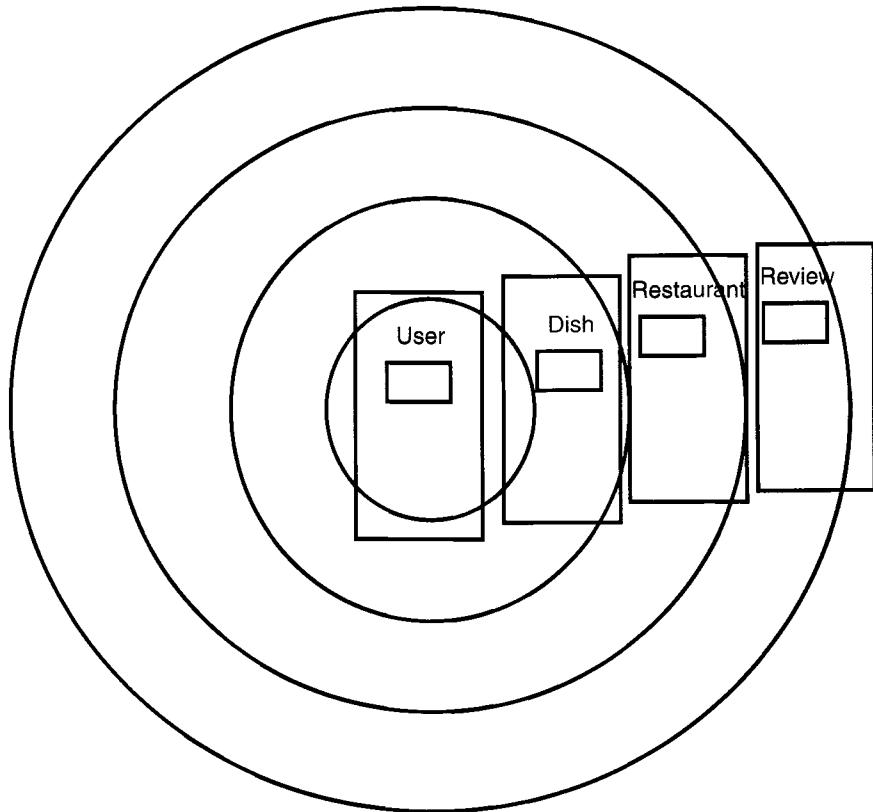
Quan sát “rừng cây”.

Trong cuộc thảo luận với Giám đốc điều hành, nhóm Scrum đã đề xuất và được Giám đốc điều hành đồng ý cho nhóm tập trung vào khái niệm người dùng (user) trước tiên để đảm bảo xác định được tất cả các kiểu người dùng khác nhau trên thế giới là một thành phần nền tảng cho ứng dụng toàn cầu này. Sau đó, nhóm sẽ tập trung vào món ăn, nhà hàng và cuối cùng là đánh giá. Nói cách khác, nhóm đang xây dựng một phần mềm theo lát cắt dọc xung quanh các yếu tố dữ liệu chung. Nhóm đã thực hiện như vậy bởi vì việc có trước được tất cả các đặc điểm về người dùng sẽ cho phép công

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

ty tin chắc rằng đây thực sự sẽ là phần mềm có phạm vi toàn cầu. Điều này cho thấy rằng tầm nhìn kiến trúc của họ trông giống như những gì được thể hiện trong Hình A.5.

Bằng cách nhìn vào các yếu tố dữ liệu nghiệp vụ, nhóm Scrum nhận thấy rằng kiến trúc dữ liệu ứng dụng sẽ giống như Hình A.6, ít nhất là ở mức khái quát.

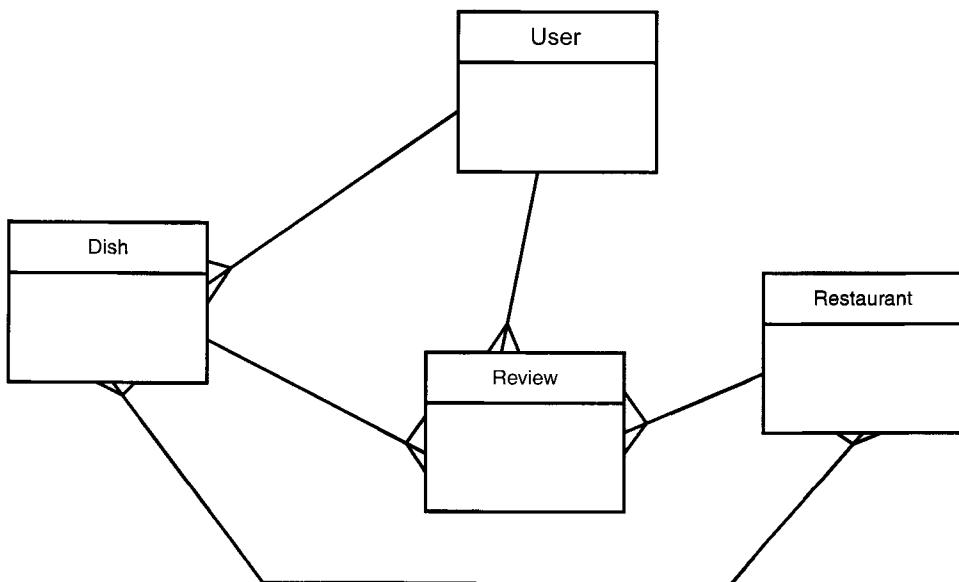


Hình A.5

Lát cắt Noshster theo chiều dọc.

**Lập Kế hoạch Phát hành và
Lập Kế hoạch Sprint cho Noshster**

Rừng	Cây	Lá
Mạng xã hội về ẩm thực		
Bản phát hành 1	Sprint 1	
	Quản lý người dùng	Đăng ký, Đăng nhập, Đăng xuất, Xem hồ sơ, Chỉnh sửa hồ sơ, Xóa hồ sơ, Duyệt người dùng
	Sprint 2	
	Quản lý món ăn	Thêm món ăn, Sửa món ăn, Xem món ăn, Duyệt món ăn
Bản phát hành 2	Sprint 3	
	Quản lý nhà hàng	Thêm nhà hàng, Sửa nhà hang, Xem nhà hàng, Duyệt nhà hàng
	Sprint 4	
	Quản lý đánh giá	Thêm đánh giá, Sửa đánh giá, Xóa đánh giá

**Hình A.6**

Mô hình dữ liệu Noshster ở mức cao (khái quát).

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Ước tính dự án sử dụng kỹ thuật dựa trên tiêu chí khách quan

Dưới đây là bảng tóm tắt các điểm ước tính đối với từng Story trong mỗi Sprint sử dụng kỹ thuật dựa trên tiêu chí khách quan đã trình bày trước đó trong cuốn sách. Bạn có thể tìm các công thức được sử dụng để tính toán mỗi cột đó trong Chương 5.

Hạng mục Backlog (Story)	Đặc điểm					Tổng UP (điểm chưa hiệu chỉnh)	Hệ số nhân	AP (điểm đã hiệu chỉnh)	ED (Khía cạnh Môi trường)	PPS (= AP * ED)/36)					
	Loại tương tác	Quy tắc nghiệp vụ	Thực thi	Loại thao tác dữ liệu											
Bản phát hành 1															
Sprint 1															
Đăng ký	3	1	1	2	7	1	7	18	3.5						
Đăng nhập	3	1	1	2	7	1	7	18	3.5						
Đăng xuất	3	1	1	1	6	1	6	18	3						
Xem hồ sơ	3	1	1	1	6	1	6	18	3						
Sửa hồ sơ	3	1	1	3	8	1	8	18	4						
Xóa hồ sơ	3	1	1	1	6	1	6	18	3						
Duyệt người dùng	3	1	1	1	6	1	6	18	3						
Sprint 2															
Thêm món ăn	3	1	1	2	7	1	7	18	3.5						
Sửa món ăn	3	1	1	3	8	1	8	18	4						
Xem món ăn	3	1	1	1	6	1	6	18	3						
Duyệt món ăn	3	1	1	1	6	1	6	18	3						
Bản phát hành 2															
Sprint 3															
Thêm nhà hàng	3	1	1	2	7	1	7	18	3.5						
Sửa nhà hàng	3	1	1	3	8	1	8	18	4						
Xem nhà hàng	3	1	1	1	6	1	6	18	3						
Duyệt nhà hàng	3	1	1	1	6	1	6	18	3						
Sprint 4															
Thêm đánh giá	3	1	1	2	7	1	7	18	3.5						
Sửa đánh giá	3	1	1	3	8	1	8	18	4						
Xóa đánh giá	3	1	1	1	6	1	6	18	3						

Phát triển Noshster

Bây giờ, chúng ta bắt đầu xem xét các đoạn mã được tạo ra từ mỗi Sprint ở mỗi bản phát hành của ứng dụng ví dụ và tìm hiểu cách tích hợp chúng.

Bản phát hành Noshster 1 – Sprint 1

Mục tiêu của sprint đầu tiên của bản phát hành đầu tiên là tạo được một nền tảng vững chắc cho các tính năng quản lý người dùng mà dựa vào đó sẽ xây dựng Sprint thứ 2.

Các Story (Bản phát hành 1—Sprint 1)

Ở Sprint 1, bản phát hành 1 sẽ phát triển những story sau:

1. Đăng ký
2. Đăng nhập
3. Đăng xuất
4. Xem hồ sơ
5. Sửa hồ sơ
6. Xóa hồ sơ
7. Duyệt người dùng

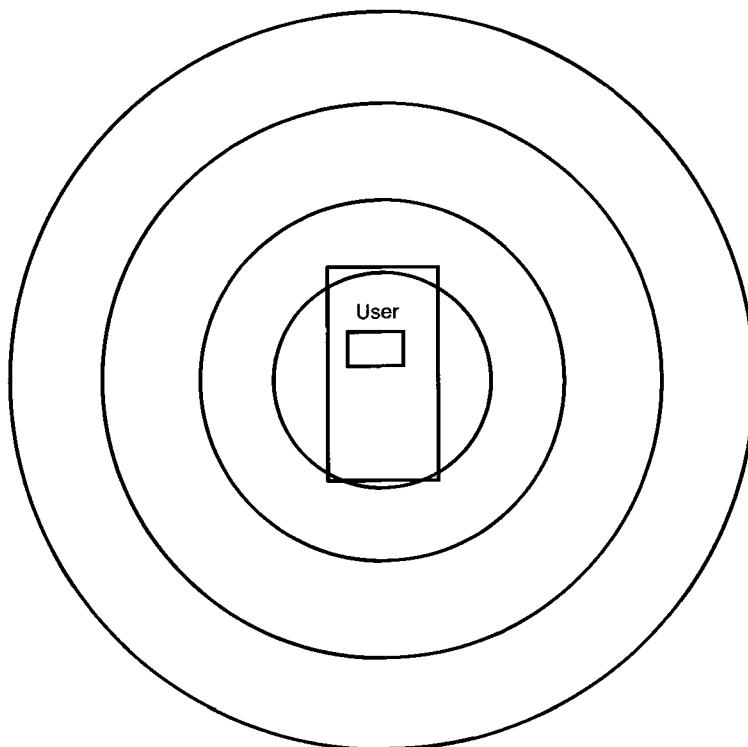
Trong Hình A.7, bạn sẽ nhìn thấy cách tổ chức các mô hình dữ liệu cho Bản Phát hành 1 – Sprint 1

Trong Hình A.8, bạn sẽ tìm thấy mô tả chi tiết hơn về mô hình dữ liệu cho Bản phát hành 1 – Sprint 1.

Model – Mô hình

Mô hình User (người dùng) cho phép ứng dụng Noshster lưu trữ thông tin đăng ký của người dùng. Mô hình này sử dụng plugin Authlogic và xác định rằng người dùng tiềm năng không cần phải gõ mật khẩu xác nhận (chỉ có tên người dùng, mật khẩu, email,...) để đăng ký. Authlogic là một plugin quản lý người dùng tuyệt vời, có thêm khả năng đăng ký và xác thực người dùng cho ứng dụng. Plugin này trừu tượng hóa một số chi tiết để bạn không phải lo lắng về việc viết mã cho những phần thường dùng và cố định của một ứng dụng. Một số thứ được trừu tượng hóa như kiểm tra tính hợp lệ người dùng (đảm bảo người dùng nhập dữ liệu cho một số trường nhất định như tên người dùng, mật khẩu, hoặc email) và mã hóa mật khẩu (thao tác với mật khẩu nhằm lưu mật khẩu vào trong cơ sở dữ liệu dưới dạng sao cho nếu cơ sở dữ liệu bị xâm nhập thì không ai có thể đăng nhập bằng mật khẩu của bạn bởi vì nó đã được mã hóa).

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm



Hình A.7

Tập trung phát triển trên khái niệm nghiệp vụ của người dùng.

User
First Name
Last Name
Address
Location
Referred By

Hình A.8

Mô hình dữ liệu trong Sprint 1 - Bản phát hành 1.

```
class User < ActiveRecord ::Base
  acts_as_authentic do |c|
    c.require_password_confirmation = false
  end
end
```

Mô hình UserSession – Cho phép ứng dụng Noshster xác thực thông tin đăng nhập của người dùng với thông tin đã đăng ký được lưu trữ trong cơ sở dữ liệu. Do đó, khi người dùng cố gắng đăng nhập với những thông tin không khớp với những gì đã lưu trong cơ sở dữ liệu, ứng dụng sẽ đưa ra một thông báo lỗi thân thiện. Trường hợp ngược lại, ứng dụng sẽ để người dùng đi vào vùng thường được giới hạn. Mô hình này trừu tượng hóa rất nhiều chi tiết và phù hợp trong trường hợp này vì việc đăng nhập vào một ứng dụng thường giống nhau trên tất cả các ứng dụng và không cần tuỳ biến gì nhiều. Mô hình này cho phép bạn tập trung vào những thành phần thực sự có thể cung cấp giá trị riêng biệt của ứng dụng.

```
class UserSession < Authlogic::Session::Base
  def to_key
    new_record??nil:[self.send(self.class.primary_key)]
  end
end
```

View – Khung nhìn

Đăng ký

The screenshot shows a web browser window with the URL http://noshster-spring-1.herokuapp.com/sign_up. The page is titled "Sign up". It contains three input fields: "Username" (placeholder: "noshster"), "Password" (placeholder: "password"), and "Email address" (placeholder: "noshster@noshster.com"). Below the fields is a blue "Sign up" button. The top of the browser window shows the title "NoshsterSignin" and the address bar with the URL.

Hình A.9

Trang đăng ký.

Khung nhìn này sẽ hiển thị form đăng ký như hình trên và thêm vào thông tin mở rộng như tiêu đề, ...

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

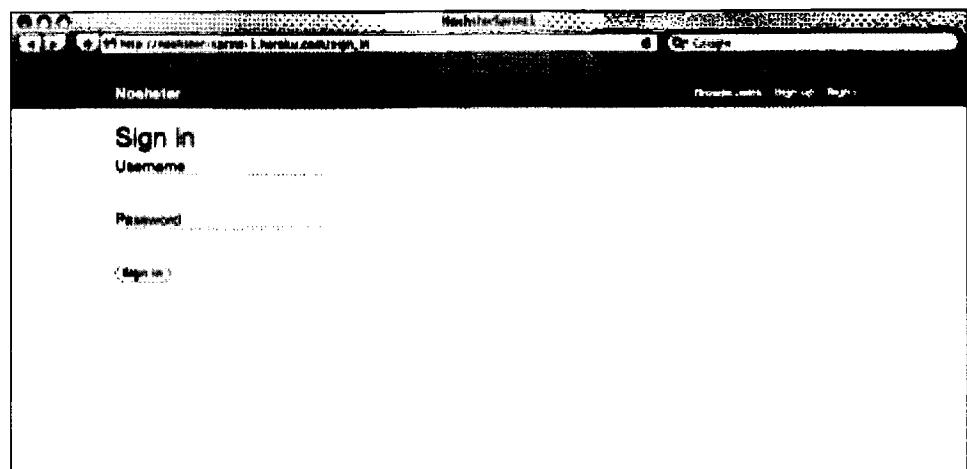
```
<h1>Sign up</h1>
<%= form_for @user do |f| %>
  <%= f.error_messages %>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Sign up' %>
<% end %>
```

Form partial (một form được sử dụng cho cả trang đăng ký và trang sửa hồ sơ)

Đây là form có thể tái sử dụng, hiển thị biểu mẫu đăng ký, người dùng có thể điền thông tin đăng ký tài khoản của mình vào đó. Trong mã nguồn của trang Đăng ký, dòng `<%= render :partial => 'form', :object => f %>` thể hiện việc sử dụng form này.

```
<p>
  <%= form.label :username %><br />
  <%= form.text_field :username %>
</p>
<p>
  <%= form.label :password %><br />
  <%= form.password_field :password %>
</p>
<p>
  <%= form.label :email_address %><br />
  <%= form.text_field :email_address %>
</p>
```

Đăng nhập



Hình A.10

Trang đăng nhập.

Khung nhìn này lấy từ form đăng ký, theo hướng cho phép người dùng sử dụng form để đăng nhập vào ứng dụng web.

```
<h1>Sign in</h1>
<%= form_for @user_session do |f| %>
  <%= f.error_messages %>
  <p>
    <%= f.label :username %><br />
    <%= f.text_field :username %>
  </p>
  <p>
    <%= f.label :password %><br />
    <%= f.password_field :password %>
  </p>
  <%= f.submit 'Sign in' %>
<% end %>
```

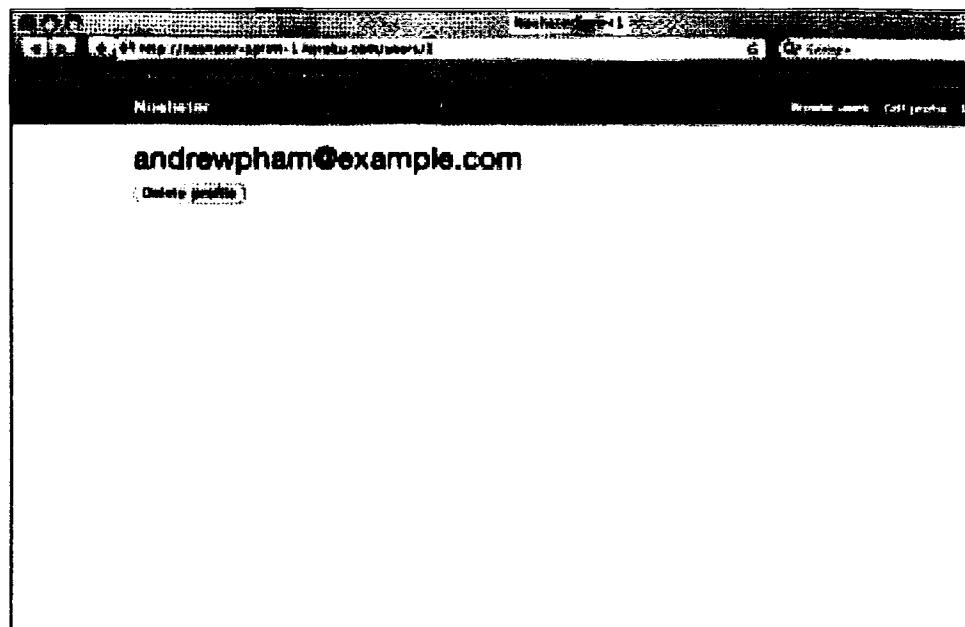
Trang đăng xuất



Hình A.11

Trang đăng xuất.

Xem/Xóa hồ sơ



Hình A.12

Trang xem/xóa hồ sơ.

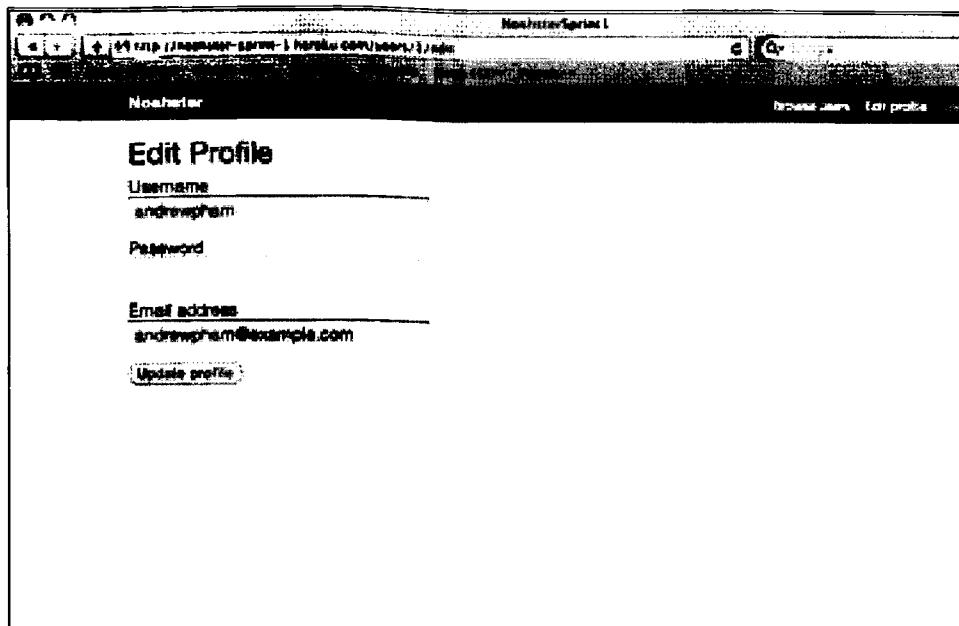
Khung nhìn này hiển thị thông tin hồ sơ liên quan tới người dùng và hiển thị một nút bấm cho phép người dùng xóa tài khoản của họ.

```
<h1><%= @user.email_address %></h1>
<% if @user == current_user %>
  <%= button_to 'Delete profile', @user, :method => :delete %>
<% end %>
```

Sửa hồ sơ

Khung nhìn này hiển thị form chỉnh sửa thông tin người dùng để người dùng có thể cập nhật thông tin của họ.

```
<h1>Edit Profile</h1>
<%= form_for @user do |f| %>
  <%= f.error_messages %>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Update profile' %>
<% end %>
```

**Hình A.13**

Trang sửa hồ sơ.

Form partial (một form được sử dụng cho cả trang đăng ký và trang sửa hồ sơ)

Đây là form có thể tái sử dụng, hiển thị biểu mẫu đăng ký, người dùng có thể điền thông tin đăng ký tài khoản của mình vào đó. Trong mã nguồn của trang Sửa hồ sơ, dòng `<%= render :partial => 'form', :object => f %>` thể hiện việc sử dụng form này.

```

<p>
  <%= form.label :username %><br />
  <%= form.text_field :username %>
</p>
<p>
  <%= form.label :password %><br />
  <%= form.password_field :password %>
</p>
<p>
  <%= form.label :email_address %><br />
  <%= form.text_field :email_address %>
</p>

```

Xóa hồ sơ



Hình A.14

Trang xóa hồ sơ.

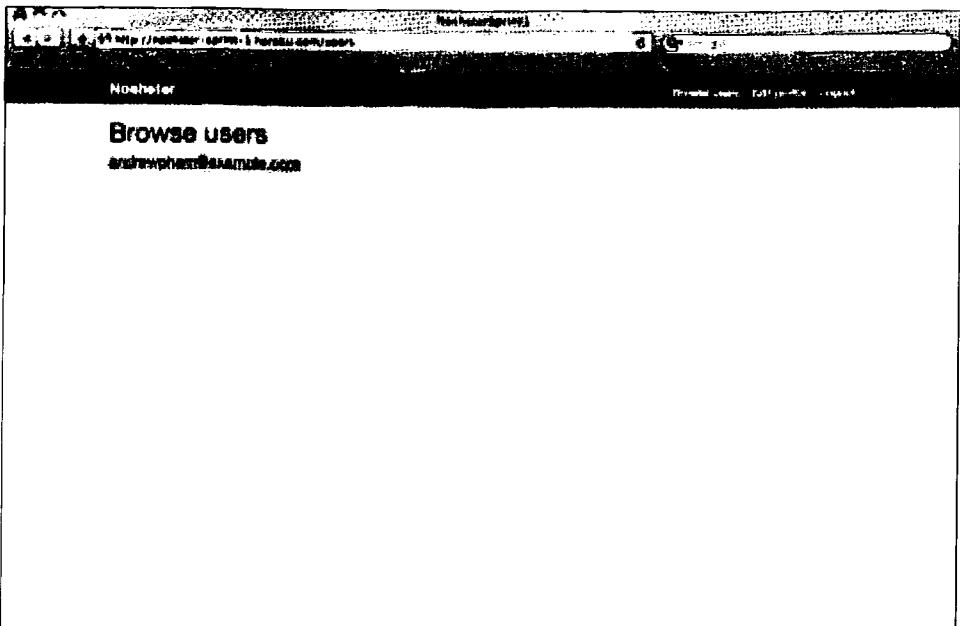
Khung nhìn này hiển thị hồ sơ cá nhân của người dùng và hiển thị một nút bấm cho phép họ xóa tài khoản của mình.

```
<h1><%= @user.email_address %></h1>
<% if @user==current_user %>
  <%= button_to 'Delete profile', @user, :method => :delete %>
<% end %>
```

Duyệt người dùng

Khung nhìn này nhận thông tin của tất cả người dùng được tàng điều khiển trả về từ cơ sở dữ liệu và hiển thị chúng.

```
<h1>Browse users</h1>
<%= render @user %>
```

**Hình A.15**

Trang duyệt người dùng.

User partial (tái sử dụng việc hiển thị chi tiết thông tin người dùng trên trang duyệt người dùng)

Đây là một phần của khung nhìn có thể tái sử dụng. Thành phần này được sử dụng cho trang web ở trên nhờ dòng mã `<%= render @users %>` nhằm hiển thị thông tin người dùng theo một cách cụ thể nào đó.

```
<div class='user'>
  <%= link_to user.email_address, user %>
</div>
```

Controller - Điều khiển

Điều khiển UserController nhận yêu cầu từ trình duyệt và trả về khung nhìn thích hợp cùng với dữ liệu cần thiết. Ví dụ, nếu người dùng đi đến trang đăng ký, điều khiển UserController sẽ trả về trang đăng ký cùng với một đối tượng User rỗng để người dùng điền thông tin vào sau. Trong một ví dụ khác, thực tế khi người dùng bấm vào nút đăng ký (Sign up), điều khiển sẽ nhận thông tin đã được nhập trên form, đưa thông tin vào một đối tượng User rỗng và lưu trữ thông tin (đây là lúc tầng mô hình - model sẽ tiếp nhận và xác thực tính hợp lệ của dữ liệu).

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
class UsersController < ApplicationController
  before_filter :require_no_user, :only => [:new, :create]
  before_filter :require_user, :only => [:edit, :update]

  # phương thức index yêu cầu mô hình User truy vấn bảng "users"
  # trong cơ sở dữ liệu (CSDL) và lấy tất cả các thông tin về người
  # dùng được sắp xếp theo email

  def index
    @users = User.order('email_address ASC').all
  end

  # phương thức new khởi tạo một đối tượng User mới, nhưng chưa
  # tạo và lưu vào CSDL.

  def new
    @user = User.new
  end

  # phương thức create tạo một đối tượng User mới, gán cho đối
  # tượng này với dữ liệu được người dùng nhập vào (params[:user]),
  # sau đó lưu thông tin. Hành động này có thể thất bại nếu
  # người dùng không nhập tất cả các dữ liệu cần thiết và hợp lệ
  # theo yêu cầu của mô hình hoặc nhập dữ liệu không chính xác.
  # Nếu lưu thông tin thành công, một thông báo sẽ được hiển thị
  # cho người dùng và chuyển họ tới một trang mới. Trường hợp ngược
  # lại, phương thức này sẽ gọi phương thức new và thông thường sẽ
  # hiển thị một form mới để người dùng thử lại.

  def create
    @user = User.new(params[:user])
    if @user.save
      flash[:notice] = 'Sign up successful!'
      redirect_to @user
    else
      render :new
    end
  end

  # phương thức show yêu cầu mô hình User truy vấn CSDL để lấy
  # thông tin về một người dùng cụ thể cho khung nhìn và khung nhìn
  # sẽ hiển thị thông tin chi tiết về người dùng này.

  def show
    @user = User.find(params[:id])
  end
```

```

# phương thức edit sử dụng một số chức năng được cung cấp bởi
# plugin Authlogic, plugin này lưu trữ thông tin về người dùng
# đang đăng nhập trong đối tượng @current_user. Sau đó phương
# thức này tạo ra một biến mới để chứa người dùng này. Khung nhìn
# của phương thức edit thường chứa một form đã hiển thị sẵn thông
# tin chi tiết về người dùng để dễ dàng cập nhật.

def edit
  @user = @current_user
end

# phương thức update lấy các dữ liệu trên form (params[:user])
# và tiến hành cập nhật thông tin chi tiết của người dùng hiện
# tại vào CSDL. Hành động này có thể thất bại nếu người dùng không
# cung cấp tất cả những thông tin hợp lệ cần cho mô hình User
# hoặc cung cấp dữ liệu không chính xác. Nếu thành công, một
# thông báo sẽ được hiển thị cho người dùng và phương thức sẽ
# chuyển người dùng tới một trang khác. Nếu thất bại, phương
# thức sẽ hiển thị lại form để người dùng thử lại.

def update
  @user = @current_user
  if @user.update_attributes(params[:user])
    flash[:notice] = 'Profile updated!'
    redirect_to @user
  else
    render :edit
  end
end

# phương thức destroy lấy người dùng hiện tại và xóa thông tin về
# người dùng này khỏi CSDL - do đó làm cho người dùng này không
# còn sử dụng ứng dụng được nữa.

def destroy
  @user = @current_user
  current_user_session.destroy
  @user.destroy
  redirect_to root_path
end

```

Điều khiển UserSessionsController tiếp nhận thông tin tài khoản được người dùng cung cấp trong form đăng nhập, rồi đưa thông tin vào trong một đối tượng UserSession rỗng và sau đó lưu thông tin này lại. Tiếp theo, UserSessionsController sẽ thực thi hành động cần thiết tùy thuộc vào việc đối tượng mô hình có thể tự lưu hay không (có nghĩa việc xác thực đã thành công). UserSessionsController cũng có

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

khả năng đăng xuất người dùng ra khỏi hệ thống và quay lại trang chủ.

```
class UserSessionsController < ApplicationController
  before_filter :require_no_user, :only => [:new,:create]
  before_filter :require_user, :only => :destroy

  # phương thức new khởi tạo một đối tượng UserSession mới
  def new
    @user_session = UserSession.new
  end

  # phương thức create khởi tạo một đối tượng UserSession mới và
  # gán cho đối tượng này các thông tin do người dùng nhập vào
  #(params[:user_session]). Sau đó, đối tượng thử lưu thông tin
  # của nó, quá trình này bao gồm việc xác thực các thông tin được
  # người dùng cung cấp và việc khớp những thông tin này với
  # dữ liệu được lưu trong CSDL. Nếu kết quả thành công, sẽ có một
  # thông báo hiển thị cho người dùng và chuyển họ tới trang dành
  # cho người dùng. Nếu thất bại, phương thức sẽ hiển thị lại form
  # để người dùng đăng nhập lại.

  def create
    @user_session = UserSession.new(params[:user_session])
    if @user_session.save
      flash[:notice] = 'Sign in successful!'
      redirect_back_or_default @user_session.user
    else
      render :new
    end
  end

  # phương thức destroy lấy thông tin từ session của người dùng
  # hiện tại và xóa các giá trị đã lưu trữ, về bản chất người dùng
  # đang đăng nhập sẽ bị đăng xuất khỏi hệ thống.

  def destroy
    current_user_session.destroy
    flash[:notice] = 'Logout successful!'
    redirect_to root_path
  end
end
```

Noshster Bản Phát hành 1—Sprint 2

Mục tiêu của Sprint 2 của bản phát hành 1 là xây dựng các chức năng quản lý món ăn dựa trên các chức năng quản lý người dùng đã được phát triển ở Sprint trước.

Các User story (Bản phát hành 1—Sprint 2)

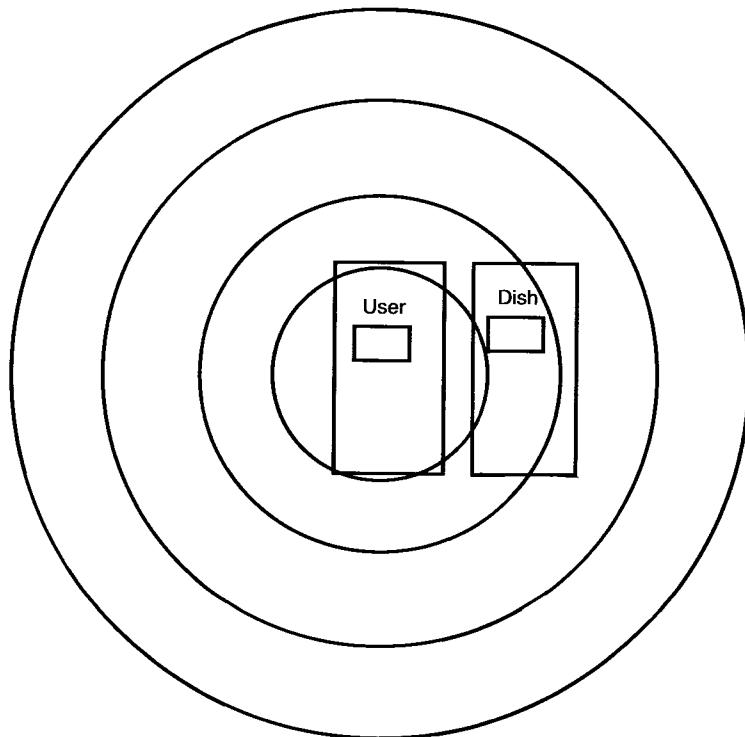
Những story được phát triển trong Sprint này là:

1. Thêm món ăn
2. Sửa món ăn
3. Xem món ăn
4. Duyệt món ăn

Như đã đề cập trong cuốn sách, một trong những lợi ích của việc tổ chức công việc phát triển xung quanh thành phần dữ liệu nghiệp vụ cốt lõi chung là cho phép các nhóm làm việc song song, đó cũng là cách đã làm với Noshster, chúng tôi chia các nhà phát triển thành ba nhóm riêng biệt để làm việc đồng thời cùng phát triển chức năng xem và chỉnh sửa các món ăn sau khi việc thực hiện user story thêm món ăn gần như đã hoàn tất.

Trong Hình A.16, bạn sẽ tìm thấy cách tổ chức mô hình dữ liệu cho Bản Phát hành 1 – Sprint 2.

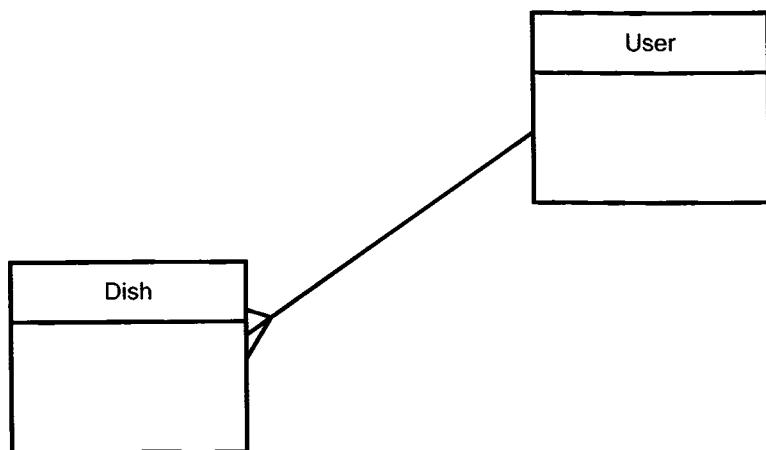
Trong Hình A.17, bạn sẽ tìm thấy quan hệ của mô hình dữ liệu cho Bản Phát hành 1 – Sprint 2.



Hình A.16

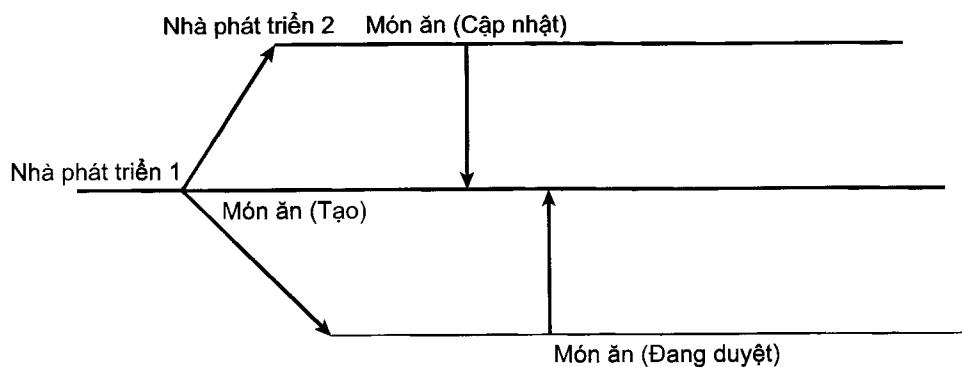
Mở rộng vòng dữ liệu đầu tiên.

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm



Hình A.17

Mô hình dữ liệu mức khái quát của Noshster trong Sprint thứ hai.



Hình A.18

Phát triển phần mềm Noshster song song.

Phát triển Phần mềm Noshster song song (đồng thời) (Bản phát hành 1 – Sprint 2)

Như đã nêu trong Chương 6 và 7, trên thực tế một trong rất nhiều ưu điểm của việc thiết kế và phát triển phần mềm xung quanh các yếu tố dữ liệu chung là sau khi thực thể được tạo ra, chúng ta có thể có hai hay nhiều nhóm làm việc song song (đồng thời) với nhau. Và điều đó đã được thực hiện cho Sprint 2 của ứng dụng Noshster, như có thể thấy trong Hình A.18. Phát triển phần mềm song song được thực hiện rất dễ dàng với việc sử dụng một hệ thống kiểm soát phiên bản, chẳng hạn như Git, từ đó các nhà phát triển có thể chỉ cần làm việc trên những phần khác nhau của một ứng dụng và hệ thống quản lý phiên bản này sẽ tự động ghép nối các thành phần này lại với nhau.

Mô hình-Khung nhìn-Điều khiển của ứng dụng Noshster (Bản phát hành 1—Sprint 2)

Mô hình Dish cho phép ứng dụng web Noshster lưu trữ thông tin món ăn do người dùng cung cấp. Mô hình này đảm bảo rằng tên và mô tả món ăn được cung cấp và tên các món ăn khác nhau. Mô hình Dish cũng chấp nhận một file và lưu trữ trên Amazon S3 trong khi kết nối tới bản ghi của món ăn trong cơ sở dữ liệu do đó sau này có thể tìm kiếm được.

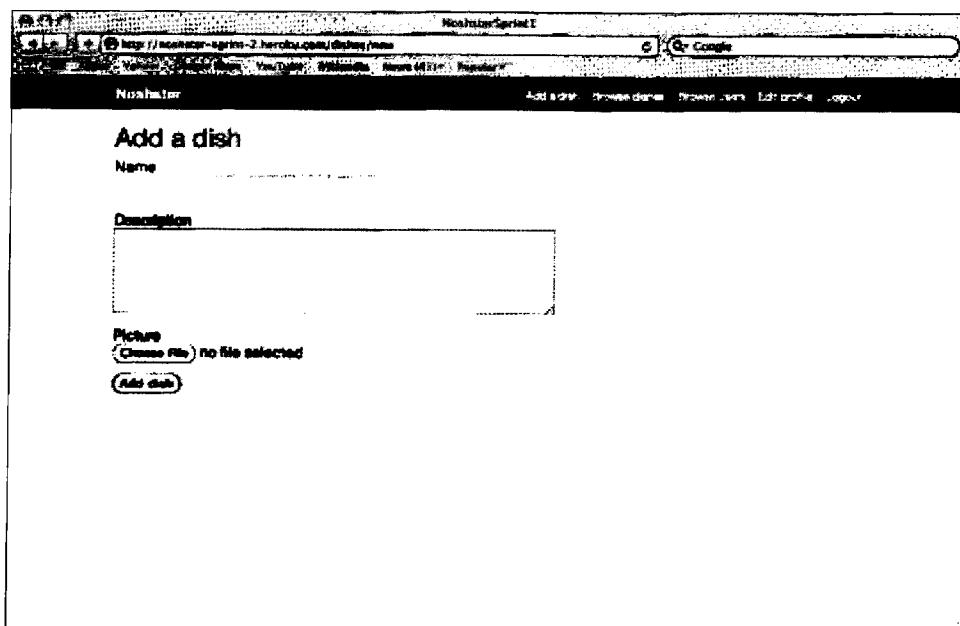
```
class Dish < ActiveRecord::Base
  validates :name, :presence => true, :uniqueness => true
  validates :description, :presence => true

  has_attached_file :picture, :styles => { :medium => '350x350#', :small =>
    '150x150#', :thumb => '75x75#' }, :storage => :s3, :s3_credentials =>
    "#{Rails.root}/config/s3.yml", :path => ":attachment/:id/:style.:extension"

  validates_attachment_presence :picture
end
```

Khung nhìn

Thêm món ăn



Hình A.19

Trang thêm món ăn.

Trang này cho phép người dùng thêm một món ăn bằng cách sử dụng form partial dưới đây và bổ sung thêm các thông tin như tiêu đề, ...

Phụ lục A - Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
<h1>Add a dish</h1>
<%= form_for @dish, :html => { :multipart => true } do |f| %>
  <%= f.error_messages %>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Add dish' %>
<% end %>
```

Form partial (form này được tái sử dụng cho trang thêm mới và cập nhật món ăn)

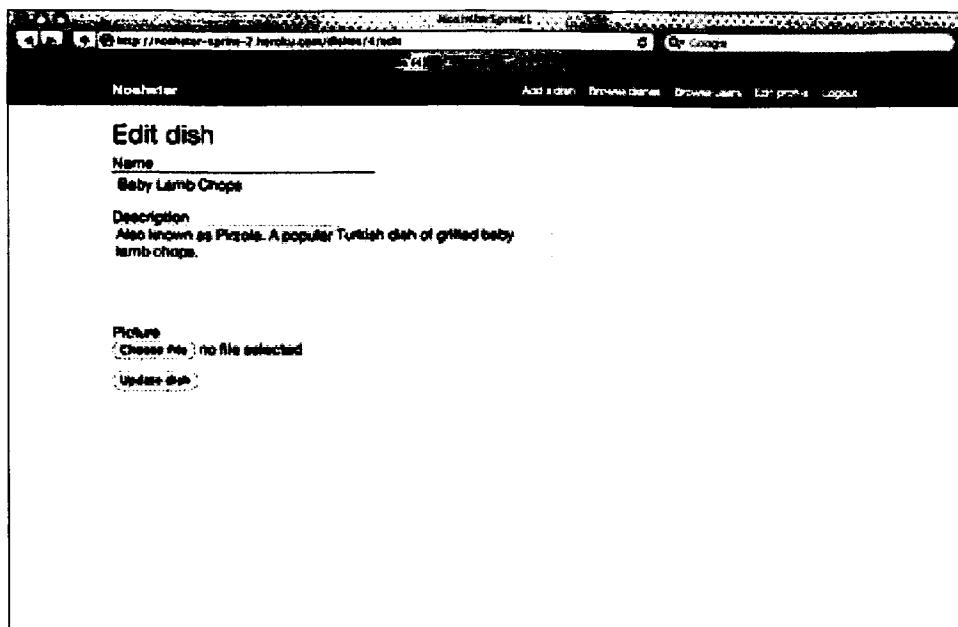
Đây là một thành phần có thể tái sử dụng, form partial này hiển thị một form cho phép người dùng nhập thông tin về món ăn để thêm mới hoặc cập nhật lại thông tin. Form partial được đưa vào nhờ dòng mã `<%= render :partial => 'form', :object => f %>` trong mã nguồn của trang Thêm món ăn.

```
<p>
  <%= form.label :name %><br />
  <%= form.text_field :name %>
</p>
<p>
  <%= form.label :description %><br />
  <%= form.text_area :description, :rows => 5, :cols => 50 %>
</p>
<p>
  <%= form.label :picture %><br />
  <%= form.file_field :picture %>
</p>
```

Sửa món ăn

Trang này cho phép người dùng cập nhật lại một món ăn bằng cách sử dụng form partial và bổ sung thêm các thông tin như tiêu đề, ...

```
<h1>Edit dish</h1>
<%= form_for @dish, :html => { :multipart => true } do |f| %>
  <%= f.error_messages %>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Update dish' %>
<% end %>
```

**Hình A.20**

Trang sửa món ăn.

Form partial (form này được tái sử dụng cho trang thêm và cập nhật lại món ăn)

Đây là một thành phần có khả năng tái sử dụng, form partial này hiển thị một form cho phép người dùng nhập thông tin về món ăn để thêm mới hoặc cập nhật lại thông tin. Form partial được đưa vào nhờ dòng mã `<%= render :partial => 'form', :object => f %>` trong mã nguồn của trang "Sửa món ăn".

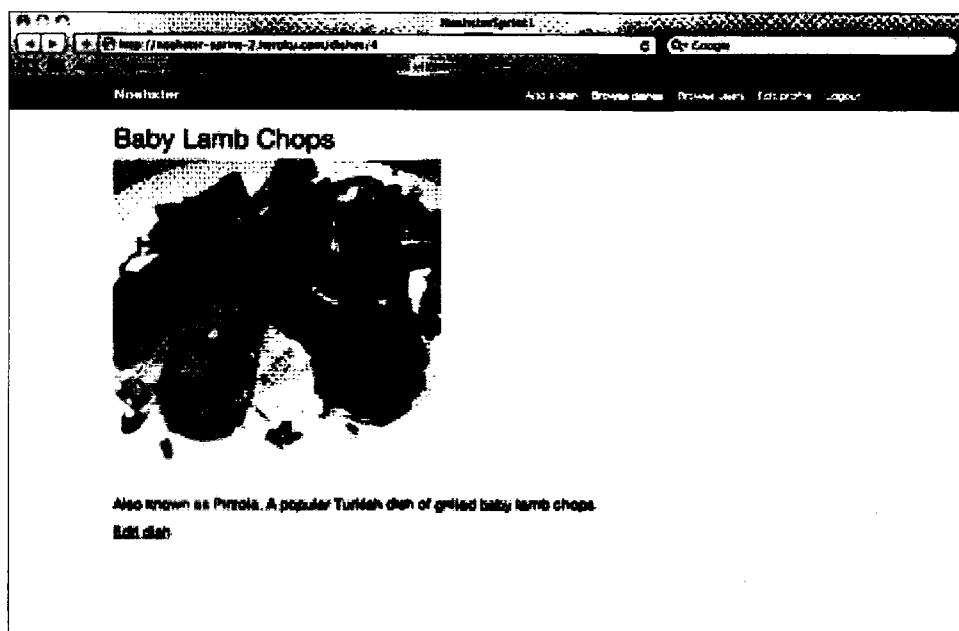
```

<p>
  <%= form.label :name %><br />
  <%= form.text_field :name %>
</p>
<p>
  <%= form.label :description %><br />
  <%= form.text_area :description, :rows => 5, :cols => 50 %>
</p>
<p>
  <%= form.label :picture %><br />
  <%= form.file_field :picture %>
</p>

```

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Xem món ăn



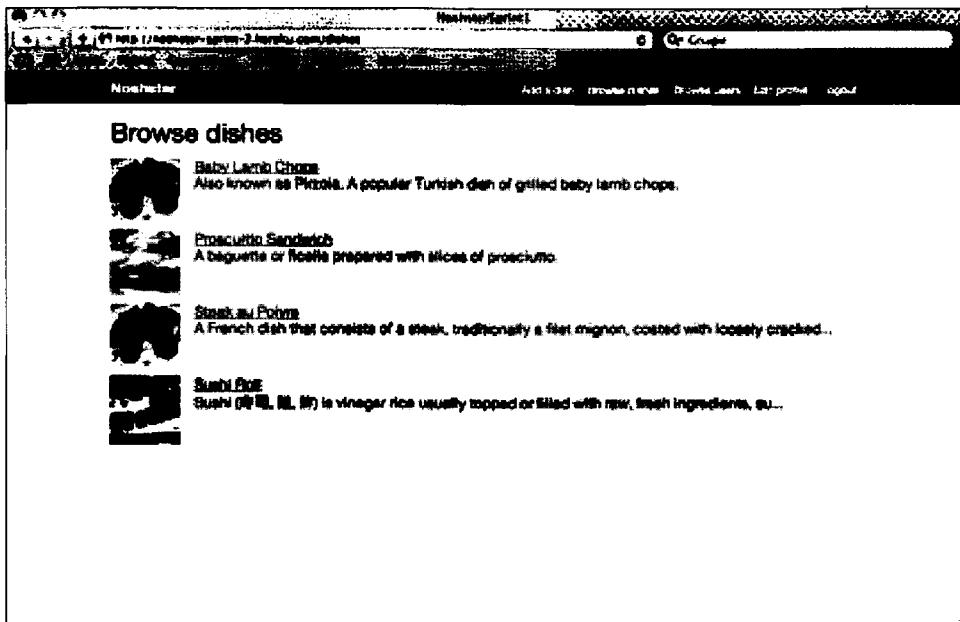
Hình A.21

Trang xem món ăn.

Trang này cho phép người dùng hiển thị thông tin món ăn bằng cách lấy đối tượng món ăn được trả về từ cơ sở dữ liệu và hiển thị đối tượng với định dạng cấu trúc như minh họa ở hình trên.

```
<h1><%= @dish.name %></h1>
<p>
  <%= image_tag @dish.picture.url(:medium), :alt => @dish.name %>
</p>
<p>
  <%= @dish.description %>
</p>
<% if signed_in? %>
  <p>
    <%= link_to 'Edit dish', edit_dish_path(@dish) %>
  </p>
<% end %>
```

Duyệt món ăn



Hình A.22

Trang duyệt món ăn.

Trang này lấy tất cả các món ăn được trả về từ cơ sở dữ liệu và hiển thị chúng cho người dùng.

```
<h1>Browse dishes</h1>
<%= render @dishes %>
```

Dish partial (để tái sử dụng cho việc hiển thị chi tiết các món ăn trên trang duyệt món ăn)

Partial này được sử dụng ở trang trước để hiển thị thông tin của một món ăn cụ thể. Thành phần này được đưa vào trang nhờ dòng mã `<%= render @dishes %>`.

```
<div class="dish">
  <div class="picture">
    <%= link_to image_tag(dish.picture.url(:thumb), :alt => dish.name), dish %>
  </div>
  <p class="details">
    <%= link_to dish.name, dish %><br />
    <%= truncate dish.description, :length => 100 %>
  </p>
</div>
```

Controller - Điều khiển

Điều khiển DishesController tiếp nhận yêu cầu từ người dùng rồi trả về khung nhìn và dữ liệu tương ứng từ cơ sở dữ liệu. Ví dụ, nếu người dùng chuyển tới trang duyệt các món ăn, điều khiển DishesController sẽ tìm kiếm tất cả các món ăn trong cơ sở dữ liệu và trả về cùng với khung nhìn duyệt món ăn tương ứng.

```
class DishesController < ApplicationController
  before_filter :require_user, :only => [:new, :create, :edit, :update]
  before_filter :get_dish, :only => [:show, :edit, :update]

  # phương thức index yêu cầu mô hình Dish thực hiện truy vấn vào
  # CSDL và lấy tất cả những món ăn sắp thứ tự theo tên.

  def index
    @dishes = Dish.order('name ASC')
  end

  # phương thức new khởi tạo một đối tượng Dish mới
  def new
    @dish = Dish.new
  end

  # phương thức create khởi tạo một đối tượng Dish mới và gán cho nó
  # những thông tin từ người dùng (params[:dish]). Sau đó đối tượng
  # này lưu thông tin của nó. Hành động này có thể thất bại nếu
  # người dùng không cung cấp tất cả những thông tin hợp lệ theo yêu
  # cầu của mô hình Dish hoặc cung cấp thông tin không chính xác.
  # Nếu thành công, một thông báo sẽ được hiển thị cho người dùng
  # và chuyển họ tới trang chi tiết về món ăn. Nếu thất bại, phương
  # thức sẽ hiển thị lại form để người dùng thử lại.

  def create
    @dish = Dish.new(params[:dish])
    if @dish.save
      flash[:notice] = 'You have successfully added a dish!'
      redirect_to @dish
    else
      render :new
    end
  end

  # phương thức show là phương thức đặc biệt của điều khiển này.
  # Ở đầu lớp DishesController có một dòng mã là
  # "before_filter :get_dish, :only => [:show, :edit, :update]", 
  # dòng mã này có nhiệm vụ gọi một filter (bộ lọc) thực thi phương
  # thức get_dish cho các phương thức show, edit, update. Phương
```

```

# thúc get_dish lấy một món ăn cụ thể từ CSDL để hiển thị thông tin.

def show
end

# phương thức edit về cơ bản cũng giống như phương thức show ở trên.

def edit
end

# phương thức update lấy món ăn có được từ phương thức get_dish
# và thử cập nhật món ăn với các dữ liệu do người dùng cung cấp
#(params[:dish]). Hành động này có thể thất bại nếu người dùng
# không cung cấp tất cả các giá trị hợp lệ theo yêu cầu của mô
# hình hoặc cung cấp các dữ liệu không chính xác. Nếu thành công,
# người dùng sẽ nhận được một thông báo và được chuyển tới trang
# thông tin chi tiết về món ăn. Nếu thất bại, phương thức sẽ hiển
# thị một form để người dùng thao tác lại.

def update
  if @dish.update_attributes(params[:dish])
    flash[:notice] = "You have successfully updated #{@dish.name}!"
    redirect_to @dish
  else
    render :edit
  end
end

private

# phương thức get_dish được tạo ra với mục đích tái sử dụng. Do
# một số phương thức của điều khiển này yêu cầu lấy món ăn từ
# CSDL, nên ta tạo riêng một phương thức cho hành động này thay vì
# phải sao chép mã nhiều lần.

def get_dish
  @dish = Dish.find(params[:id])
end

```

Noshster Bản phát hành 2—Sprint 3

Mục tiêu của Sprint thứ ba của bản phát hành 2 là xây dựng các chức năng quản lý nhà hàng dựa trên nền tảng các chức năng quản lý người dùng và món ăn đã được phát triển trong Sprint trước.

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm

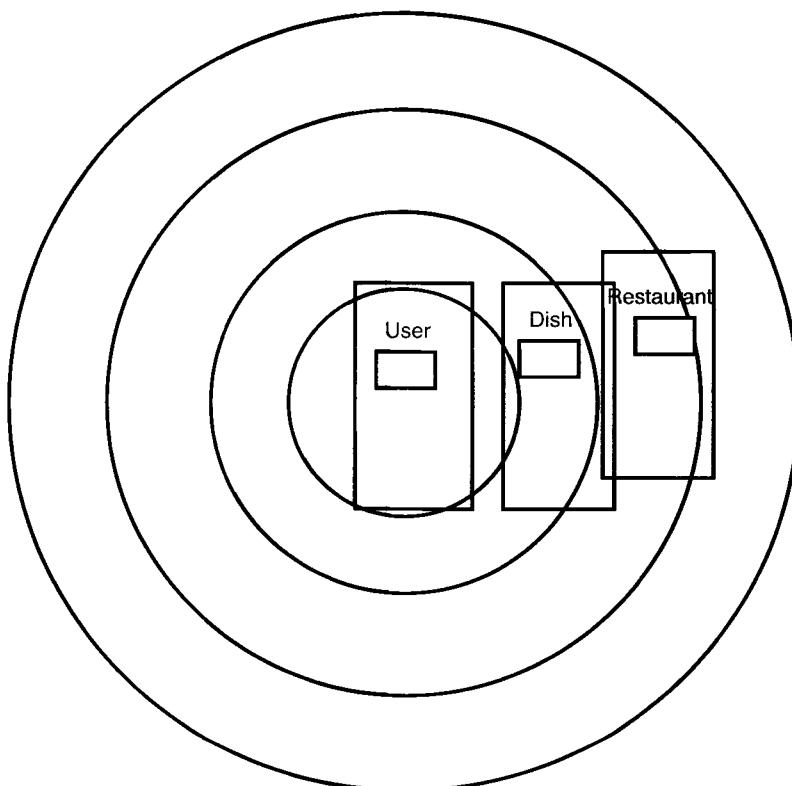
Các User story (Bản phát hành 2—Sprint 3)

Các story sẽ được phát triển trong Sprint này là:

1. Thêm nhà hàng
2. Sửa nhà hàng
3. Xem nhà hàng
4. Duyệt nhà hàng

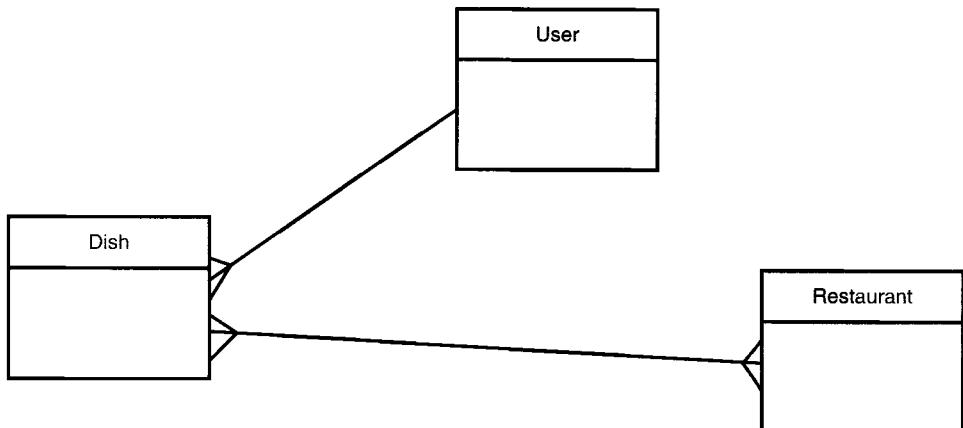
Phát triển các chức năng này trước sẽ giúp ích cho Sprint cuối cùng, Sprint cuối đời hỏi phải có các chức năng quản lý nhà hàng và món ăn đã sẵn sàng hoạt động tốt.

Trong Hình A.23, bạn sẽ tìm thấy cách thức tổ chức mô hình dữ liệu cho Bản phát hành 2—Sprint 3.



Hình A.23

Vòng dữ liệu chung trong Sprint 3.

**Hình A.24**

Mô hình dữ liệu của ứng dụng Noshster trong Sprint 3.

Trong Hình A.24, bạn sẽ tìm thấy quan hệ của mô hình dữ liệu cho Bản phát hành 2-Sprint 3.

Mô hình-Khung nhìn-Điều khiển của ứng dụng Noshster (Bản phát hành 2—Sprint 3)

Mô hình Restaurant cho phép ứng dụng web Noshster lưu trữ thông tin nhà hàng và đảm bảo rằng tên (name), địa chỉ (address), thành phố (city), trạng thái (state) và mã bưu điện (zip code) được truyền vào trong cơ sở dữ liệu. Mô hình Restaurant cũng đảm bảo việc mã bưu điện là dữ liệu số và nhà hàng không bị trùng lặp.

```

class Restaurant < ActiveRecord::Base
  validates :name, :presence => true
  validates :address_1, :presence => true
  validates :city, :presence => true
  validates :state, :presence => true
  validates :zip_code, :presence => true, :numericality => true
  validate :restaurant_is_not_duplicate

  private
  def restaurant_is_not_duplicate
    if Restaurant.where(:name => self.name, :address_1 => self.address_1).count > 0
      errors[:base] << 'That restaurant already exists.'
      return false
    end
  end
end
  
```

Phụ lục A ▷ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

View - Khung nhìn

Thêm nhà hàng

The screenshot shows a web page titled "Add a restaurant". The URL in the address bar is <http://neophytes-spring-3.herokuapp.com/restaurants/new>. The page contains several input fields: "Name" (with placeholder "Restaurant name"), "Address 1" (with placeholder "Address 1"), "Address 2" (with placeholder "Address 2"), "City" (with placeholder "City"), "State" (with placeholder "Select a state" and a dropdown menu showing "CA"), and "Zip code" (with placeholder "Zip code"). Below these fields is a button labeled "[Add reviews]". At the top of the page, there is a navigation bar with links for "Logout", "Edit profile", "Browse restaurants", "Browse users", "Browse classes", "Browse dashboards", "Add a class", and "Add a user". The top right corner of the browser window shows "On Google".

Hình A.25

Trang thêm nhà hàng.

Trang này cho phép người dùng thêm nhà hàng bằng cách sử dụng form partial và bổ sung thêm một số thông tin như tiêu đề, ...

```
<h1>Add a restaurant</h1>
<%= form_for @restaurant do |f| %>
  <%= f.error_messages %>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Add restaurant' %>
<% end %>
```

Form partial (form này được tái sử dụng cho các trang thêm nhà hàng và chỉnh sửa thông tin nhà hàng)

Đây là một thành phần có thể tái sử dụng, hiển thị một form cho phép người dùng nhập thông tin về nhà hàng cho chức năng thêm mới hoặc là chỉnh sửa thông tin nhà hàng. Form partial này được đưa vào trang Thêm Nhà hàng thông qua dòng mã `<%=render :partial => 'form', :object => f %>`.

```

<p>
  <%= form.label :name %><br />
  <%= form.text_field :name %>
</p>
<p>
  <%= form.label :address_1 %><br />
  <%= form.text_field :address_1 %>
</p>
<p>
  <%= form.label :address_2 %><br />
  <%= form.text_field :address_2 %>
</p>
<p>
  <%= form.label :city %><br />
  <%= form.text_field :city %>
</p>
<p>
  <%= form.label :state %><br />
  <%= select :restaurant, :state, ['[List of states]'] %>
</p>
<p>
  <%= form.label :zip_code %><br />
  <%= form.text_field :zip_code %>
</p>

```

Sửa nhà hàng

Trang này cho phép người dùng chỉnh sửa thông tin nhà hàng bằng cách sử dụng form partial và bổ sung thêm một số thông tin như tiêu đề, ...

```

<h1>Edit restaurant</h1>
<%= form_for @restaurant do |f| %>
  <%= f.error_messages %>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Update restaurant' %>
<% end %>

```

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm

The screenshot shows a web page titled "Edit restaurant". The URL is "http://localhost:8000/restaurant/1/edit". The page has a header with links for "Home", "About", "Contact", "Logout", and "Edit profile". The main content area is titled "Edit restaurant". It contains several input fields:

- Name: (with a red asterisk)
- Category: (with a red asterisk)
- Address 1:** 100 North Central Expressway
- Address 2:** _____
- City: _____
- State: (with a red asterisk)
- Zip code: 78071
- Update reviews

Hình A.26

Trang chỉnh sửa nhà hàng.

Form partial (form này được tái sử dụng cho cả trang thêm và chỉnh sửa nhà hàng)

Đây là một thành phần có thể tái sử dụng, hiển thị một form cho phép người dùng nhập thông tin về nhà hàng cho chức năng thêm mới hoặc là chỉnh sửa thông tin nhà hàng. Form partial này được đưa vào trang Sửa nhà hàng thông qua dòng mã `<%= render :partial => 'form', :object => f %>`.

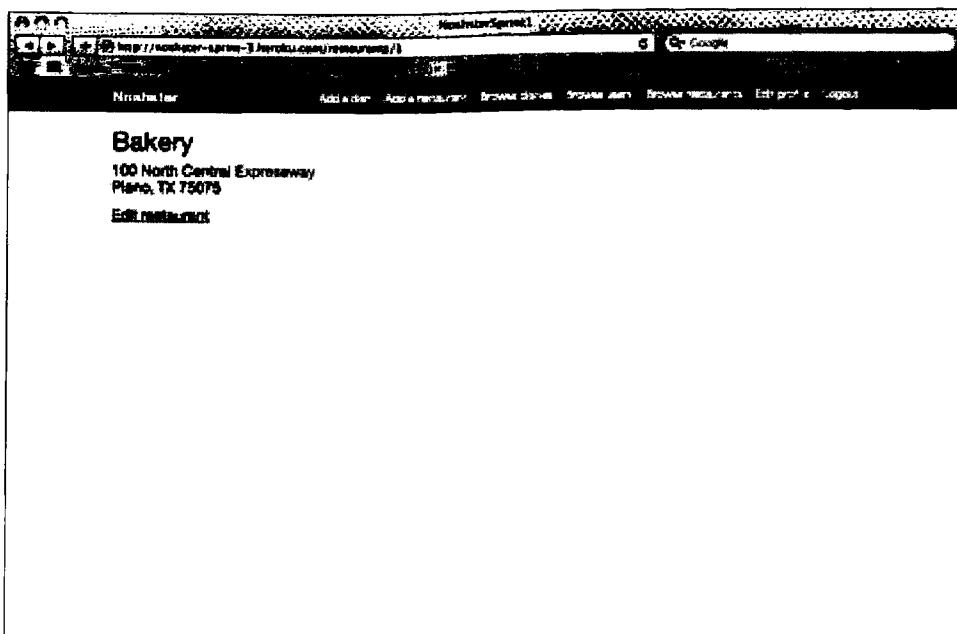
```
<p>
  <%= form.label :name %><br />
  <%= form.text_field :name %>
</p>
<p>
  <%= form.label :address_1 %><br />
  <%= form.text_field :address_1 %>
</p>
<p>
  <%= form.label :address_2 %><br />
  <%= form.text_field :address_2 %>
</p>
<p>
```

```

<%= form.label :city %><br />
<%= form.text_field :city %>
</p>
<p>
  <%= form.label :state %><br />
  <%= select :restaurant, :state, ['[List of states]'] %>
</p>
<p>
  <%= form.label :zip_code %><br />
  <%= form.text_field :zip_code %>
</p>

```

Xem nhà hàng



Hình A.27

Trang xem nhà hàng.

Trang này cho phép người dùng xem thông tin về nhà hàng bằng cách lấy đối tượng nhà hàng được trả về từ cơ sở dữ liệu và hiển thị đối tượng với định dạng như hình minh họa ở trên.

```

<h1><%= @restaurant.name %></h1>
<p>
  <%= @restaurant.address_1 %><br />
  <% unless @restaurant.address_2.empty? %>

```

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
<%= @restaurant.address_2 %><br />
<% end %>
<%= @restaurant.city %>, <%= @restaurant.state %> <%= @restaurant.zip_code %>
</p>
<% if signed_in? %>
<p>
  <%= link_to 'Edit restaurant', edit_restaurant_path(@restaurant) %>
</p>
<% end %>
```

Duyệt nhà hàng



Hình A.28

Trang duyệt nhà hàng.

Trang này lấy tất cả nhà hàng được trả về từ cơ sở dữ liệu và hiển thị chúng cho người dùng.

```
<h1>Browse restaurants</h1>
<%= render @restaurants %>
```

Restaurant partial (nhằm tái sử dụng phần hiển thị chi tiết nhà hàng trên trang duyệt nhà hàng)

Đây là một phần của trang ở trên được sử dụng để hiển thị thông tin của một nhà hàng cụ thể. Restaurant partial được đưa vào trang nhờ dòng mã `<%= render @restaurants %>`.

```
<div class = "restaurant">
  <%= link_to restaurant.name, restaurant %>
</div>
```

Controller - Điều khiển Điều khiển RestaurantsController nhận yêu cầu từ người dùng rồi trả về khung nhìn và dữ liệu tương ứng từ cơ sở dữ liệu. Ví dụ, nếu người dùng đi tới trang duyệt nhà hàng thì điều khiển RestaurantsController sẽ tìm kiếm tất cả nhà hàng trong cơ sở dữ liệu và trả về cùng với khung nhìn duyệt nhà hàng tương ứng.

```
class RestaurantsController < ApplicationController
  before_filter :require_user, :only => [:new, :create, :edit, :update]
  before_filter :get_restaurant, :only => [:show, :edit, :update]

  # phương thức index yêu cầu mô hình Restaurant truy vấn bảng
  # "restaurants" trong CSDL và lấy tất cả các nhà hàng sắp thứ
  # tự theo tên.

  def index
    @restaurants = Restaurant.order('name ASC')
  end

  # phương thức new khởi tạo một đối tượng Restaurant (nhà hàng) mới.
  def new
    @restaurant = Restaurant.new
  end

  # phương thức create khởi tạo một đối tượng Restaurant mới,
  # truyền cho đối tượng này dữ liệu do người dùng nhập vào
  #(params[:restaurant]) và sau đó thủ lưu thông tin của nó. Hành
  # động này có thể thất bại nếu người dùng không cung cấp tất cả
  # các thông tin hợp lệ theo yêu cầu của mô hình Restaurant hoặc
  # cung cấp thông tin không chính xác. Nếu lưu thông tin thành
  # công, người dùng sẽ nhận được một thông báo và được chuyển tới
  # một trang mới. Nếu thất bại, phương thức sẽ gọi phương thức
  # new, phương thức thường có nhiệm vụ tạo một form để người dùng
  # thử nhập lại thông tin.

  def create
    @restaurant = Restaurant.new(params[:restaurant])
    if @restaurant.save
      flash[:notice] = 'You have successfully added a restaurant!'
      redirect_to @restaurant
    else
      render :new
    end
  end
end
```

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
# phương thức show là một phương thức đặc biệt của điều khiển này.  
# Ở phần đầu của điều khiển RestaurantsController, có dòng mã  
# "before_filter :get_restaurant, :only => [:show, :edit, :update]".  
# Dòng mã này để gọi một filter (bộ lọc) dành cho các phương thức  
# show, edit và update. Filter này sẽ thực thi phương thức  
# get_restaurant để lấy một nhà hàng cụ thể nào đó từ CSDL cho  
# việc hiển thị thông tin.  
  
def show  
end  
  
# phương thức edit về cơ bản cũng giống như phương thức show  
  
def edit  
end  
  
# phương thức update lấy đối tượng nhà hàng có được từ phương thức  
# get_restaurant và thử cập nhật thông tin nhà hàng này bằng dữ  
# liệu do người dùng cung cấp (params[:restaurant]). Hành động này  
# có thể thất bại nếu người dùng không cung cấp tất cả những dữ  
# liệu hợp lệ theo yêu cầu của mô hình Restaurant hoặc cung cấp những  
# dữ liệu không chính xác. Nếu thành công, người dùng sẽ nhận được  
# một thông báo và được chuyển tới trang chi tiết về nhà hàng. Nếu  
# thất bại, phương thức sẽ hiển thị form để người dùng thử lại.  
  
def update  
  if @restaurant.update_attributes(params[:restaurant])  
    flash[:notice] = "You have successfully updated #{@restaurant.name}!"  
    redirect_to @restaurant  
  else  
    render :edit  
  end  
end  
  
private  
  
# phương thức get_restaurant được tạo ra nhằm mục đích tái sử dụng.  
# Do một số phương thức trong điều khiển này đòi hỏi thông tin về  
# một nhà hàng từ CSDL nên có thể tạo riêng hành động này trong một  
# phương thức thay vì phải sao chép nhiều lần một đoạn mã.  
  
def get_restaurant  
  @restaurant = Restaurant.find(params[:id])  
end  
end
```

Noshster Bản phát hành 2—Sprint 4

Mục tiêu của Sprint 4 - bản phát hành 2 là xây dựng các chức năng quản lý thông tin đánh giá dựa trên nền tảng các chức năng quản lý nhà hàng đã được phát triển ở Sprint trước.

Các User story (Bản phát hành 2—Sprint 4)

Các story sẽ được phát triển trong suốt Sprint này là:

1. Thêm đánh giá
2. Sửa đánh giá
3. Xem đánh giá

Sau khi phát triển tất cả các chức năng khác nhau trong Sprint trước, giờ đây chúng ta có thể phát triển “cây” chức năng cuối cùng.

Trong Hình A.29, bạn sẽ tìm thấy cách tổ chức mô hình dữ liệu cho Bản phát hành 2 – Sprint 4.

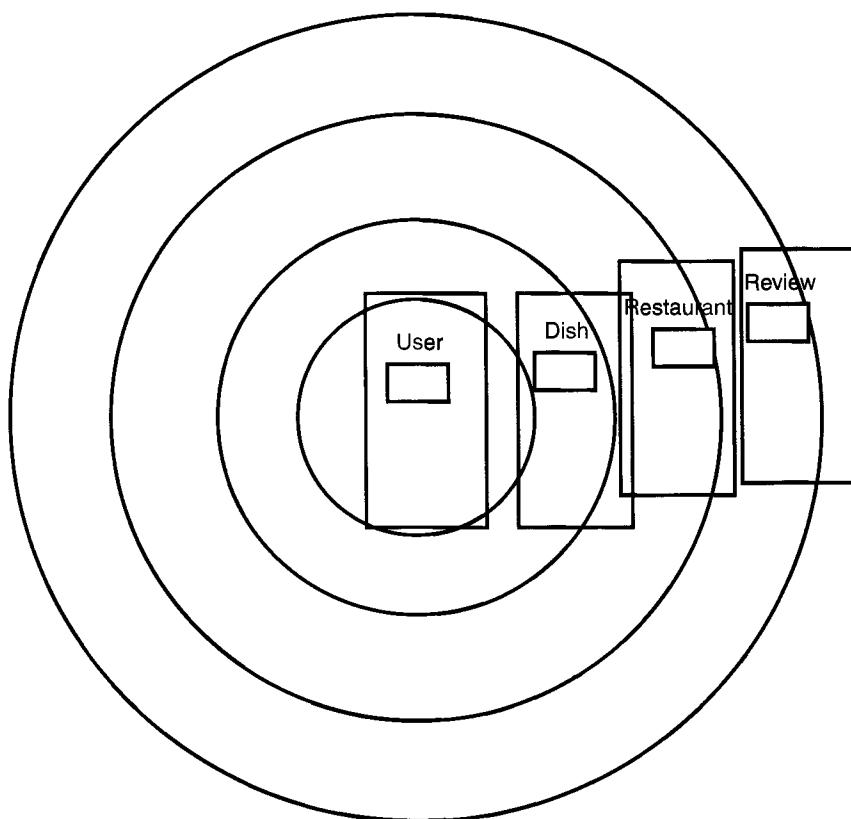
Trong Hình A.30, bạn sẽ tìm thấy quan hệ của mô hình dữ liệu cho Phát hành 2 – Sprint 4.

Mô hình-Khung nhìn-Điều khiển của ứng dụng Noshster (Bản phát hành 2—Sprint 4)

Mô hình Review cho phép ứng dụng web Noshster lưu trữ thông tin đánh giá của người dùng về món ăn và nhà hàng trong cơ sở dữ liệu để báo cáo sau này. Mô hình này đảm bảo rằng mỗi đánh giá sẽ bao gồm phần nội dung (body), điểm số đánh giá (rating) và nhà hàng (restaurant).

```
class Review < ActiveRecord::Base
  validates :body, :presence => true
  validates :rating, :presence => true
  validates :restaurant_id, :presence => true
  belongs_to :user
  belongs_to :dish
  belongs_to :restaurant
end
```

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

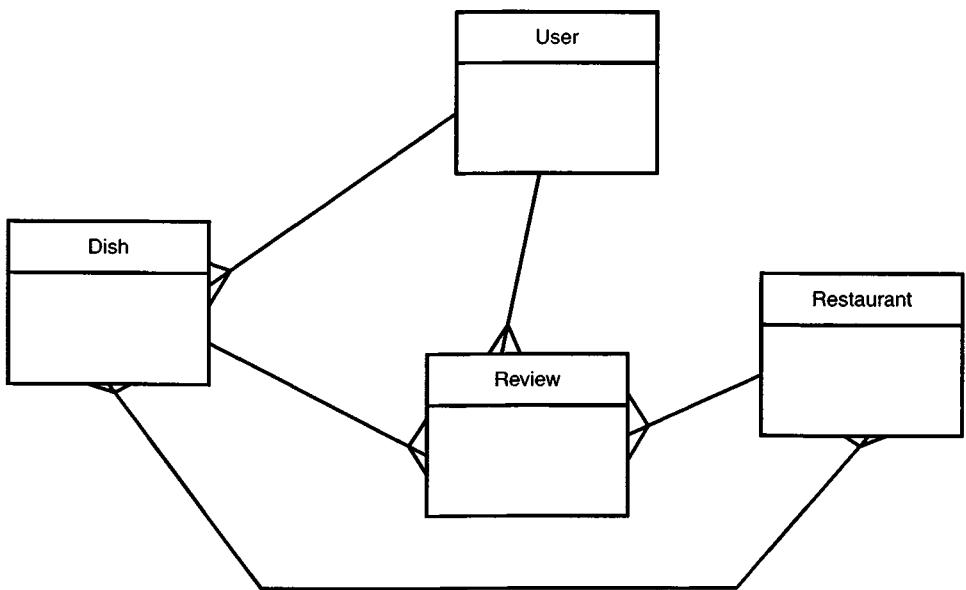


Hình A.29

Các vòng dữ liệu trong Sprint 4.

Mô hình User được chỉnh sửa để tạo ra mối quan hệ giữa người dùng và đánh giá – đó là một người dùng có nhiều đánh giá, do đó khi đánh giá được hiển thị, chúng ta cần hiển thị cả thông tin người đã viết đánh giá.

```
class User < ActiveRecord::Base
  acts_as_authentic do |c|
    c.require_password_confirmation = false
  end
  has_many :reviews
end
```

**Hình A.30**

Mô hình dữ liệu Noshster trong Sprint 4.

Mô hình Dish được chỉnh sửa nhằm tạo ra một mối quan hệ giữa các món ăn, đánh giá và nhà hàng để khi lấy một món ăn từ cơ sở dữ liệu, bạn cũng có thể xem tất cả các đánh giá và nhà hàng có liên quan.

```

class Dish < ActiveRecord::Base
  validates :name, :presence => true, :uniqueness => true
  validates :description, :presence => true
  has_attached_file :picture, :styles => { :medium => '350x350#',
    :small => '150x150#', :thumb => '75x75#' }, :storage => :s3,
  :s3_credentials => "#{Rails.root}/config/s3.yml", :path =>
  ":attachment/:id/:style.:extension"
  validates_attachment_presence :picture
  has_many :reviews
  has_many :restaurants, :through => :reviews
end
-----
```

Mô hình Restaurant được chỉnh sửa để kết nối đánh giá và món ăn vì những lý do tương tự như mô hình Dish.

```

class Restaurant < ActiveRecord::Base
  validates :name, :presence => true
  validates :address_1, :presence => true
  
```

Phụ lục A - Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
validates :city, :presence => true
validates :state, :presence => true
validates :zip_code, :presence => true, :numericality => true
validate :restaurant_is_not_duplicate

has_many :reviews
has_many :dishes, :through => :reviews

private
def restaurant_is_not_duplicate
    if Restaurant.where(:name => self.name, :address_1 =>
    self.address_1).count > 0
        errors[:base]<<"That restaurant already exists."
        return false
    end
end
end
```

Khung nhìn

Thêm đánh giá

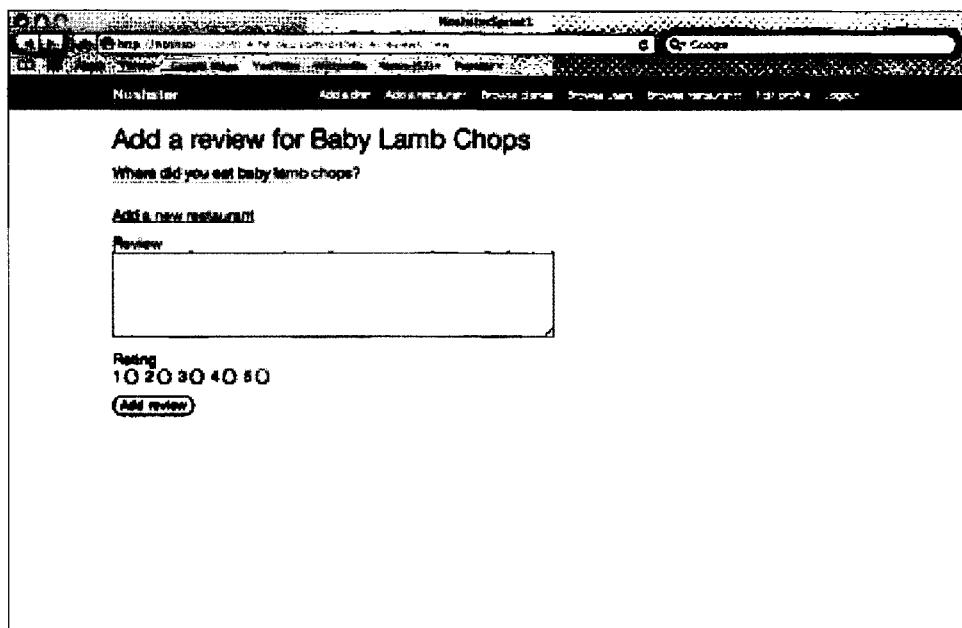
Trang này cho phép người dùng thêm một đánh giá bằng cách sử dụng form partial có trước đó và bổ sung thêm một số thông tin như tiêu đề, ...

```
<% content_for :head do %>
<%= javascript_include_tag '/restaurants' %>
<script type="text/javascript">
$(document).ready(function(){
    $('#restaurant_name').autocomplete(data, {
        matchContains:true,
        mustMatch:true,
        formatItem:function(item){
            return item.name+ '<br />' + item.address;
        },
        formatMatch:function(item){
            return item.name;
        },
        formatResult:function(item){
            return item.name;
        }
    }).result(function(event, item){
        $('#review_restaurant_id').attr('value', item.id);
    });
});
```

```

    });
</script>
<% end %>
<h1>Add a review for <%= @dish.name %></h1>
<%= form_for [@dish, @review] do |f| %>
  <%= f.error_messages %>
  <p>
    <%= label_tag "Where did you eat #{@dish.name}?" %><br />
    <%= text_field_tag :restaurant_name, params[:restaurant_name] %><br />
    <%= link_to 'Add a new restaurant', new_restaurant_path %>
    <%= f.hidden_field :restaurant_id %>
  </p>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Add review' %>
<% end %>

```

**Hình A.31**

Trang thêm đánh giá.

Form partial (form này được tái sử dụng cho trang thêm và sửa đánh giá)

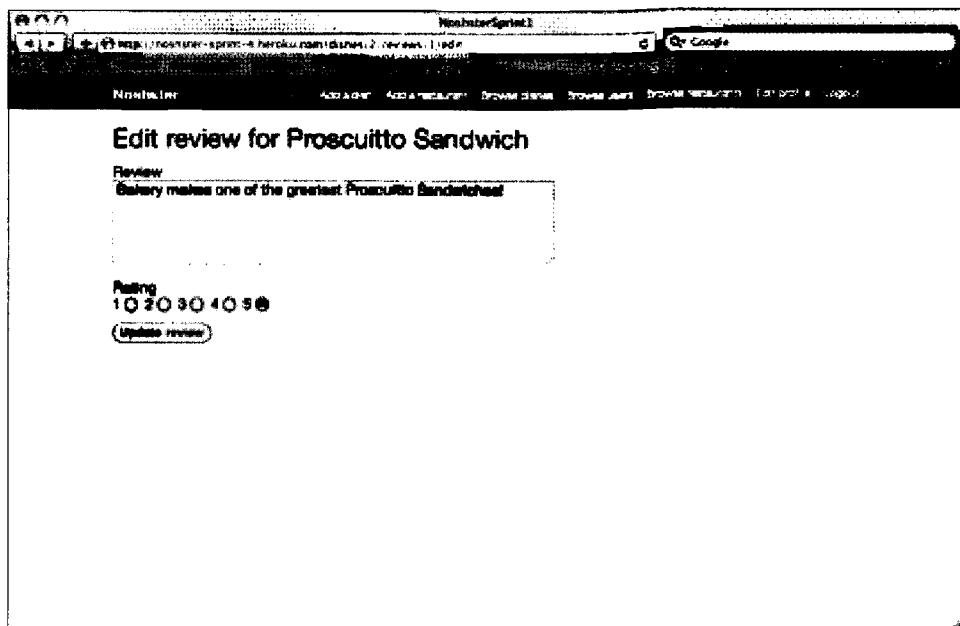
Đây là thành phần có thể tái sử dụng, hiển thị một form cho phép người dùng nhập thông tin đánh giá mới hoặc chỉnh sửa thông tin đánh giá về món ăn. Form partial này

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

được đưa vào trang “Thêm Đánh giá” nhờ dòng mã <%= render :partial => ‘form’, :object => f %>.

```
<p>
<%= form.label :body, 'Review' %><br />
<%= form.text_area :body,:rows => 5, :cols => 50 %>
</p>
<p>
<%= form.label :rating %><br />
1 &nbsp;<%= form.radio_button :rating, 1 %>&nbsp;
2 &nbsp;<%= form.radio_button :rating, 2 %>&nbsp;
3 &nbsp;<%= form.radio_button :rating, 3 %>&nbsp;
4 &nbsp;<%= form.radio_button :rating, 4 %>&nbsp;
5 &nbsp;<%= form.radio_button :rating, 5 %>&nbsp;
</p>
```

Chỉnh sửa đánh giá



Hình A.32

Trang chỉnh sửa đánh giá.

Trang này cho phép người dùng chỉnh sửa đánh giá bằng cách sử dụng form partial có trước đó và bổ sung thêm một số thông tin như tiêu đề, ...

```
<h1>Edit review for <%= @dish.name %></h1>
<%= form_for [@dish, @review] do |f| %>
```

```
<%= f.error_messages %>
<%= render :partial => 'form', :object => f %>
<%= f.submit 'Update review' %>
<% end %>
```

Form partial (form này được tái sử dụng cho trang thêm đánh giá và sửa đánh giá)

Đây là một thành phần có thể tái sử dụng, hiển thị một form cho phép người dùng nhập thông tin đánh giá mới hoặc chỉnh sửa thông tin đánh giá. Form partial này được đưa vào trang “Chỉnh sửa Đánh giá” nhờ dòng mã `<%= render :partial => 'form', :object => f %>`.

```
<p>
  <%= form.label :body, 'Review' %><br />
  <%= form.text_area :body, :rows => 5, :cols => 50 %>
</p>
<p>
  <%= form.label :rating %><br />
  1 <%= form.radio_button :rating, 1 %>&nbsp;
  2 <%= form.radio_button :rating, 2 %>&nbsp;
  3 <%= form.radio_button :rating, 3 %>&nbsp;
  4 <%= form.radio_button :rating, 4 %>&nbsp;
  5 <%= form.radio_button :rating, 5 %>&nbsp;
</p>
```

Review partial/xóa đánh giá

Partial này hiển thị các đánh giá theo định dạng cụ thể cùng với nút xóa đánh giá (Delete review) nhằm cho phép người dùng xóa đánh giá của họ.

```
<div class="review">
  <p>
    <%= link_to review.user.email_address, review.user %>
    gave <%= link_to review.restaurant.name, review.restaurant %><%= review.rating %> stars for this dish!
  </p>
  <p>
    <%= review.body %>
  </p>
  <p>
    <% if signed_in? %>
      <% if review.user == current_user %>
        <%= link_to 'Edit review', edit_dish_review_path(review.dish,
review) %>
    <% end %>
  </p>
```

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
<%= button_to 'Delete review', dish_review_path(review.dish,
review), :method => :delete %>
<% end %>
<% end %>
</p>
</div>
```



Hình A.33

Trang xóa đánh giá.

Controller - Điều khiển Điều khiển ReviewsController nhận yêu cầu từ người dùng rồi trả về khung nhìn tương ứng chứa dữ liệu từ cơ sở dữ liệu. Ví dụ, nếu người dùng đi đến trang thêm đánh giá, thì điều khiển ReviewsController sẽ tìm món ăn và nhà hàng liên quan trong cơ sở dữ liệu và một đối tượng đánh giá trống để điền vào sau này và trả về cùng với khung nhìn thêm đánh giá tương ứng.

```
class ReviewsController < ApplicationController
  before_filter :require_user, :only => [:new, :create, :edit, :update, :destroy]
  before_filter :get_dish
  before_filter :get_review, :only => [:edit, :update, :destroy]
  before_filter :authorize_user, :only => [:edit, :update, :destroy]
  # phương thức new sử dụng Authlogic được xây dựng sẵn trong
  # phương thức current_user để lấy người dùng đang đăng nhập và
  # khởi tạo một đối tượng Review.
```

```

def new
  store_location
  @user = current_user
  @review = @user.reviews.build
end

# phương thức create lấy người dùng đang đăng nhập và khởi tạo
# một đối tượng Review với những dữ liệu do người dùng cung cấp
# (params[:review]). Đối tượng Review này khởi tạo đối tượng
# cha là món ăn (Dish) lấy được từ CSDL thông qua phương thức
# get_dish. Sau đó đối tượng này thử lưu thông tin của nó. Điều
# này có thể thất bại nếu người dùng không cung cấp tất cả các
# dữ liệu hợp lệ theo yêu cầu của mô hình Review hoặc cung cấp
# thông tin không chính xác. Nếu thành công, người dùng sẽ nhận
# được một thông báo và được chuyển tới trang chi tiết về món ăn
# với các đánh giá về món ăn này. Nếu thất bại, phương thức sẽ
# hiển thị form để người dùng thử lại.

def create
  @user = current_user
  @review = @user.reviews.build(params[:review])
  @review.dish = @dish
  if @review.save
    flash[:notice] = "You have successfully reviewed #{@dish.name}!"
    redirect_to @dish
  else
    render :new
  end
end

# phương thức edit sử dụng filter before_filter khai báo ở phần
# đầu của điều khiển, filter này thực thi phương thức get_review
# để cung cấp đối tượng Review cho khung nhìn.

def edit
end

# phương thức update sử dụng filter before_filter khai báo ở phần
# đầu của điều khiển. Filter này thực thi phương thức get_review
# để lấy một đánh giá, sau đó đánh giá này thử cập nhật các dữ liệu
# do người dùng cung cấp và thử lưu thông tin của nó. Hành động
# này có thể thất bại nếu người dùng không cung cấp tất cả các dữ
# liệu hợp lệ theo yêu cầu của mô hình Review hoặc cung cấp dữ
# liệu không chính xác. Nếu thành công, người dùng sẽ nhận được
# một thông báo và được chuyển tới trang chi tiết về món ăn. Nếu
# thất bại, phương thức sẽ hiển thị form để người dùng thử lại.

```

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
def update
  if @review.update_attributes(params[:review])
    flash[:notice] = "You have successfully updated your review!"
    redirect_to @dish
  else
    render :edit
  end
end

# phương thức destroy cũng sử dụng before_filter khai báo ở phần
# đầu của điều khiển, filter này thực thi phương thức get_review để
# lấy một đánh giá cụ thể trước khi thủ xóa đánh giá này khỏi CSDL.

def destroy
  @review.destroy
  flash[:notice] = 'You have successfully deleted your review!'
  redirect_to @dish
end

private
def get_dish
  @dish = Dish.find(params[:dish_id])
end

def get_review
  @review = Review.find(params[:id])
end

def authorize_user
  unless @review.user == current_user
    flash[:notice] = 'You do not have permission to perform this action.'
    redirect_to current_user
  end
end
end
```

Kiểm thử Noshster

Ví dụ về kiểm thử đơn vị

Ví dụ về kiểm thử đơn vị cho mô hình Dish:

```
Class DishTest < ActiveSupport::TestCase
  test 'Dish is successfully created' do
    old_count = Dish.count
    Dish.create(:name => 'Example', :description => 'Example',
```

```

:picture => 'Example.jpg')
    new_count = Dish.count
    assert_equal(new_count, old_count+1) # Đảm bảo rằng đã có thêm một
                                         # món ăn được lưu vào CSDL.
end
end

```

Bản thân các kiểm thử đảm bảo rằng bạn có thể thêm thành công một món ăn vào cơ sở dữ liệu với tất cả các thuộc tính cần thiết như tên, mô tả và hình ảnh.

Một ví dụ về kiểm thử chức năng cho điều khiển Dish:

```

class DishControllerTest < ActiveSupport::TestCase
  test 'Dish is successfully created through the controller' do
    post :create, :dish => { :name => 'Example', :description => 'Example',
:picture => 'Example.jpg' }
    assert :redirect # Đảm bảo yêu cầu POST tới phương thức create
                     # thành công
  end
end

```

Kiểm thử này đảm bảo rằng điều khiển Dish chấp nhận thông tin một cách đúng đắn, lưu món ăn mới vào cơ sở dữ liệu và sau đó phản hồi bằng cách chuyển hướng người dùng tới trang cần thiết.

Ví dụ về kiểm thử tích hợp

Một ví dụ về kiểm thử tích hợp cho chức năng “Thêm món ăn”:

```

class DishFlowsTest < ActionController::IntegrationTest
  test 'Sign in and add a dish' do
    get '/sign_in'
    assert_response :success
    post_via_redirect '/sign_in', :username => 'example', :password => 'example'
    assert_equal '/welcome', path

    post :create, :dish => { :name => 'Example', :description => 'Example', :picture => 'Example.jpg' }
    assert :redirect # Đảm bảo yêu cầu POST tới phương thức create
                     # thành công
  end
end

```

Kiểm thử này đảm bảo cho một luồng chức năng nhất định của người dùng (đến trang đăng nhập, đang đăng nhập, được chuyển đến trang đăng nhập thành công và sau đó tạo một món ăn) hoạt động tốt.

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Ví dụ về kiểm thử chấp nhận người dùng

Một ví dụ kiểm thử chấp nhận người dùng cho chức năng đăng ký của ứng dụng web Noshster là:

Feature: Sign up

In order to use Noshster

As a non-member

I want to be able to register

Scenario: Sign up

Given I have filled in all the form's fields username, password, and email address with andrewpham, password, andrewpham@example.com

When I click sign up

Then I should be redirected to my newly created profile page

Given /^I have filled in all the form's required fields username, password, and email address with (.+)/ do |inputs|

 user_fields = inputs.split(',')

 @user = User.new

 @user.username = user_fields[0]

 @user.password = user_fields[1]

 @user.email_address = user_fields[2]

end

When /^I click sign up\$/ do

 @user.save!

end

Then /^I should be redirected to my newly created profile page\$/ do

 assert_response :redirect

 assert_redirected_to @user

end

Đây là kiểm thử giả lập một người dùng trong thực tế và đảm bảo rằng khi người dùng thực tế sử dụng ứng dụng web, ứng dụng sẽ hoạt động như mong đợi.

CASE STUDY 2 (CONFEROUS)

Conferous là một ứng dụng web quản lý cuộc gọi hội nghị trực tuyến cho phép các nhóm và tập thể dễ dàng cộng tác bằng cách thiết lập các cuộc gọi hội nghị đơn giản.

Tầm nhìn và Mục tiêu của sản phẩm

Tầm nhìn sản phẩm của Conferous là tạo ra một ứng dụng web quản lý cuộc gọi hội nghị trực tuyến, đơn giản và dễ sử dụng. Ứng dụng này có thể được sử dụng để người dùng dễ dàng truy cập hoặc ghi nhớ các cuộc gọi hội nghị và lưu lại các điện đàm này để nhóm tham khảo sau.

Mục tiêu tổng thể của Conferous là tạo ra một tổng đài để phục vụ các cuộc gọi hội nghị đơn giản.

- **Ai:** Những người tham dự hội nghị của doanh nghiệp
- **Tại sao:** Tạo ra các cuộc gọi hội nghị dễ sử dụng và ghi nhớ
- **Cái gì:** Các cuộc gọi hội nghị
- **Ở đâu:** Chỉ giới hạn trong phạm vi Hoa Kỳ
- **Khi nào:** 24x7

Thu thập yêu cầu bằng cách sử dụng kỹ thuật trực quan trong cuốn sách

Sử dụng kỹ thuật thu thập yêu cầu trực quan trong Chương 4, nhóm có khả năng xác định các yêu cầu sau sử dụng phép loại suy “rừng và cây” (Hình A.34).

Tầm nhìn kiến trúc và lập kế hoạch phát hành/Lập Kế hoạch Sprint

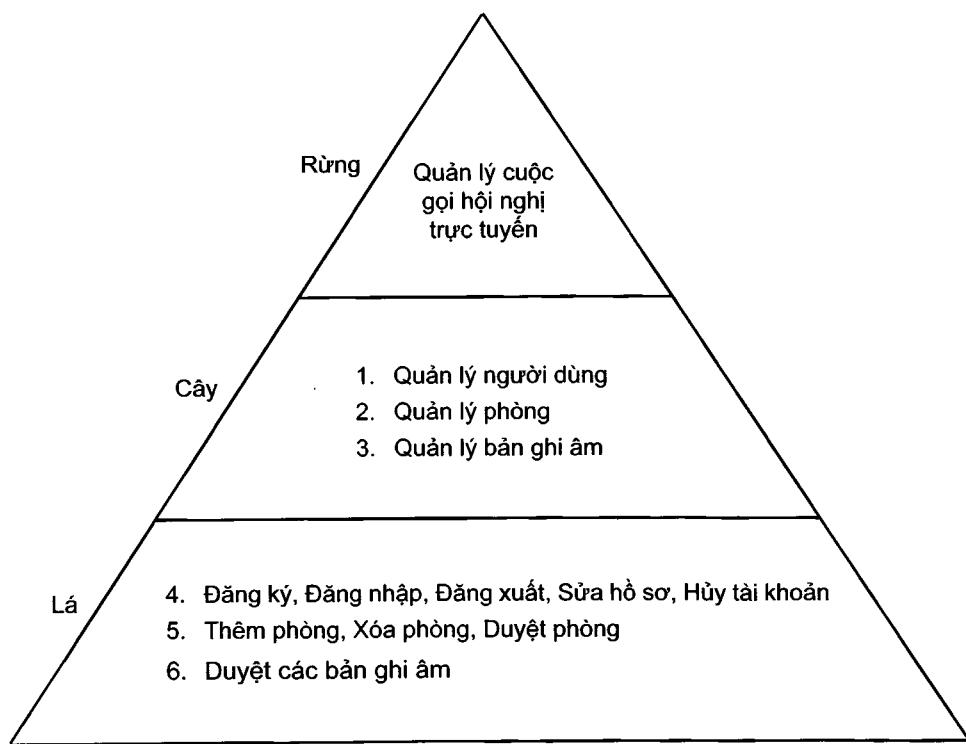
Ban đầu, chúng tôi đã trình bày một số yêu cầu về ứng dụng, những yêu cầu đó khi tập hợp lại với nhau trông sẽ khá nhiều như trong Hình A.35.

Tầm nhìn Kiến trúc

Sử dụng phương pháp đã học trong Chương 6, chúng ta nhanh chóng thấy mọi thứ rõ ràng hơn, như trình bày trong Hình A.36.

Với ứng dụng Conferous, chúng tôi quyết định thực hiện theo lát cắt ngang; tức là, thay vì phát triển toàn bộ các cây chức năng trong cùng một Sprint, chúng tôi quyết định phát triển các nhóm nhỏ trên nhiều cây chức năng khác nhau trong một Sprint và mở rộng dần sang các chức năng khác khi chúng tôi chuyển sang các Sprint tiếp theo.

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm



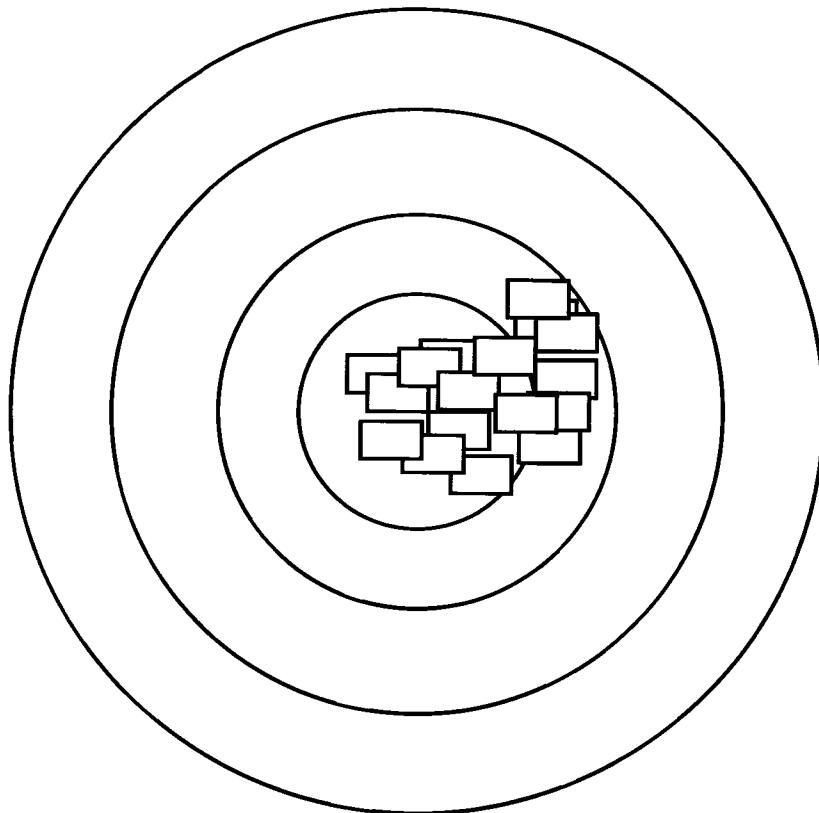
Hình A.34

Phân cấp các yêu cầu cho phần mềm quản lý hội nghị Conferous.

Kế hoạch phát hành và kế hoạch Sprint của ứng dụng Conferous

Rừng	Cây	Lá
Quản lý cuộc gọi hội nghị trực tuyến		
Bản phát hành 1		
	Sprint 1	
	Quản lý người dùng	Đăng ký, Đăng nhập, Đăng xuất
	Quản lý phòng	Thêm phòng, Xóa phòng, Duyệt phòng
	Sprint 2	
	Quản lý bản ghi âm	Duyệt các bản ghi âm
	Quản lý người dùng	Sửa hồ sơ, Hủy tài khoản

Lý do để thực hiện theo cách thức này là chúng tôi cần người dùng có thể đăng ký các tính năng cơ bản của phòng hội đàm để bắt đầu sử dụng ứng dụng ngay lập tức thay vì phải chờ đợi tất cả các tính năng của một phòng hội đàm được phát triển xong. Và hóa ra là chúng tôi đã tính toán đúng.



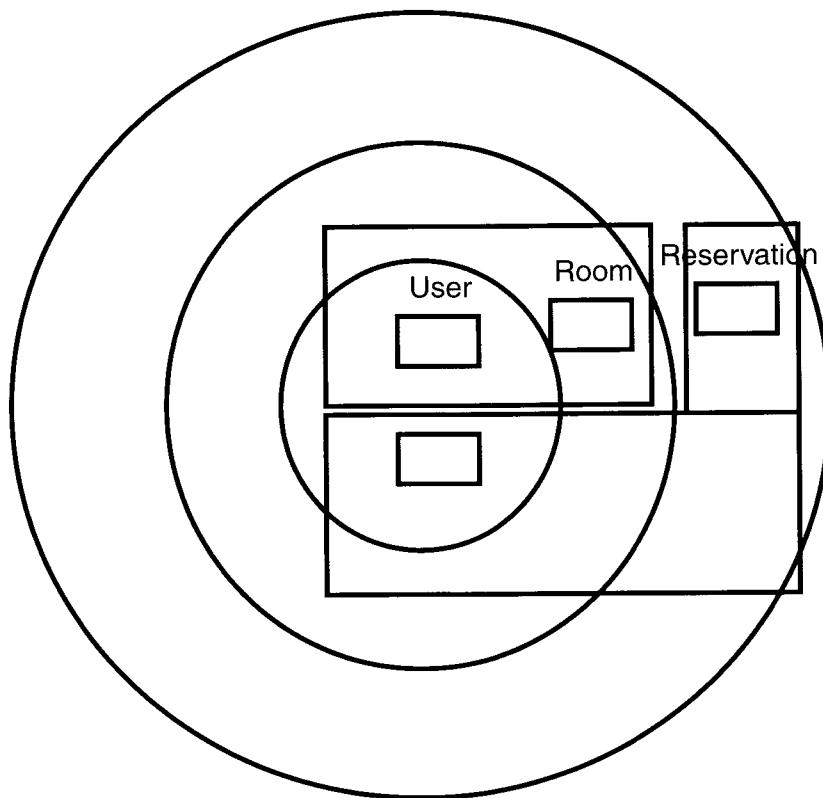
Hình A.35

Hình ảnh khi lần đầu tiên danh sách các yêu cầu được tập hợp lại với nhau.

Ước tính dự án sử dụng kỹ thuật dựa trên tiêu chí khách quan

Bảng sau là một bản tóm tắt điểm ước tính của từng Story trong mỗi Sprint bằng cách sử dụng kỹ thuật dựa trên tiêu chí khách quan trình bày trong phần trước của cuốn sách. Bạn có thể tìm các công thức tính cho mỗi cột trong Chương 5.

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm



User	Người dùng
Room	Phòng
Reservation	Đặt phòng

Hình A.36

Cắt ứng dụng theo chiều ngang.

PBI (Story)	Đặc điểm					Tổng UP (Điểm chưa hiệu chỉnh)	Hệ số	AP (Điểm đã hiệu chỉnh)	ED (Khía cạnh Môi trường)	PPS =(AP*ED)/36)
	Kiểu Tương tác	Quy tắc	Nghiệp vụ	Thực thể	Kiểu Thao tác Dữ liệu					
Sprint 1										
Đăng ký	3	1	1	2	7	1	7	18	3.5	
Đăng nhập	3	1	1	2	7	1	7	18	3.5	
Đăng xuất	3	1	1	1	6	1	6	18	3	
Thêm phòng	3	1	1	2	7	1	7	18	3.5	
Xóa phòng	3	1	1	3	8	1	8	18	4	
Duyệt phòng	3	1	1	1	6	1	6	18	3	
Sprint 2										
Sửa hồ sơ	3	1	1	3	8	1	8	18	4	
Hủy tài khoản	3	1	1	2	7	1	7	18	3.5	
Duyệt các bản ghi âm	3	1	1	1	6	1	6	18	3	

Phát triển Conferous

Sprint 1 của Conferous

Mục tiêu của Sprint đầu tiên chỉ là phát triển những chức năng cần thiết cho việc phát triển các chức năng khác của ứng dụng sau này.

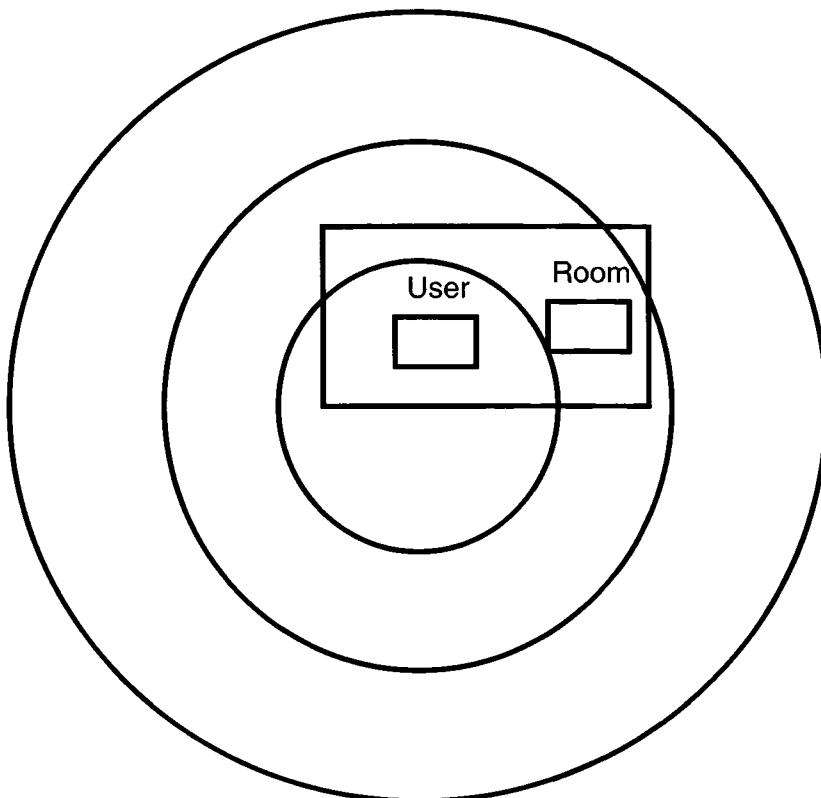
Các User story (Sprint 1) Các Story được phát triển trong Sprint này là:

1. Đăng ký
2. Đăng nhập
3. Đăng xuất
4. Thêm phòng
5. Xóa phòng
6. Duyệt phòng

Trong Hình A.37, bạn sẽ thấy sự tổ chức các mô hình dữ liệu cho Bản phát hành 1 – Sprint 1.

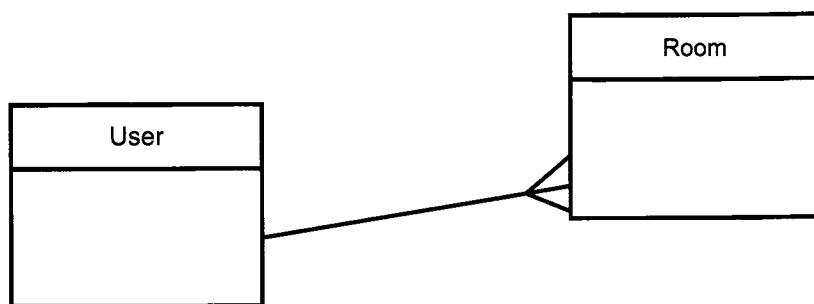
Trong Hình A.38, bạn sẽ thấy mối quan hệ giữa các mô hình dữ liệu cho Bản phát hành 1 – Sprint 1.

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm



Hình A.37

Vòng dữ liệu của Conferous trong Sprint 1.



Hình A.38

Mô hình dữ liệu của Conferous trong Sprint 1.

Mô hình-Khung nhìn-Điều khiển của Conferous (Sprint 1) Mô hình User cho phép ứng dụng Conferous lưu trữ thông tin đăng ký của người dùng. Mô hình này sử dụng plugin Authlogic và chỉ rõ rằng một người dùng tiềm năng không cần thiết phải nhập xác nhận mật khẩu (chỉ nhập tên đăng nhập, mật khẩu, email, ...) để đăng ký.

```

class User < ActiveRecord::Base
  acts_as_authentic do |c|
    c.require_password_confirmation = false
  end

  has_many :rooms
end
-----
```

Mô hình UserSession cho phép ứng dụng Conferous xác thực người dùng đăng nhập với thông tin đăng ký đã được lưu trong cơ sở dữ liệu. Do đó, khi một người dùng đăng nhập với những thông tin không hợp lệ, không trùng khớp với thông tin trong cơ sở dữ liệu, ứng dụng sẽ hiển thị một thông báo lỗi thân thiện. Ngược lại người dùng thường sẽ được chuyển vào một vùng hạn chế tương ứng với quyền của họ.

```

class UserSession < Authlogic::Session::Base
  def to_key
    new_record?? nil: [ self.send(self.class.primary_key) ]
  end
end
-----
```

Mô hình Room cho phép ứng dụng Conferous lưu trữ thông tin của phòng. Mô hình này đảm bảo mỗi phòng đều có một tên và mô tả, tên thì phải là duy nhất. Mô hình Room cũng thiết lập một kết nối giữa phòng và người dùng đã tạo ra nó để phục vụ việc báo cáo về sau.

```

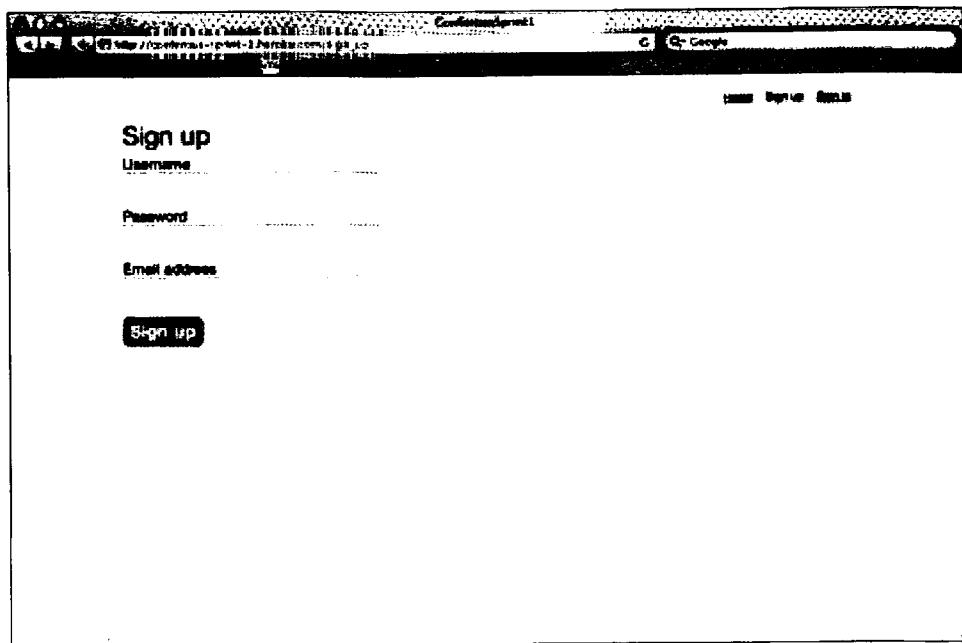
class Room < ActiveRecord::Base
  validates :name, :presence => true, :uniqueness => true
  validates :description, :presence => true
  belongs_to :user
end
-----
```

Khung nhìn

Đăng ký

Khung nhìn này gồm một form partial đăng ký cùng với các thông tin khác như tiêu đề, ...

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm



Hình A.39

Trang đăng ký.

```
<h1>Sign up</h1>
<%= form_for @user do |f| %>
  <%= f.error_messages %>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Sign up' %>
<% end %>
```

Form partial (dùng cho trang đăng ký và chỉnh sửa hồ sơ sau này)

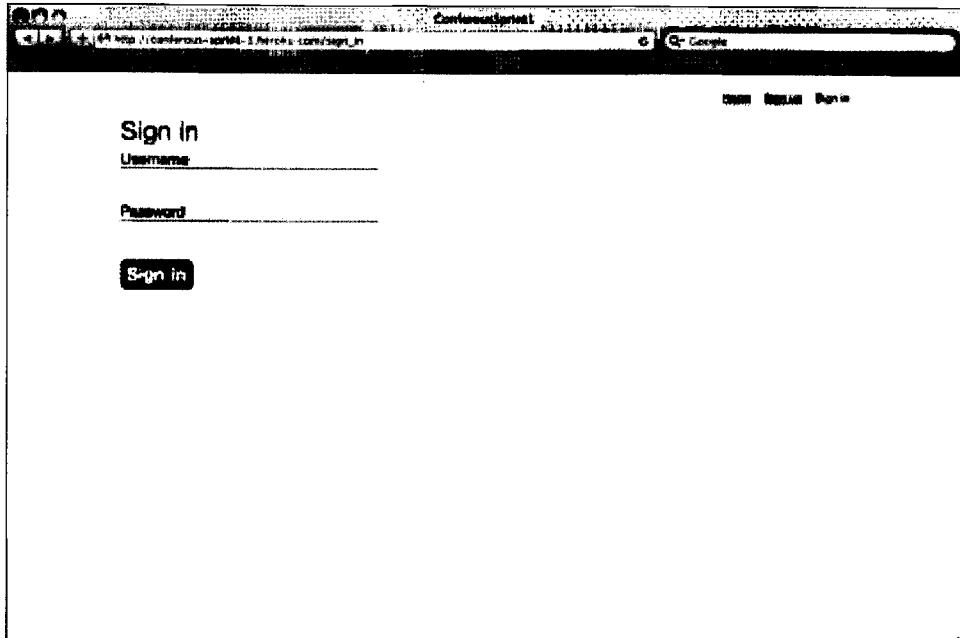
Đây là một form tái sử dụng được nhằm hiển thị cho người dùng một form đăng ký, họ có thể điền thông tin đăng ký của mình vào đây. Form này được đưa vào trang Đăng ký thông qua dòng mã `<%= render :partial => 'form', :object => f %>`.

```
<p>
  <%= form.label :username %><br />
  <%= form.text_field :username %>
</p>
<p>
  <%= form.label :password %><br />
  <%= form.password_field :password %>
</p>
<p>
```

```
<%= form.label :email_address %><br />
<%= form.text_field :email_address %>
</p>
```

Đăng nhập

Khung nhìn này sử dụng form đăng ký nhưng được tùy chỉnh theo hướng cho phép người dùng đăng nhập vào ứng dụng web.



Hình A.40

Trang đăng nhập.

```
<h1>Sign in</h1>
<%= form_for @user_session do |f| %>
  <%= f.error_messages %>
  <p>
    <%= f.label :username %><br />
    <%= f.text_field :username %>
  </p>
  <p>
    <%= f.label :password %><br />
    <%= f.password_field :password %>
  </p>
  <%= f.submit 'Sign in' %>
<% end %>
```

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

Thêm Phòng

Khung nhìn này hiển thị form thêm phòng cho phép người dùng nhập thông tin vào để tạo một phòng.

The screenshot shows a web page titled "Add a room". There are two input fields: one for "Name" and one for "Description". Below the "Name" field is a small error message: "There was a problem saving your changes. Please try again." At the bottom of the form is a large red "Add room" button.

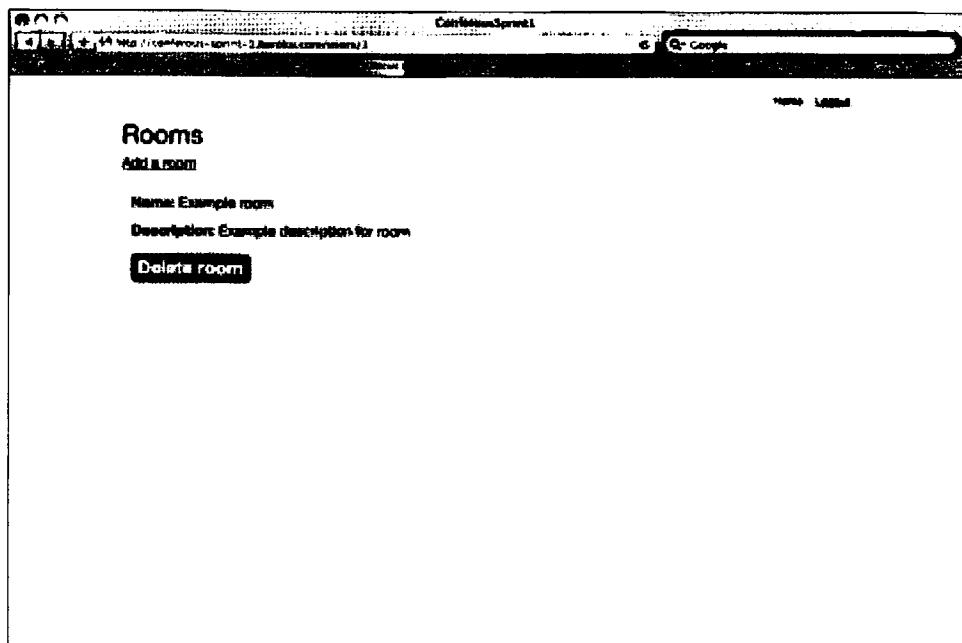
Hình A.41

Trang thêm một phòng.

```
<h1>Add a room</h1>
<%= form_for [@user, @room] do |f| %>
  <%= f.error_messages %>
  <p>
    <%= f.label :name %><br />
    <%= f.text_field :name %>
  </p>
  <p>
    <%= f.label :description %><br />
    <%= f.text_area :description, :rows => 5, :cols => 50 %>
  </p>
  <%= f.submit 'Add room' %>
<% end %>
```

Xóa Phòng

Khung nhìn này hiển thị một nút bấm cho phép người dùng xóa một phòng nhất định.



Hình A.42

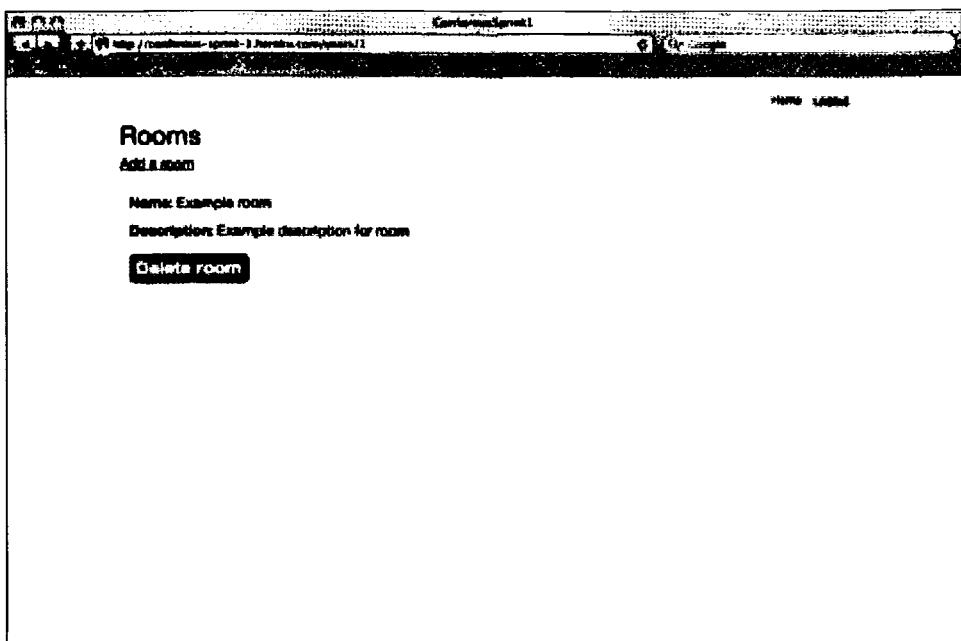
Trang xóa một phòng.

```
<div class="room">
  <p>
    <span class="bold">Name:</span>&ampnbsp=<%= room.name %>
  </p>
  <p>
    <span class="bold">Description:</span>&ampnbsp=<%= room.description %>
  </p>
  <p>
    <%= button_to 'Delete room', user_room_path(@user, room),
      :method => :delete %>
  </p>
</div>
```

Duyệt Phòng

Khung nhìn này lấy thông tin tất cả các phòng do điều khiển trả về từ cơ sở dữ liệu và hiển thị chúng.

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm



Hình A.43

Trang duyệt phòng.

```
<h1>Rooms</h1>
<%= link_to 'Add a room', new_user_room_path(@user) %>
<div id="rooms">
  <%= render @user.rooms %>
</div>
```

Room partial (để tái sử dụng trong trang hiển thị chi tiết phòng)

Đây là một phần của khung nhìn có thể tái sử dụng, đã được dùng ở trang trước để hiển thị thông tin phòng theo một cách nhất định.

```
<div class="room">
  <p>
    <span class="bold">Name:</span>&nbsp;<%= room.name %>
  </p>
  <p>
    <span class="bold">Description:</span>&nbsp;<%= room.description %>
  </p>
  <p>
    <%= button_to 'Delete room', user_room_path(@user, room), :method => :delete %>
  </p>
</div>
```

Controller - Điều khiển Điều khiển UserController nhận yêu cầu từ trình duyệt và trả về khung nhìn thích hợp kèm theo dữ liệu cần thiết. Ví dụ, nếu người dùng đi đến trang đăng ký, điều khiển UserController trả về trang đăng ký cùng với một đối tượng User rỗng để điền thông tin vào. Một ví dụ khác, khi người dùng nhấn vào nút đăng ký (Sign up), điều khiển nhận thông tin được điền trên form và đưa vào trong đối tượng User rỗng, sau đó cho đối tượng này tự lưu vào cơ sở dữ liệu (đây chính là lúc mô hình sẽ đảm nhận và thực hiện quá trình kiểm tra tính hợp lệ của dữ liệu).

```

class UsersController < ApplicationController
  before_filter :require_no_user, :only => [:new, :create]
  # phương thức new khởi tạo một đối tượng User mới.
  def new
    @user = User.new
  end

  # phương thức create khởi tạo một đối tượng User mới và truyền
  # cho nó các dữ liệu do người dùng cung cấp (params[:user]). Sau
  # đó đối tượng này thử lưu thông tin của nó. Hành động này có thể
  # thất bại nếu người dùng không cung cấp tất cả các dữ liệu hợp
  # lệ theo yêu cầu của mô hình User hoặc cung cấp dữ liệu không
  # chính xác. Nếu thành công, người dùng sẽ nhận được một thông
  # báo và được chuyển tới trang thông tin chi tiết người dùng. Nếu
  # thất bại, phương thức sẽ hiển thị form để người dùng thử lại.

  def create
    @user = User.new(params[:user])
    if @user.save
      flash[:notice] = 'Sign up successful!'
      redirect_to @user
    else
      render :new
    end
  end

  # phương thức show lấy người dùng đang đăng nhập thông qua một
  # phương thức được Authlogic cung cấp và gán đối tượng này cho
  # biến @user

  def show
    @user = current_user
  end
end
-----
```

Điều khiển UserSessionsController nhận thông tin do người dùng cung cấp trên form đăng nhập và đưa vào trong một đối tượng UserSession rỗng rồi cho nó tự lưu. Sau

Phụ lục A ▪ Tìm hiểu hai case study về phát triển sản phẩm phần mềm

đó, UserSessionsController sẽ đưa ra những hành động cần thiết tùy thuộc vào đối tượng mô hình có tự lưu được thông tin của nó hay không (có nghĩa là đối tượng đã xác thực thành công). Điều khiển này cũng có khả năng đăng xuất người dùng khỏi hệ thống và quay trở về trang chủ.

```
class UserSessionsController < ApplicationController
  before_filter :require_no_user, :only => [:new, :create]
  before_filter :require_user, :only => :destroy

  # phương thức new khởi tạo một đối tượng UserSession mới
  def new
    @user_session = UserSession.new
  end

  # phương thức create khởi tạo một đối tượng UserSession mới và truyền
  # cho nó dữ liệu do người dùng cung cấp (params[:user_session]).
  # Sau đó, đối tượng này thử lưu thông tin của nó. Hành động này có
  # thể thất bại nếu người dùng không cung cấp tất cả các dữ liệu
  # hợp lệ theo yêu cầu của mô hình UserSession hoặc cung cấp dữ liệu
  # không chính xác. Nếu thành công, người dùng sẽ nhận được một thông
  # báo và được chuyển tới trang thông tin chi tiết người dùng. Nếu
  # thất bại, phương thức sẽ hiển thị form để người dùng thử lại.

  def create
    @user_session = UserSession.new(params[:user_session])
    if @user_session.save
      flash[:notice] = 'Sign in successful!'
      redirect_back_or_default @user_session.user
    else
      render :new
    end
  end

  # phương thức destroy lấy session của người dùng đang đăng nhập và
  # xóa các giá trị đã lưu trữ, về cơ bản là đăng xuất người dùng đang
  # đăng nhập hệ thống.

  def destroy
    current_user_session.destroy
    flash[:notice] = 'Logout successful!'
    redirect_to root_path
  end
-----
```

```

class RoomsController < ApplicationController
  before_filter :require_user
  before_filter :get_user

  # phương thức new lấy người dùng đang đăng nhập nhờ filter get_user,
  # và khởi tạo một đối tượng Room.

  def new
    @room = @user.rooms.build
  end

  # phương thức create khởi tạo một đối tượng Room và truyền cho
  # nó dữ liệu do người dùng cung cấp (params[:room]). Sau đó, đối
  # tượng thử lưu thông tin của nó. Quá trình lưu bao gồm việc xác thực
  # các thông tin được cung cấp bởi người dùng và thử khớp với thông
  # tin lưu trong CSDL. Nếu thành công, người dùng sẽ nhận được một
  # thông báo và được chuyển tới trang thông người dùng. Nếu thất
  # bại, phương thức sẽ hiển thị form để người dùng thử lại.

  def create
    @room = @user.rooms.build(params[:room])
    if @room.save
      flash[:notice] = 'You have successfully created a room!'
      redirect_to @user
    else
      render :new
    end
  end

  # phương thức destroy lấy một phòng cụ thể và xóa phòng này
  # khỏi CSDL.

  def destroy
    @room = @user.rooms.find(params[:id])
    @room.destroy
    flash[:notice] = 'You have successfully deleted a room!'
    redirect_to @user
  end

  private
  def get_user
    @user = current_user
  end
end

```

Conferous - Sprint 2

Mục tiêu của Sprint thứ hai là xây dựng dựa trên nền tảng của Sprint đầu tiên và phát triển các tính năng trung tâm có giá trị của ứng dụng.

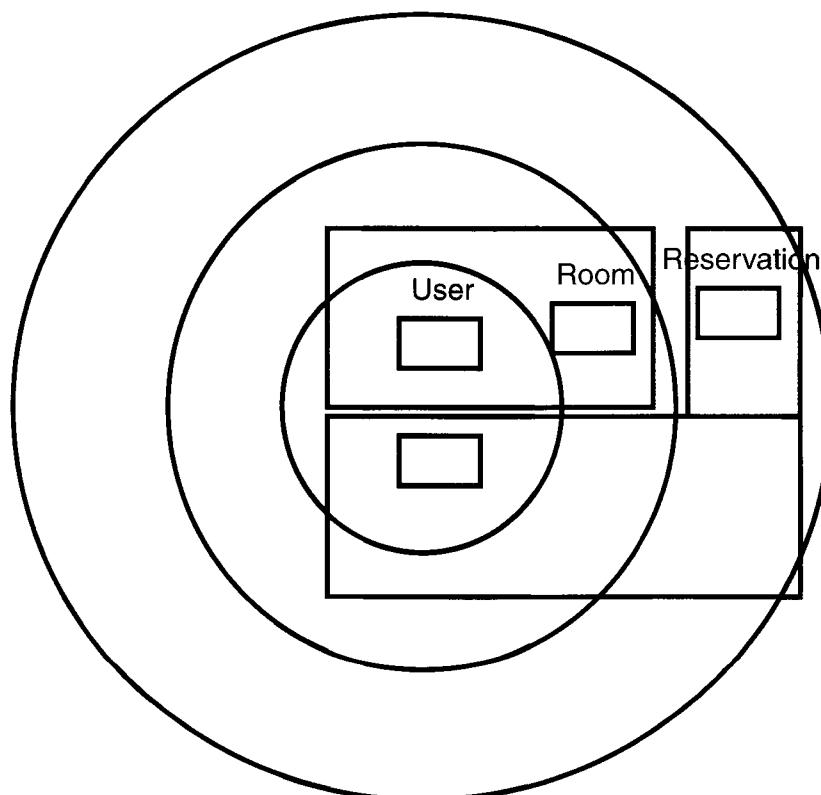
Các User story (Sprint 2)

Các story sẽ được phát triển trong suốt Sprint này là:

1. Duyệt các bản ghi âm
2. Sửa hồ sơ
3. Hủy tài khoản

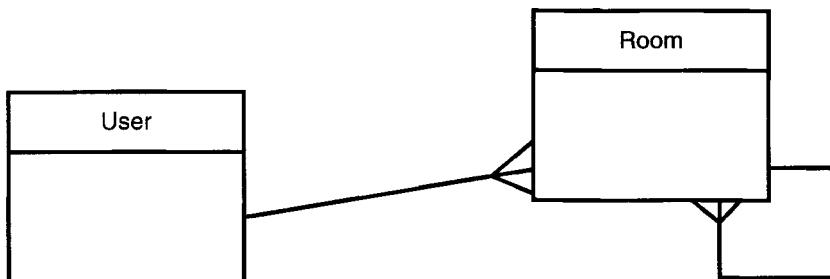
Trong Hình A.44, bạn sẽ tìm thấy cách tổ chức mô hình dữ liệu cho Bàn phát hành 1 – Sprint 2.

Trong Hình A.45, bạn sẽ tìm thấy mối quan hệ của mô hình dữ liệu cho Bàn phát hành 1 – Sprint 2.



Hình A.44

Vòng tròn dữ liệu Conferous trong suốt Sprint 2.

**Hình A.45**

Mô hình dữ liệu của ứng dụng Conferous trong suốt Sprint 2.

Mô hình — Khung nhìn— Điều khiển của ứng dụng Conferous (Sprint 2)

Mô hình Recording cho phép ứng dụng web Conferous lưu trữ thông tin bản ghi âm đối với mỗi cuộc gọi hội nghị được thực hiện trong một phòng xác định.

```

class Recording < ActiveRecord::Base
  belongs_to :room
end
-----
```

Mô hình Room được chỉnh sửa để tạo ra kết nối giữa phòng và bản ghi tương ứng, do đó khi người dùng thử xem bản ghi âm được tạo ra trong một phòng xác định, họ có thể dễ dàng tìm kiếm trong cơ sở dữ liệu.

```

class Room < ActiveRecord::Base
  validates :name, :presence => true, :uniqueness => true
  validates :description, :presence => true
  has_many :recordings
  belongs_to :user
  after_create :setup_test_data
  private
  def setup_test_data
    1.upto(3) do
      self.recordings.create
    end
  end
end
```

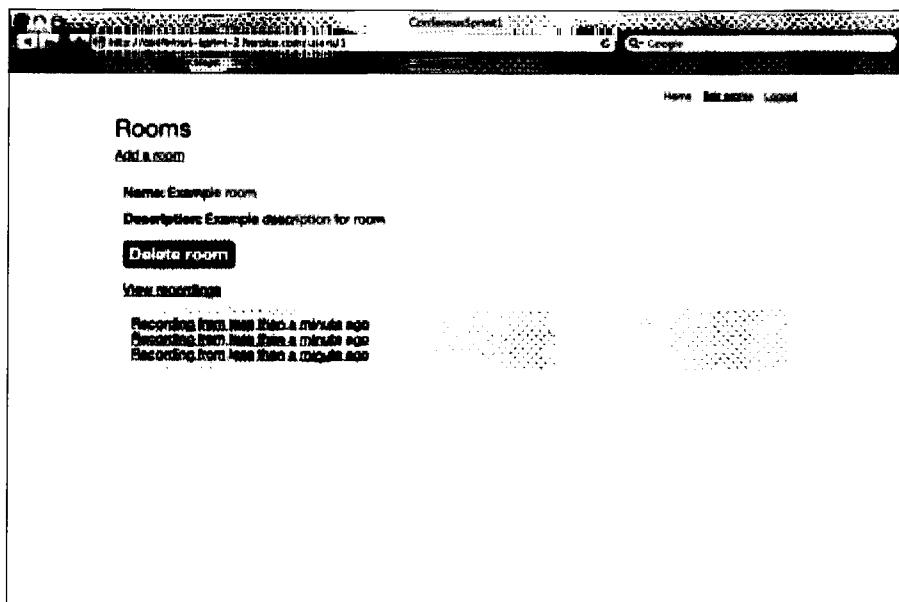
Khung Nhìn

Duyệt các bản ghi âm

Khung nhìn này hiển thị tất cả các phòng được tàng điều khiển trả về từ cơ sở dữ liệu và hiển thị chúng cùng với tất cả các bản ghi âm tương ứng.

Phụ lục A - Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
<% content_for :head do %>
<script type="text/javascript">
$(document).ready(function() {
    $('#view_recordings').click(function() {
        $('.recordings').slideToggle();
        return false;
    })
});
</script>
<% end %>
<h1>Rooms</h1>
<%= link_to 'Add a room', new_user_room_path(@user) %>
<div id = "rooms">
    <%= render @user.rooms %>
</div>
```



Hình A.46

Trang duyệt các bản ghi âm.

Room partial (tài sử dụng để hiển thị thông tin chi tiết về phòng)

Đây là một phần của khung nhìn có khả năng tái sử dụng, được dùng ở trang trước để hiển thị thông tin phòng theo cách nhất định. Partial này được chỉnh sửa từ Sprint trước để hiển thị các bản ghi âm của một phòng hội nghị. Room Partial được đưa vào trang “Duyệt Phòng” nhờ dòng mã `<%= render @user.rooms %>`.

```

<div class ="room">
  <p>
    <span class="bold">Name:</span>&nbsp;<%= room.name %>
  </p>
  <p>
    <span class="bold">Description:</span>&nbsp;<%= room.description %>
  </p>
  <p>
    <%= button_to 'Delete room', user_room_path(@user, room), :method => :delete %>
  </p>
  <p>
    <%= link_to 'View recordings', nil, :id => 'view_recordings' %>
  </p>
  <p class = "recordings">
    <%= render room.recordings %>
  </p>
</div>

```

Sửa hồ sơ

Khung nhìn này sử dụng form partial chỉnh sửa hồ sơ dưới đây và bổ sung thêm một số thông tin như tiêu đề, ...

The screenshot shows a web browser window with the URL <http://localhost:3001/auth/edit>. The page title is 'Edit profile'. It contains the following form fields:

- Username:** andrewpham
- Password:** (password field)
- Email address:** andrewpham@example.com

Below the form are two buttons:

- A blue rectangular button labeled 'Update profile'.
- A dark rectangular button labeled 'Delete account'.

Hình A.47

Trang sửa hồ sơ.

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
<h1>Edit profile</h1>
<%= form_for @user do |f| %>
  <%= f.error_messages %>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Update profile' %>
<% end %>
```

Form partial (để sử dụng cho trang đăng ký và sửa hồ sơ)

Đây là một form có thẻ tái sử dụng để hiển thị cho người dùng form sửa hồ sơ, sau đó họ cũng có thể sử dụng để cập nhật thông tin đăng ký người dùng. Form partial này được đưa vào trang “Sửa hồ sơ” nhờ dòng mã `<%= render :partial => 'form', :object => f %>`.

```
<p>
  <%= form.label :username %><br />
  <%= form.text_field :username %>
</p>
<p>
  <%= form.label :password %><br />
  <%= form.password_field :password %>
</p>
<p>
  <%= form.label :email_address %><br />
  <%= form.text_field :email_address %>
</p>
```

Hủy tài khoản

Khung nhìn này sẽ lấy người dùng hiện tại trong cơ sở dữ liệu và hiển thị form để người dùng chỉnh sửa thông tin của mình. Đồng thời, khung nhìn này cũng cho phép người dùng hủy tài khoản của mình bằng cách dùng nút xóa tài khoản (Delete account).

```
<h1>Edit Profile</h1>
<%= form_for @user do |f| %>
  <%= f.error_messages %>
  <%= render :partial => 'form', :object => f %>
  <%= f.submit 'Update profile' %>
<% end %>
<%= button_to 'Delete account', @user, :method => :delete %>
```

Form partial (để sử dụng cho trang đăng ký và sửa hồ sơ sau này)

Đây là một form có thẻ tái sử dụng để hiển thị cho người dùng form đăng ký, người dùng có thể dùng form này để điền thông tin đăng ký của họ. Form partial này được đưa vào trang “Hủy tài khoản” nhờ dòng mã `<%= render :partial => 'form', :object => f %>`.

```

<p>
  <%= form.label :username %><br />
  <%= form.text_field :username %>
</p>
<p>
  <%= form.label :password %><br />
  <%= form.password_field :password %>
</p>
<p>
  <%= form.label :email_address %><br />
  <%= form.text_field :email_address %>
</p>

```

Controller - Điều khiển UserController được chỉnh sửa để hỗ trợ tính năng Hủy tài khoản.

```

class UsersController < ApplicationController
  before_filter :require_no_user, :only => [:new, :create]
  # phương thức new khởi tạo một đối tượng User mới.
  def new
    @user = User.new
  end

  # phương thức create khởi tạo một đối tượng User mới và truyền cho
  # nó dữ liệu do người dùng cung cấp (params[:user]). Sau đó đối
  # tượng này thử lưu thông tin của nó. Hành động này có thể thất
  # bại nếu người dùng không cung cấp tất cả các dữ liệu hợp lệ theo
  # yêu cầu của mô hình User hoặc cung cấp dữ liệu không chính xác.
  # Nếu thành công, người dùng sẽ nhận được một thông báo và được
  # chuyển tới trang thông tin chi tiết người dùng. Nếu thất bại,
  # phương thức sẽ hiển thị form để họ thử lại.
  def create
    @user = User.new(params[:user])
    if @user.save
      flash[:notice] = 'Sign up successful!'
      redirect_to @user
    else
      render :new
    end
  end

  # phương thức show lấy người dùng đang đăng nhập qua một
  # phương thức được cung cấp bởi Authlogic và gán đối tượng này
  # cho biến @user

```

Phụ lục A • Tìm hiểu hai case study về phát triển sản phẩm phần mềm

```
def show
  @user = current_user
end

# phương thức destroy tìm một người dùng cụ thể và xóa thông tin
# về người dùng này khỏi CSDL, túi đây tài khoản của người dùng
# bị hủy bỏ
def destroy
  @user = User.find(params[:id])
  @user.destroy
  flash[:notice] = 'You have successfully canceled your account!'
  redirect_to root_path
end
end
```

PHỤ LỤC B

BẠN CÓ THỂ HOẶC CÓ NÊN CHẤM DỨT BẤT THƯỜNG MỘT SPRINT

LỜI GIỚI THIỆU

Nếu bạn đã đọc và tuân thủ tất cả những đề xuất đã trình bày trong suốt cuốn sách thì bạn sẽ không bao giờ đặt ra câu hỏi này hoặc tự mình vượt qua thử thách này, trừ khi việc chấm dứt Sprint là do yêu cầu của quản lý hoặc là tình hình kinh doanh.

Thông thường, việc chấm dứt một Sprint trước kỳ hạn là một điều bạn không muốn làm vì việc này sẽ làm nhóm khá nản lòng, nhưng lại là điều mà có khả năng bạn sẽ phải đối phó bởi vì nó xảy ra trong thế giới thực.

Trước khi chúng ta thảo luận về việc làm cách nào để khởi động lại sau khi một Sprint bị chấm dứt bất thường, hãy xem qua ba kịch bản khác nhau có thể xảy ra.

KHI NÀO MỘT SPRINT CÓ THỂ ĐƯỢC KẾT THÚC SỚM HƠN DỰ ĐỊNH?

- Do yêu cầu của quản lý:

Một ví dụ về tình huống này đó là người quản lý yêu cầu kết thúc dự án hoặc Sprint khi họ phải tuyên bố:

1. Dừng hoạt động
2. Đóng dự án
3. Tái cấu trúc

Tất cả những điều này đòi hỏi cần chấm dứt dự án hoặc Sprint của bạn.

Phụ lục B ▪ Bạn có thể hoặc có nên chấm dứt bất thường một Sprint

- Do tình hình kinh doanh:

Một ví dụ về tình huống này đó là khi đội ngũ tiếp thị phải chuyển hướng chiến lược của mình sang một phân khúc thị trường mới dẫn đến việc phải chuyển hướng quỹ đầu tư đã được phân bổ cho các dự án của bạn.

- Do nhóm phát triển:

Một ví dụ cho tình huống này đó là khi nhóm nhận ra rằng họ sẽ không thể đạt được cam kết của mình, hoặc do một quyết định kỹ thuật sai hoặc bởi vì một số xung đột đã gây rắc rối cho công việc của nhóm.

LÀM THẾ NÀO ĐỂ TRÁNH DỪNG MỘT SPRINT SỚM HƠN DỰ KIẾN

Như chúng ta đã thấy, việc chấm dứt một Sprint gần như là không thể tránh khỏi nếu đó là quyết định của quản lý hoặc các quyết định về kinh doanh.

Tuy nhiên, khi việc chấm dứt một Sprint xuất phát từ hành động của nhóm hoặc do thiếu khả năng đạt được mục tiêu của Sprint, có một số điều bạn có thể làm – hoặc nên làm – để tránh phải chấm dứt Sprint trước thời hạn:

- Làm giảm tốc độ của nhóm (hay phạm vi) khi mới bắt đầu vào dự án Scrum:

Khi bạn mới bắt đầu Scrum, hãy luôn thận trọng với thời gian của mọi người và hãy sắp xếp thời gian của họ như thế họ sẽ làm việc bán thời gian vào lúc đầu.

- Đừng vội vàng để đưa ra những quyết định sai về kỹ thuật hoặc kiến trúc:

Để đơn giản, chúng tôi khuyến nghị bạn nên làm theo các lời khuyên trong Chương 6 và 7 trước khi đưa ra bất cứ quyết định gì liên quan tới kiến trúc trong tương lai của ứng dụng.

- Hãy chắc chắn để có được cam kết từ cấp quản lý về việc cung cấp tất cả các tài nguyên cần thiết trước khi bạn bắt đầu dự án.

Dù có hay không dùng Scrum thì bạn cũng không có cách nào để hoàn thành bất cứ điều gì nếu không có những tài nguyên phù hợp.

- Hãy theo dõi dự án hằng ngày

- Quan sát biểu đồ Burndown mỗi ngày.

- Hãy chấn chỉnh là người quản lý hiểu rằng họ không nên giúp bạn châm lo điều gì đó trong suốt quá trình triển khai một Sprint.

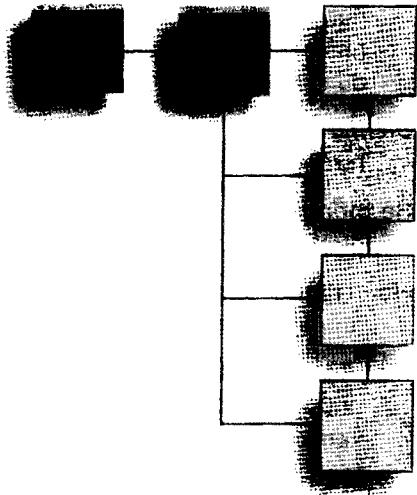
Các trở ngại kinh điển vẫn xảy ra thường xuyên trong tổ chức của chúng ta.

Làm thế nào để khởi động lại sau khi chấm dứt một Sprint sớm hơn dự định

LÀM THẾ NÀO ĐỂ KHỞI ĐỘNG LẠI SAU KHI CHẤM DỨT MỘT SPRINT SỚM HƠN DỰ ĐỊNH

Bất cứ khi nào bạn khởi động lại sau khi chấm dứt bất thường một Sprint, hãy luôn chắc chắn rằng bạn biết lý do dẫn tới việc Sprint đó bị chấm dứt, không chỉ các dấu hiệu, mà cả những nguyên nhân sâu xa dẫn tới điều này. Tiếp theo, những gì bạn cần làm là tổ chức một phiên họp lập kế hoạch mới và đảm bảo rằng bạn có tính tới đến tình huống và môi trường mới trước khi lao đầu vào một Sprint

THUẬT NGỮ



Agile (Agile Software Development - Phát triển Phần mềm Linh hoạt). Tên sử dụng chung cho một nhóm các phương pháp phát triển phần mềm có xu hướng đổi lặp với những phương pháp hoặc quy trình phát triển phần mềm/quản lý dự án cứng nhắc và nặng nề.

Bên liên quan. Tên gọi chung để chỉ bất cứ ai hoặc nhóm nào có can dự tới sự thành công của nhóm dự án Scrum.

Biểu đồ Burndown. Biểu đồ được sử dụng bởi Nhóm Scrum (*xem định nghĩa bên dưới*) để hiển thị khối lượng công việc còn lại trong một Sprint (*xem định nghĩa bên dưới*).

Các thành phần dữ liệu nghiệp vụ cốt lõi. Các thành phần dữ liệu liên quan tới phần cốt lõi của lĩnh vực nghiệp vụ hay nghiệp vụ của tổ chức. Ví dụ, phòng và đặt phòng là những thành phần dữ liệu cốt lõi của một khách sạn, trong khi sách và bạn đọc là các thành phần cốt lõi của một thư viện. Việc thiết kế và xây dựng ứng dụng phần mềm xoay quanh những thành phần dữ liệu cốt lõi này đảm bảo tính ổn định lâu dài cho sản phẩm phần mềm và giúp việc mở rộng dễ dàng hơn trong tương lai mà không tốn quá nhiều chi phí xây dựng lại.

Đặc điểm của một nhà lãnh đạo chu đáo. Phẩm chất giúp một số người trở thành lãnh đạo giỏi. Sự trung thực, cởi mở, chân thành, sẵn sàng và chu đáo là một số ví dụ về những phẩm chất ấy.

Đảm bảo Chất lượng (QA - Quality Assurance). Một nhóm trong nhiều đơn vị CNTT, chịu trách nhiệm về các hoạt động đảm bảo chất lượng và kiểm thử đối với tất cả phần mềm mới phát triển. Họ không thuộc khung quy trình Scrum nhưng là một bộ phận mà Nhóm Scrum cần phải tương tác, ngay cả khi chỉ để mượn một kiểm thử viên chuyên nghiệp tới Nhóm Scrum nhằm tạo ra nhóm có khả năng liên chức năng.

Đánh giá sự sẵn sàng đối với Scrum. Một đánh giá nhanh khi bắt đầu một dự án Scrum, giúp đánh giá khả năng chuyển giao các kết quả của nhóm trong khi áp

Thuật ngữ

dụng Scrum. Thông qua đánh giá để biết được vị trí của nhóm, từ đó giúp họ nhận ra những gì cần phải làm nhằm cải thiện cơ hội thành công trong việc sử dụng Scrum.

Điểm ước tính chưa hiệu chỉnh. Ước tính các yêu cầu trước khi tính đến những ảnh hưởng của môi trường xung quanh.

Hệ thống Sản xuất Toyota (TPS - Toyota Production System). Hệ thống sản xuất được xem là tinh gọn nhất và hiệu quả nhất thế giới.

Hoàn thành. Bản thỏa thuận của tất cả ba thành phần trong Nhóm Scrum để xác định những gì được coi là hoàn thành bởi Nhóm Phát triển cuối mỗi Sprint.

Hợp Cài tiến Sprint (Sprint retrospective). Cuộc họp ở đó Nhóm Scrum sẽ nhìn lại những việc đã làm và những gì chưa làm được trong Sprint vừa qua. Họ xác định xem có bất cứ điều gì có thể học hỏi từ những kinh nghiệm vừa qua để giúp quy trình làm việc của Sprint tiếp theo tốt hơn.

Hợp Kế hoạch Sprint (Sprint planning meeting). Một cuộc họp gồm hai phần: trong phần đầu, Product Owner cho nhóm biết những yêu cầu mà anh/cô ta nghĩ nên đưa vào Sprint này và trong phần thứ hai, nhóm quyết định cách làm như thế nào để biến những yêu cầu đã được chọn thành phần tăng trưởng sản phẩm có thể chuyển giao được.

Hợp Scrum Hằng ngày (Cuộc họp đứng Hằng ngày). Cuộc họp tiến độ ngắn diễn ra hằng ngày, ở đó các thành viên cùng nhau (cùng một chỗ hoặc trực tuyến) đồng bộ và tìm hiểu về tiến độ hướng tới mục tiêu Sprint. Chỉ có ba câu hỏi được trả lời trong suốt cuộc họp này:

1. Những gì tôi đã làm kể từ buổi họp hôm trước?
2. Những gì tôi dự định làm từ giờ tới buổi họp kế tiếp?
3. Những gì cần trao công việc của tôi?

Huấn luyện (coaching). Nghệ thuật phát triển tài năng của một ai đó hoặc hỗ trợ ai đó phát triển hay đạt được một số mục tiêu cụ thể thông qua việc giúp họ xác định lộ trình và những cam kết cần thực hiện để đạt được kết quả.

kanban (viết thường). Xuất phát từ khung làm việc Kanban, có nghĩa là một tấm thẻ và kỹ thuật được dùng để giới hạn số công việc thực hiện đồng thời (work-in-process), do đó giúp gia tăng luồng công việc.

Kanban. Một khung làm việc để cải tiến quy trình, nổi tiếng với những đóng góp cho Hệ thống sản xuất Toyota (TPS).

Khối lượng công việc còn lại. Số giờ ước tính cần để nhóm hoàn tất bất cứ công việc nào chưa hoàn thành trong Sprint.

Kiến trúc Doanh nghiệp. Bản thiết kế cấp cao (khái quát) về hệ sinh thái của công ty, cho thấy các mối quan hệ và sự phụ thuộc lẫn nhau giữa tầm nhìn kinh doanh, chiến lược, quy trình, các ứng dụng phần mềm, dữ liệu và cơ sở hạ tầng.

Kiến trúc. Thiết kế tổng thể hoặc sự bố trí của các thành phần chính trong một ứng dụng phần mềm.

Kỹ thuật CUTFIT. Những quy tắc đơn giản mà Nhóm Scrum có thể sử dụng để kiểm định các User Story (trong khi thu thập các yêu cầu). CUTFIT là cụm từ viết tắt của Consistent (Nhất quán), Unambiguous (Không mơ hồ), Testable (Kiểm thử được), Feasible (Khả thi), Independent (Độc lập) và Traceable (Theo dõi được).

Kỹ thuật giải quyết xung đột. Các nhóm trong Scrum đều là nhóm tự tổ chức; sau đây là những kỹ thuật mà các thành viên trong nhóm dự án có thể sử dụng để giải quyết những xung đột trong nội bộ: Điều tiết, Thỏa hiệp, Cạnh tranh, Cộng tác và Né tránh.

Kỹ thuật thu thập yêu cầu trực quan. Kỹ thuật trực quan dựa trên hệ thống phân cấp cây và rừng, có thể giúp đơn giản hóa việc thu thập các yêu cầu người dùng cho những dự án phức tạp.

Lãnh đạo kiểu phục vụ. Một triết lý lãnh đạo cho rằng người lãnh đạo hoặc quản lý có thể thành công hơn và hiệu quả hơn khi hỗ trợ nhóm loại bỏ những trở ngại thay vì ra lệnh cho họ theo phong cách cũ là mệnh lệnh và kiểm soát.

Lập Kế hoạch Phát hành. Cuộc họp ở đó Product Owner chia sẻ với Nhóm Scrum về tầm nhìn sản phẩm, những chức năng nào (các User Story) nên được chuyển giao vào thời điểm nào. Với cách tiếp cận kiến trúc đã được khuyến nghị trong cuốn sách này, các thành viên Nhóm Phát triển có thể chủ động đề xuất những User Story cần được phát triển trước, thay vì thụ động tham dự cuộc họp như một số nhóm đã làm trong quá khứ.

Lean (Tinh gọn). Tên dùng chung cho tất cả các quy trình hoặc kỹ thuật giúp giảm thiểu những lãng phí của hệ thống hoặc quy trình.

Mục tiêu Sprint (Sprint Goal). Mục tiêu của Sprint được Product Owner đưa cho nhóm, chẳng hạn như “đặt nền tảng cho thẻ tín dụng của Hoa Kỳ”.

Nhóm (Nhóm Phát triển). Một nhóm thuộc nhóm Scrum, chịu trách nhiệm với tất cả hoạt động phát triển phần mềm, từ viết yêu cầu tới thiết kế, viết mã và kiểm thử.

Nhóm Kiến trúc Doanh nghiệp. Một nhóm có trong nhiều tổ chức công nghệ thông tin (CNTT), họ chịu trách nhiệm về kiến trúc tổng thể của các hệ thống CNTT trong công ty. Đây là nhóm mà Nhóm Scrum thường xuyên phải tương tác để có được sự chấp thuận về ứng dụng trong dự án của họ và để đảm bảo rằng kiến trúc ứng dụng mới phù hợp với kiến trúc doanh nghiệp tổng thể.

Nhóm Scrum. Tên gọi chung của một nhóm bao gồm ScrumMaster, Product Owner và Nhóm Phát triển (Development Team).

Nhóm tự tổ chức. Một khái niệm trong Agile và Scrum chỉ ra rằng Nhóm Phát triển phải chịu trách nhiệm quản lý chính bản thân họ ví như các thành viên thấy được những gì phù hợp để hoàn thành một công việc nào đó. Trong trường hợp nhóm không chuyển giao được phần tăng trưởng sản phẩm, tự tổ chức không có nghĩa rằng ScrumMaster hoặc Product Owner không thể hoặc không nên can thiệp vào

Thuật ngữ

theo cách tinh tế và gián tiếp để giúp nhóm tiến bộ.

Phân đoạn (Iteration). Trong Scrum, thuật ngữ này được gọi là Sprint. Đây là khung thời gian cho một chu kỳ lặp trong Scrum, trong thời gian đó Nhóm Phát triển có nhiệm vụ đưa một số User Story đã được chọn vào triển khai để tạo ra một phần tăng trưởng sản phẩm hoàn chỉnh có thể chuyển giao được.

Phòng Quản lý Dự án (Project Management Office – PMO). Một nhóm trong nhiều đơn vị CNTT, họ chịu trách nhiệm về sự phù hợp của CNTT và việc kinh doanh, theo dõi hiệu suất CNTT, chủ yếu là về chi phí dự án, các lợi ích và bàn giao đúng hạn.

Phần tăng trưởng sản phẩm có thể chuyển giao được (Potentially shippable product increment). Một phần tăng trưởng hoàn chỉnh của sản phẩm phần mềm chạy được, được phát triển trong một Sprint và có thể được chuyển giao cho khách hàng.

Product Owner (Chủ sản phẩm). Một thành viên của Nhóm Scrum, người chịu trách nhiệm về tầm nhìn sản phẩm, các hạng mục trong Product Backlog (các yêu cầu) và đánh thứ tự ưu tiên cho các hạng mục này dựa trên quan điểm kinh doanh. Trong một số trường hợp, Product Owner được yêu cầu giữ vai trò hòa giải giữa các đơn vị kinh doanh khác nhau trong công ty để đặt ưu tiên cho các nhu cầu kinh doanh và các yêu cầu đang xung đột.

ScrumBut (tích cực). Thích ứng tốt với Scrum và Scrum đã giúp có được những nhóm tiếp tục tiến bộ bất chấp những khó khăn mà họ gặp phải trong thực tế.

ScrumBut (tiêu cực). Việc áp dụng sai Scrum, thường như một cái cớ để che giấu những điểm yếu của tổ chức.

ScrumMaster. Chuyên gia Scrum, người chịu trách nhiệm hỗ trợ những thành viên còn lại của Nhóm Scrum (và tổ chức) hiểu và áp dụng đúng Scrum cho một dự án.

SMART (kỹ thuật). Kỹ thuật đơn giản và rất hiệu quả trong việc hỗ trợ nhà quản lý và các nhóm xác định mục tiêu mà họ có thể đạt được trong một khoảng thời gian hợp lý và với những nguồn lực sẵn có. SMART viết tắt của các từ Specific (Rõ ràng), Measurable (Đo được), Achievable (Có thể đạt được), Realistic (Thực tế) và Time-Based (Dựa trên thời gian).

Sơ kết Sprint. Cuộc họp mà trong đó nhóm tiến hành trình diễn những gì họ đã xây dựng cho Product Owner và các bên liên quan.

Sprint Backlog. Danh sách tất cả các công việc mà Nhóm Phát triển phải làm trong suốt một Sprint.

Sprint. Một chu kỳ có khung thời gian, hoặc phân đoạn, trong đó Nhóm Phát triển được hỗ trợ để biến một số yêu cầu (User Story) đã được chọn thành phần tăng trưởng sản phẩm có thể chuyển giao được.

Tầm nhìn Kiến trúc. Tương tự như một số người gọi là ý đồ kiến trúc, đây là tầm nhìn ở mức khái quát về kiến trúc, thứ sẽ nhìn thấy một khi tất cả những thành phần chi tiết đã được xác định rõ.

Task board (Bảng công việc). Bảng trắng nơi các công việc và nhiệm vụ của nhóm được thể hiện một cách minh bạch.

Thác nước (Waterfall). Một quy trình phát triển phần mềm tuần tự, trong đó giả định rằng tất cả các yêu cầu cần phải được thu thập trước tiên, tiếp theo mới đến phân tích, rồi đến phát triển và chỉ sau đó mới được kiểm thử trước khi đưa sản phẩm ra sử dụng.

Thẻ story (Story card). Trong các quy trình Agile và Lean là thẻ được sử dụng để mô tả một User Story.

Tốc độ. Số lượng User Story (thường được xác định bằng điểm) mà Nhóm Phát triển có thể hoàn thành trong một Sprint.

Tuyên ngôn Agile. Tài liệu được soạn thảo trong một cuộc họp tại Utah (Hoa Kỳ) năm 2001, tài liệu này đã khởi xướng cho phong trào phát triển phần mềm thích ứng và gọn nhẹ.

User story. Tên gọi chung được sử dụng trong phát triển phần mềm linh hoạt để chỉ các yêu cầu nghiệp vụ của người dùng.

Vai trò của người dùng. Các vai trò được đảm nhiệm bởi những người dùng khác nhau trong suốt quá trình của dự án. Đôi khi một người có thể đảm nhiệm nhiều hơn một vai trò; ví dụ, một người có thể vừa là quản trị hệ thống vừa là người phê duyệt hóa đơn.

THAM KHẢO

Kanban, tác giả David Anderson, xuất bản năm 2010 bởi Blue Hole Press.

ScrumButs Are the Best Part of Scrum (tạm dịch: “ScrumBut là phần tốt nhất của Scrum”), tác giả Jurgen Appelo, đăng tháng 9/2009 (tại địa chỉ <http://www.noop.nl/2009/09/scrumbuts-are-the-best-part-of-scrum.html>).

Managing Agile Projects (tạm dịch: “Quản lý Dự án Linh hoạt”), tác giả Sanjiv Augustine, xuất bản năm 2005 bởi Prentice-Hall.

User Stories Applied: For Agile Software Development (tạm dịch: “Áp dụng User story cho Phát triển Phần mềm Linh hoạt”), tác giả Mike Cohn, xuất bản năm 2004 bởi Upper Saddle River: Addison Wesley.

Agile Estimating & Planning (tạm dịch: “Ước tính và Lập kế hoạch Linh hoạt”), tác giả Mike Cohn, xuất bản năm 2005 bởi Boston: Prentice-Hall.

Succeeding with Agile: Leading a Self-Organizing Team (tạm dịch: “Thành công với Agile: Lãnh đạo một Nhóm tự-tổ chức”), tác giả Mike Cohn, đăng tháng 8/2009 (tại địa chỉ <http://www.informit.com/articles/article.aspx?p=1382538>).

Succeeding with Agile: Software Development Using Scrum (tạm dịch: “Thành công với Agile: Phát triển Phần mềm sử dụng Scrum”), tác giả Mike Cohn, xuất bản năm 2009 bởi Addison-Wesley.

Agile Coaching (tạm dịch: “Huấn luyện Agile”), tác giả Rachel Davies và Liz Sedley, xuất bản năm 2009 bởi The Pragmatic Programmer.

Agile Retrospectives: Making Good Teams Great (tạm dịch: “Cải tiến Agile: Tạo ra những nhóm tuyệt vời”), tác giả Esther Derby và Diana Larson, xuất bản năm 2006 bởi Raleigh: Pragmatic Bookshelf.

“*The New New Product Development Game*” (tạm dịch: “Trò chơi phát triển sản phẩm mới”) của Hirotaka Takeuchi và Ikujiro Nonaka đăng trên *Harvard Business Review* năm 1986.

Tham khảo

Agile Software Development Ecosystems (tạm dịch: “Hệ sinh thái Phát triển Phần mềm Linh hoạt”), tác giả Jim Highsmith, xuất bản năm 2002 bởi Addison-Wesley.

Agile Project Management (tạm dịch: “Quản lý Dự án Linh hoạt”), tác giả Jim Highsmith, xuất bản năm 2004 bởi Addison-Wesley.

Kanban: Just in time at Toyota (tạm dịch: “Kanban: JIT tại Toyota”) của Hiệp hội Quản lý Nhật Bản, được chuyển ngữ và cập nhật phiên bản tiếng Anh bởi Productivity Inc. năm 1989.

Scrum and XP from the Trenches (tạm dịch: “Scrum và XP từ những chiến hào”), tác giả Henrik Kniberg, xuất bản năm 2007 bởi InfoQ.

Five Dysfunctions of Teams (tạm dịch: “Năm dấu hiệu bất ổn của Nhóm”), tác giả Patrick Lencioni, xuất bản năm 2002 bởi Jossey-Bass, công ty A Wiley.

Agile Software Development: Principles, Patterns and Practices (tạm dịch: “Phát triển Phần mềm Linh hoạt: Các nguyên tắc, Mô hình và Phương pháp”), tác giả Robert Martin, xuất bản năm 2005 bởi Upper Saddle River, New Jersey: Pearson Education.

Crossing the Chasm (tạm dịch: “Băng qua vực thẳm”), tác giả Geoffrey Moore, xuất bản năm 2004 bởi Harper Business Essentials.

Influence of Architecture on Team Velocity and Software Quality (tạm dịch: “Ảnh hưởng của Kiến trúc đối với Tốc độ Nhóm và Chất lượng Phần mềm”), của Andrew Pham thuyết trình tại Scrum User Group ở Dallas, tháng 3/2010.

Bài thuyết trình về Scrum tại Khoa Dịch vụ Thông tin Y tế, Đại học Pennsylvania của Andrew Pham, tháng 6/2010.

Lean Software Development (tạm dịch: “Phát triển Phần mềm Tinh gọn”), tác giả Tom và Mary Poppendieck, xuất bản năm 2006 bởi Addison-Wesley Professional.

The Wisdom of Teams: Creating a High Performance Organization (tạm dịch: “Trí tuệ Nhóm: Tạo ra một Tổ chức hiệu năng cao”), tác giả Jon Katzenbach và Douglas Smith, xuất bản năm 1993 bởi Collins Business.

Fearless Change: Patterns for Introducing New Ideas (tạm dịch: “Sự thay đổi dũng cảm: Các mô hình để đưa ra những ý tưởng mới”), tác giả Linda Rising và Mary Lynn Manns, xuất bản năm 2004 bởi Addison-Wesley.

Agile Software Development with Scrum (tạm dịch: “Phát triển Phần mềm Linh hoạt với Scrum”), tác giả Ken Schwaber và Mike Beedle, xuất bản năm 2002 bởi Prentice Hall.

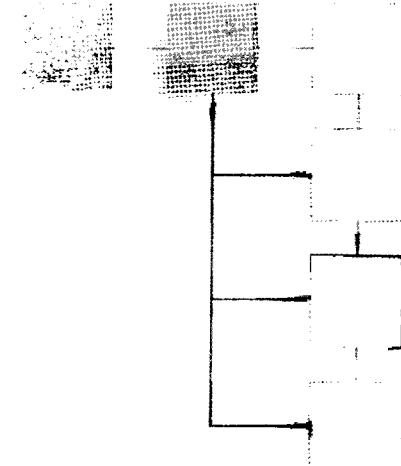
Agile Project Management with Scrum (tạm dịch: “Quản lý Dự án Linh hoạt với Scrum”), tác giả Ken Schwaber, xuất bản năm 2004 bởi Microsoft Press.

Scrum Guide (tạm dịch: “Hướng dẫn Scrum”), tác giả Ken Schwaber và Jeff Sutherland, cập nhật tháng 2/2010 bởi Scrum.org.

Lean-Agile Software Development Achieving Enterprise Agility (tạm dịch: “Phát triển Phần mềm Tinh gọn-Linh hoạt cho toàn bộ Doanh nghiệp”), tác giả Alan Shalloway, Guy Beaver và James Trott, xuất bản năm 2010 bởi Addison-Wesley.

Scrum in Church (tạm dịch: “Scrum trong Nhà thờ”), tác giả Jeff Sutherland, Irene Sutherland và Christine Hegarty, được báo cáo tại Hội thảo Agile 2009.

CHỈ MỤC

- 
- ## A
- ActiveRecord, Ruby on Rails** 172–173
 - Agile Project Management with Scrum** 4, 27, 125
 - Ảnh hưởng của lãnh đạo lên nhóm** 129
 - Ánh xạ Đối tượng-Quan hệ** 172–173
 - An toàn công việc** 123, 129
 - AP (Điểm đánh giá chính)** 63
 - Appelo, Jurgen** 136, 139
 - Array (mảng), Ngôn ngữ Ruby** 169
 - Authlogic, Plugin**
 - Conferous 230
 - Noshster 183
- ## B
- Backlog, Product** 41–54
 - Định nghĩa** 6
 - Quy tắc CUTFIT** 45–47
 - Quy tắc SMART** 42–43
 - Quy trình thu thập trực quan các yêu cầu** 41–47
 - Thu thập yêu cầu cho Product Backlog** 43–45
 - Xác định các bên liên quan và mục tiêu** 41–43
 - Backlog, Sprint** 6–7, 92, 94
- ## C
- BAC (Ngân sách lúc hoàn thành)** 24
 - Bản cập nhật** 34–35
 - Bản chất con người** 116–117
 - Bản khảo sát, Tự đánh giá độ sẵn sàng** 147–150, 163
 - Bên liên quan**
 - Quy tắc SMART** 42–43
 - Quy trình thu thập yêu cầu trực quan** 48–49
 - Xác định** 41–42
 - Biến, Ngôn ngữ Ruby** 168–169
 - Biểu đồ Burndown** 8, 137–138
 - Biểu đồ cột Story** 137–138
 - Build hàng ngày** 143
 - Cách tiếp cận dựa trên các thành phần dữ liệu chung** 87–92, 140
 - Cài tiến Sprint** 9–10
 - Cam kết, của cả hai vai trò Product Owner và ScrumMaster** 141
 - Cá nhân trong nhóm** 116–117
 - Capital One** 135
 - Case Study, Phát triển phần mềm** 167–246.
 - Xem Conferous, Case Study; Noshster, Case Study**

- Ngôn ngữ Ruby** 167–174
- Phát triển Web bằng Ruby on Rails** 174–176
- Ruby on Rails (RoR), Khung làm việc Web** 171–174
- Cá trúc, Ngôn ngữ Ruby** 167
- Cây quản lý người dùng** 50
- CDE (Container, Difference, Exchange)** 128
- Chân thật, Đặc điểm của nhà lãnh đạo** 132
- Chất lượng phần mềm, và Nợ kỹ thuật** 95. *Xem* **Tầm nhìn kiến trúc**
- Chênh lệch chi phí (CV)** 21
- Chênh lệch lúc hoàn thành (VAC)** 24
- Chênh lệnh tiến độ (SV)** 22–23
- Chiến lược, Quản lý kinh doanh cấp cao** 30
- Chi phí**
 - Nguồn nhân lực của dự án** 15–16
 - Thời gian hoàn vốn** 16–17
- Chi phí dự án, Tính toán** 15–16
- Chi phí nguồn nhân lực của dự án Scrum** 15–16
- Chi phí thực hiện**
 - Chênh lệch chi phí** 21
 - Chi số chi phí thực hiện** 21–22
- Công thức** 20

Chỉ mục

Chi phí thực tế (AC) 21, 24
Chi số Chi phí thực hiện (CPI) 21–22, 23
Chi số tiến độ thực hiện (SPI) 23
Chu kỳ Thanh tra và Thích nghi, Scrum 11
Cơ chế giảm thiểu rủi ro, Scrum 10
Cohn, Mike 125–126, 128
Conferous, Case Study 225–246
 Lập kế hoạch phát hành 225–227
 Lập kế hoạch Sprint 225–227
 Mục tiêu và Tầm nhìn sản phẩm 225
 Noshster, Bản phát hành 1-Sprint 1 191–194
 Quy trình thu thập yêu cầu trực quan 225
 Sprint 1 229–239
 Sprint 2 240–246
 Tầm nhìn kiến trúc 225–227
 Ước tính dự án 227–229
Cộng tác, trong Scrum 10.
 Xem Quản lý dự án
Controller, trong MVC
 Conferous, Sprint 1 237–239
 Conferous, Sprint 2 245–246
 Noshster, Bản phát hành 1-Sprint 2 202–203
 Noshster, Bản phát hành 2-Sprint 3 211–213
 Noshster, Bản phát hành 2-Sprint 4 220–222
 Tổng quan 173–174
Convention over configuration, Ruby on Rails 171
Cottmeyer, Mike 136
Có vấn
 Product Owner 103
 Vai trò lãnh đạo 129
CRUD (Tạo, Đọc, Cập nhật, Xóa) 59
Cú pháp, Ngôn ngữ Ruby 167
CV (Chênh lệch Chi phí) 21

D

Dấu chấm phẩy trong ngôn ngữ Ruby 167
Dish partial, Noshster 201
Doanh thu, Thời gian hoàn vốn 16–17
Dòng tiền, Thời gian hoàn vốn 16–17
Don't repeat yourself, Ruby on Rails 171
Ducker, Peter 126
Dự toán để hoàn thành (ETC) 24
Dự toán lúc hoàn thành (EAC) 24
Dự toán ngân sách 23–24
 Chênh lệch lúc hoàn thành 24
 Dự toán để hoàn thành 24
 Dự toán lúc hoàn thành (EAC) 23–24
Dự toán ngân sách dự án 23–24
 Chênh lệch lúc hoàn thành (VAC) 24
 Công thức 20
 Dự toán để hoàn thành (ETC) 24
 Dự toán lúc hoàn thành (EAC) 23–24

D

Đặc điểm của một nhà lãnh đạo chu đáo 132–133
Đánh giá độ sẵn sàng của dự án 145–157
 Cải thiện điểm 155–156
 Công cụ đơn giản 145–151
 Khía cạnh công nghệ 148
 Khía cạnh hạ tầng 148
 Khía cạnh nghiệp vụ 149
 Khía cạnh nhóm 148
 Khía cạnh quy trình 149
 Khía cạnh tổ chức 147
 Tổng quan 163, 165
 Ví dụ 151–156
Đào tạo, Công nghệ 142
Đầu tư, Chọn dự án 16–20
 Giá trị hiện tại thuần (NPV) 18–19
 Mua so với xây dựng 17–18

Thời gian hoàn vốn 16
Tỷ lệ hoàn vốn đầu tư (ROI) 19–20
Điểm chưa hiệu chỉnh (UP) 60
Điểm đã hiệu chỉnh (AP) 63
Điểm hòa vốn 16
Điểm Story không so sánh được, Vấn đề 55–56
Điểm tối đa, Tự đánh giá độ sẵn sàng của dự án 150–151
Điểm tối thiểu, Tự đánh giá độ sẵn sàng của dự án 150
Điểm trung bình, Tự đánh giá độ sẵn sàng của dự án 150–152
Điểm, Tự đánh giá độ sẵn sàng của dự án 147, 150–157
Điều khiển DishesController, Noshster 202–203
Điều khiển
 RestaurantsController, Noshster 211–212
Điều khiển ReviewsController, Noshster 220–222
Điều khiển, trong MVC
 Noshster, Kiểm thử 223
Điều khiển UserController
 Conferous 237, 245–246
 Noshster 191–193
Điều khiển UserSessionsController
 Conferous 237–239
 Noshster 193–194
Độ đo, Bên liên quan 43, 49
Đối tượng, Ngôn ngữ Ruby 168
Được tôn trọng 116

E

EAC (Dự toán lúc hoàn thành) 24
EV (Giá trị thu được)
 Chênh lệch chi phí 21
 Chênh lệnh tiến độ 22–23
 Dự toán ngân sách dự án 23–24

F

Five Dysfunctions of Teams
118, 165

Fixnums 170

Form partial

Conferous 232–233, 244
Noshster 186, 189,
198–199, 206–209,
217–219

G

Giá trị định, Kỹ thuật mua so với xây dựng 17–18

Giai đoạn chuyên biệt trong Scrum 142

Giai đoạn của nhóm

Hình thành 120
Hoạt động thành công 120
Quy chuẩn 120
Sóng gió 120
Thoái trào 120

Giải pháp, Mô hình GROW
131

Giải quyết xung đột

Kỹ năng của ScrumMaster
162
Kỹ thuật 121–122

Giao tiếp với Quản lý CNTT cấp trung 34

Giá trị của đồng tiền theo thời gian 18–19

Giá trị dự kiến (PV) 22–23

Giá trị hiện tại (PV) 18–19

Giá trị hiện tại thuần (NPV)
18–19

Giá trị thu được (EV)

Chênh lệch chi phí 21
Chênh lệch tiến độ 22–23
Dự toán ngân sách dự án
23–24
Định nghĩa 20

Giá trị tương lai (FV) 18

H

Hạng mục Product Backlog (PBI). Xem Ước tính điểm Story; User Story

Định nghĩa 41

Triển khai đồng thời 76

Hash (băng băm), trong ngôn ngữ Ruby 170

Hạ tầng

Kinh nghiệm của nhóm về hạ tầng 142

Tổ chức kiểm thử 111–113

Hệ số thao tác dữ liệu (CRUD) trong Ước tính điểm Story
59

Hệ thống quản lý phiên bản 174, 196

Hệ thống quản lý phiên bản Git 174

Hiểu biết về sản phẩm, Product Owner 101–102

Hiệu suất dự án, Theo dõi 20–24

Chi phí thực hiện 21–22

Dự toán ngân sách dự án 23–24

Tiến độ thực hiện 22–23

Hoàn thành, Định nghĩa 107–109, 142

Hợp. Xem Đứng hàng ngày

cải tiến Sprint 9–10

lập kế hoạch Sprint 7

sơ kết Sprint 9

Vai trò của ScrumMaster trong việc tổ chức họp 161

Hợp đứng hàng ngày 7–8, 96, 140. *Xem Hợp Scrum hàng ngày*

Hợp Scrum hàng ngày 7–8, 96. *Xem Hợp đứng hàng ngày*

Hỗ trợ

Lãnh đạo 128

Product Owner 103

Huấn luyện

bởi Product Owner 103

dễ đạt hiệu quả cao nhất 130–132

Thích ứng với Scrum 140
trong vai trò lãnh đạo 129

Hướng dẫn

bởi Product Owner 103

trong vai trò lãnh đạo 129

I

InformIT 125

K

Kanban, Kỹ thuật 71

Khả năng cộng tác, Hệ thống quản lý phiên bản 174

Khẳng định (assertion), trong kiểm thử 175–176

Khía cạnh công nghệ

Thích ứng với Scrum 142

Tự đánh giá độ sẵn sàng của dự án Scrum 146, 148, 153

Ước tính điểm Story 62

Khía cạnh hạ tầng

Thích ứng với Scrum 140

Tự đánh giá độ sẵn sàng của dự án 145, 148, 153

Khía cạnh hạ tầng phát triển 61

Khía cạnh môi trường (ED)

Thảo luận chung 60–62

Ví dụ 63

Khía cạnh nghiệp vụ

Product Owner, không có người đảm nhận 143

Thích ứng với Scrum 143

Tự đánh giá độ sẵn sàng của dự án 146, 149

Ước tính điểm Story 62

Khía cạnh nhóm

Thích ứng với Scrum 141

Tự đánh giá độ sẵn sàng của dự án 145, 148, 153

Ước tính điểm Story 61

Khía cạnh quy trình

Thích ứng với Scrum 142–143

Tự đánh giá độ sẵn sàng của dự án 146, 149, 154

Ước tính điểm Story 62

Khía cạnh tổ chức

Thích ứng với Scrum 137–140

trong ước tính điểm Story 61

Tự đánh giá độ sẵn sàng của dự án 145, 147, 152

Khung làm việc kiểm thử Test::Unit 175–176

Chỉ mục

- Khung làm việc Web, Ruby on Rails 171–174**
- MVC (Model - Mô hình, View - Khung nhìn, Control - Điều khiển) 171–174
 - Thảo luận chung 171
- Khung nhìn**
- Conferous, Case Study 231–232, 241–245
 - Noshster, Case Study 185–191, 197–201, 206–211, 216–220
 - Tổng quan 171
- Khung nhìn Đăng ký 231–233**
- Noshster, Case Study 185–186
- Khung nhìn Đăng nhập**
- Conferous, Case Study 233
 - Noshster, Case Study 186–187
- Khung nhìn Duyệt các bản ghi âm, Conferous 241–243**
- Khung nhìn Duyệt người dùng, Noshster 190–191**
- Khung nhìn Duyệt phòng, Conferous 235–236**
- Khung nhìn Hủy tài khoản, Conferous 244–245**
- Khung nhìn Sửa hồ sơ**
- Conferous, Case Study 243–244
 - Noshster, Case Study 188–189
- Khung nhìn Thêm phòng, Conferous 234**
- Khung nhìn Xóa phòng, Conferous 235**
- Kiểm thử 105–114**
- chấp nhận 108–110
 - chấp nhận người dùng 108–110, 176, 224
 - chức năng 175, 223
 - đơn vị 107–108, 175, 222
 - Hoàn thành, Định nghĩa 107–109
 - hồi quy 175
 - Khung làm việc Kiểm thử Test::Unit 176
 - Noshster, Case Study 222–224
 - Phát triển Web với Ruby on Rails 174–176
- Thích ứng với Scrum 140
 - thủ công 110, 175
 - tích hợp 108, 109, 111, 175, 223
 - tích hợp liên tục 111, 140, 143
 - Tổ chức hạ tầng 111–113
 - Tổng quan 165
 - tự động 105, 107, 110–111, 140, 175–176
- Kiểm thử chấp nhận người dùng**
- Hoàn thành, Định nghĩa 108–110
- Kiểm thử cháp nhận người dùng**
- Noshster, Case Study 224
- Phát triển Web với Ruby on Rails 176**
- Kiểm thử đơn vị**
- Định nghĩa 175
 - Hoàn thành, Định nghĩa 107–108
 - Noshster, Case Study 222–223
 - trong TDD 141
- Kiểm thử tự động**
- Hạ tầng kiểm thử 140
 - Phát triển Web bằng RoR 174–176
- Kiểm tra kết quả, trong mô hình GROW 132**
- Kiểm tra tính hợp lệ người dùng 183**
- Kiến thức lý thuyết về Scrum, ScrumMaster 160**
- Kiến thức thực hành Scrum, ScrumMaster 160–161**
- Kiến thức về Scrum, ScrumMaster 160**
- Kiến trúc**
- Định nghĩa 67–68
 - Doanh nghiệp 35–38, 73–74, 80, 83
 - Dữ liệu 35–39, 73–74, 78–79, 180, 181
 - Phần mềm 164
- Kiến trúc ba tầng 171**
- Kiến trúc doanh nghiệp (EA) 35–38, 73–74, 80, 83**
- Kiến trúc dữ liệu**
- Kiến trúc doanh nghiệp 35–38
 - Noshster, Case Study 180
- Tầm nhìn kiến trúc 73–74, 78–80
- Kiến trúc dữ liệu theo chiều dọc 74, 79**
- Kiến trúc dữ liệu theo chiều ngang 74, 78**
- Kiến trúc hướng dịch vụ (SOA) 35–36**
- Kiến trúc phần mềm 164**
- Kiểu dữ liệu, Ngôn ngữ Ruby 169–170**
- Kilmann, Ralph 121**
- Kniberg, Henrik 6, 107**
- Kỹ năng giao tiếp**
- Product Owner 103
 - ScrumMaster 161–162
- Kỹ năng phát triển nhân lực, ScrumMaster 162**
- Kỹ năng thuyết trình, ScrumMaster 161**
- Kỹ năng tốt về mặt tổ chức, ScrumMaster 161**
- Kỹ thuật 5W 101–102**
- Kỹ thuật cạnh tranh (Competitive - COMPE) 121**
- Kỹ thuật cộng tác (Collaborating - COLLA) 121**
- Kỹ thuật để giải quyết xung đột 121–122**
- Kỹ thuật điều tiết (Accommodating - ACCO) 121–122**
- Kỹ thuật né tránh (Avoidance - AVOID) 121**
- Kỹ thuật Rừng và Cây**
- Conferous, Case Study 225–226
 - Noshster, Case Study 177
 - Tổng quan 43–47
 - Ví dụ 48–52
- Kỹ thuật thỏa hiệp (Compromise - COMP) 121–122**
- Kỳ vọng của các bên liên quan, Sự quản lý của Product Owner 101**

L

- Lãi suất 18–19**
- Làm việc nhóm 115–124**

Các giai đoạn của nhóm 120	Tỷ lệ hoàn vốn đầu tư 19	Sprint 2 241
Các loại tính cách Keirsey 118–120	Lớp, Ngôn ngữ Ruby 168	Mô hình Restaurant, Noshster 205, 215
Cá nhân 116–117	Lược đồ hình sao 80	Mô hình Review, Noshster 213
Điều kiện tuyệt vời 122–123	Lưu bằng ActiveRecord 172	Mô hình Room, Conferous 231, 241
Lòng tin 123	M	Mô hình, trong MVC
Nhóm, Định nghĩa 118	Mã	Conferous, Sprint 1 230
Tập hợp 117	Quản lý phiên bản 174	Conferous, Sprint 2 241
Tổng quan 165	Tái sử dụng trong Ruby on Rails 171	Noshster, Bản phát hành 1-Sprint 1 183, 185
Lãnh đạo 125–134	Mã hóa mật khẩu 183	Noshster, Bản phát hành 1-Sprint 2 197
Huấn luyện với mô hình GROW 130	Ma trận	Noshster, Bản phát hành 2-Sprint 3 205
so sánh với quản lý 125–130	Giải quyết xung đột 121–122	Noshster, Bản phát hành 2-Sprint 4 213–216
Tổng quan 165	Quản lý bên liên quan 101	Noshster, Kiểm thử 222–223
Lãnh đạo kiểu phục vụ	Tự đánh giá độ sẵn sàng với Scrum 146, 150–152	Tổng quan 172–173
Phương diện nội bộ nhóm của vai trò lãnh đạo 128–129	Ước tính 155, 156	Mô hình UserSession
Product Owner 103	Ước tính điểm Story 64	Conferous 231
ScrumMaster 139, 161	Ưu tiên hóa kinh doanh CNTT 32	Noshster 185
Lập kế hoạch phát hành	Ma trận quản lý bên liên quan 101	Môi trường phát triển 111–113
Conferous, Case Study 225–227	Ma trận ưu tiên hóa kinh doanh CNTT 32	Môi trường vận hành thực tế 111–112
Noshster, Case Study 177–181	Mô hình Dish, Noshster 197, 215, 222	Mua so với Xây dựng 17–18
Tầm nhìn kiến trúc 85–95	Mô hình dữ liệu	Mục tiêu
Tổng quan 6	Conferous, Sprint 1 229–230	Bản phát hành 92–93
Lập kế hoạch Sprint	Conferous, Sprint 2 240–241	Bên liên quan 41–43, 48–49
Conferous, Case Study 225–227	Noshster, Bản phát hành 1-Sprint 1 183–184	có thể đạt được 43
Noshster, Case Study 177–181	Noshster, Bản phát hành 1-Sprint 2 195–196	đo được 42
Tầm nhìn kiến trúc 85–95	Noshster, Bản phát hành 2-Sprint 3 204–205	dựa trên thời gian 43
Tổng quan 7–8	Noshster, Bản phát hành 2-Sprint 4 213–215	Mô hình GROW mở rộng 130–132
Lát cắt dọc 90–92, 177–178, 179–180	Mô hình-Giao diện-Điều khiển. Xem Điều khiển, trong MVC; Mô hình, trong MVC; Khung nhìn, trong MVC	Quản lý kinh doanh 30
Lát cắt ngang	Mô hình GROW (Goal - Mục tiêu, Reality - Thực tại, Options - Giải pháp, và Will - Ý chí) 130–132	Quy tắc SMART 42–43
Conferous, Case Study 225–226, 228	Mô hình GROW mở rộng 130–132	rõ ràng 42
so sánh với Lát cắt dọc 178	Mô hình Recording, Conferous	sản phẩm 70, 101–102, 128, 176, 225
Tổng quan 88–90, 93		Sprint 92–93
Lencioni, Patrick 118, 165		thực tế 43
Loại tương tác trong ước tính điểm Story 57		trong tập hợp người (group) 117
Lợi nhuận, Tính toán trong dự án 16–20		Mục tiêu và tầm nhìn của sản điện tử
Giá trị hiện tại thuần 18		Conferous, Case Study 225
Mua so với xây dựng 17–18		Noshster, Case Study 176
Thời gian hoàn vốn 16–17		Product Owner 101–102, 128

Chỉ mục

- Tầm nhìn kiến trúc 70
- MVC (Mô hình-Khung nhìn-Điều khiển)**
- Conferous, Sprint 1 230–240
 - Conferous, Sprint 2 241–246
 - Noshster, Bản phát hành 1-Sprint 2 197–203
 - Noshster, bản phát hành 2-Sprint 3 205–212
 - Noshster, Bản phát hành 2-Sprint 4 213–222
 - Tổng quan 171–174
- N**
- Nền văn hóa khác với phương Tây, Planning Poker** 56
- Ngân sách lúc hoàn thành (BAC)** 24
- Ngôn ngữ lập trình hướng đối tượng** 168
- Ngôn ngữ Ruby** 167–171
- Biển 168
 - Cấu trúc 167–168
 - Cú pháp 167–168
 - Đối tượng 168
 - Khieu dữ liệu 169
 - Lớp 168
 - MVC 171–174
 - Phương thức 170
- Nhà quản lý chức năng**
- Can thiệp 129–130
 - là ScrumMaster 139–140
- Nhà tổ chức, Product Owner** 103
- Nhóm con** 95–96
- Nhóm, Định nghĩa** 118
- Nhóm hạ tầng** 143
- Nhóm hiệu suất cao** 118, 123
- Nhóm hiệu suất tầm thường** 123
- Nhóm hiệu suất thấp** 118, 123
- Nhóm hiệu suất trung bình** 123
- Nhóm phát triển**
- Gợi ý cho Product Owner 85
 - Product Owner luôn sẵn sàng 102
- Quản lý 27–28, 125–126
- TDD 141
- Nhóm Scrum. Xem Nhóm phát triển; Lãnh đạo nhóm; Làm việc nhóm**
- Kích thước 140
- Kỹ năng giải quyết xung đột của ScrumMaster 162
- Lập kế hoạch phát hành 85–95
- Lập kế hoạch Sprint 85–95
- Phát triển phần mềm song song 95–97
- Product Owner tích cực ứng hộ 103
- Quản lý dự án 27, 29
- Quy trình ước tính dựa trên tiêu chí khách quan 63
- Tầm nhìn kiến trúc 85–95
- Thảo luận chung 5–7
- Trách nhiệm 9
- Vấn đề với điểm Story không so sánh được 55–56
- Nợ kỹ thuật** 93, 95
- Nonaka, Ikujiro** 4
- Noshster, Case Study** 176–225
- Bản phát hành 1-Sprint 1 183–194
 - Bản Phát hành 1-Sprint 2 194–203
 - Bản phát hành 2-Sprint 3 203–213
 - Bản phát hành 2-Sprint 4 213–222
 - Kiểm thử 222–224
 - Kỹ thuật 5W 101–102
 - Lập kế hoạch phát hành 177–181
 - Lập kế hoạch Sprint 177–181
 - Tầm nhìn kiến trúc 177–182
 - Tầm nhìn và mục tiêu của sản phẩm 176
 - Thu thập yêu cầu trực quan 177
 - Ước tính dự án 182
- NPV (Giá trị hiện tại thuần)** 18–19
- P**
- PBI (Hạng mục Product Backlog). Xem Ước tính điểm Story; User Story**
- Định nghĩa 41
- Triển khai đồng thời 76
- Phản chất của Product Owner**
- Kỹ năng giao tiếp 103
 - Là nhà tổ chức giỏi 103
 - Lãnh đạo kiểu phục vụ 103
 - Quản lý kỳ vọng của các bên liên quan và độ ưu tiên 101
 - Sẵn sàng 102
- Tầm nhìn và hiểu biết về sản phẩm 101–102
- Thu thập yêu cầu cho Product Backlog 102
- Tổng quan 99–100
- Phản chất của Scrum Master** 159–162
- Khả năng lãnh đạo kiểu phục vụ 161
 - Kiến thức về Scrum 160
 - Kỹ năng giải quyết xung đột 162
 - Kỹ năng thuyết trình 161–162
 - Kỹ năng về giao tiếp 161
 - Kỹ năng về phát triển nhân lực 162
 - Kỹ năng về tổ chức 161
- Phản chênh lệch giá, Mua so với xây dựng** 17–18
- Phân nhóm User story** 75–76
- Phân tích và Thiết kế hướng đối tượng (Object-Oriented Analysis and Design - OOAD), UML** 68
- Phát triển hướng kiểm thử (TDD)** 141
- Phát triển phần mềm đồng thời (song song)**
- Hệ thống quản lý phiên bản 174
 - Noshter 195–196
 - Tầm nhìn kiến trúc 71, 95–97
- Phát triển phần mềm và quản lý dự án linh hoạt** 1–13
- Áp dụng vào các dự án thực tế 163–165

- H**
- Hiệu quả của Agile và Scrum 10–13
 - Nền tảng 2–4
- P**
- Phát triển phần mềm với Scrum 1–13.** *Xem Case Study, Phát triển phần mềm; Áp dụng quản lý dự án vào tình huống thực tế, 163–165*
 - Hiệu quả của Agile và Scrum 10–13
 - Nền tảng của Agile 2–4
 - Nguồn gốc của Scrum 4
 - Quy trình Scrum 4–10
 - Phát triển Web bằng Ruby on Rails 174–176**
 - Kiểm thử 174–176
 - Quản lý Phiên bản Git 174
 - Phi hàng tháng, Kỹ thuật mua so với xây dựng 17–18**
 - Phòng hội đàm.**
 - Xem Conferous, Case Study*
 - Phương diện bên ngoài của vai trò lãnh đạo nhóm 127–130**
 - Phương diện nội bộ nhóm của lãnh đạo 127–129
 - Phương thức attr_accessor, Ngôn ngữ Ruby 169**
 - Phương thức new, Ruby 168**
 - Phương thức, Ngôn ngữ Ruby 170–171**
 - Planning Poker 55–56**
 - PMO (Phòng Quản lý chương trình) 31–32**
 - PPS (Điểm cho mỗi Story) 63**
 - Product Backlog 41–54**
 - Định nghĩa 6
 - Quy tắc CUTFIT 45–47
 - Quy tắc SMART 42–43
 - Quy trình thu thập các yêu cầu trực quan 41–47
 - Thu thập yêu cầu cho Product Backlog 43–45
 - Ví dụ 48–52
 - Xác định bên liên quan và mục tiêu 41–42
 - Product Owner 99–104**
 - Cộng tác trong việc lập kế hoạch phát hành và lập kế hoạch Sprint 97
 - Định nghĩa 5–6
- G**
- Giải quyết xung đột 122
 - Gợi ý từ nhóm phát triển 85
 - Kỹ năng giao tiếp 103
 - Kỹ vọng của các bên liên quan và độ ưu tiên, Quản lý 101
 - Là nhà tổ chức giới 103
 - Lãnh đạo kiểu phục vụ 103
 - Là ScrumMaster 141
 - Product Backlog, Kỹ năng thu thập yêu cầu 102
 - Quản lý dự án 27, 29
 - Quản lý hay lãnh đạo 125–130
 - Quy tắc CUTFIT 44
 - Sẵn sàng 102–103
 - Sơ kết Sprint 9
 - Sự an toàn công việc của các thành viên trong nhóm 123
 - Tầm nhìn kiến trúc 72
 - Tầm nhìn và hiểu biết về sản phẩm 101–102
 - Tổng quan 164
 - Trách nhiệm 9
 - Vai trò trợ giúp Product Owner của ScrumMaster 161
- PV (Giá trị dự kiến) 22**
- PV (Giá trị hiện tại) 18–19**
- Q**
- Quản lý CNTT cấp cao 31–32**
- Quản lý CNTT cấp trung 32–39**
 - Đảm bảo chất lượng 34
 - Kiến trúc doanh nghiệp (EA) 35–38
 - Quản lý vận hành 34–35
- Quản lý dự án 1–13, 27–40, 125–134.** *Xem Thích ứng với Scrum*
- Biến người quản lý thành đồng minh 38–39
 - Đặc điểm của một nhà lãnh đạo và quản lý chu đáo 132–133
 - Hiệu quả của Agile và Scrum 10–13
 - Huấn luyện với mô hình GROW 130–132
- Kh**
- Khả năng thích ứng 12
 - Nền tảng Agile 2–4
 - Nguồn gốc của Scrum 4–5
 - Phòng quản lý chương trình 31–32
 - Quản lý CNTT cấp trung 32–39
 - Quản lý kinh doanh cấp cao 28–31
 - Quan tâm đến vấn đề tài chính 163
 - So sánh với lãnh đạo 125–134
 - Tổng quan 165
 - Quản lý dự án phần mềm.** *Xem Quản lý dự án*
 - Quản lý kinh doanh**
 - đặt nặng vấn đề tài chính 163
 - Làm việc với quản lý kinh doanh cấp cao 28–31
 - Quản lý kinh doanh cấp cao 28–31**
 - Quản lý theo kiểu mệnh lệnh và kiểm soát 139**
 - Quản lý vận hành, IT 32–39**
 - Đảm bảo chất lượng 34
 - Kiến trúc doanh nghiệp (EA) 35
 - Mỗi quan tâm 32–34
 - Quản lý vận hành 34–35
 - Quan tâm**
 - là đặc điểm của Lãnh đạo và Quản lý 133–134
 - trong làm việc nhóm 123
 - Quản trị, Dự án CNTT, PMO 32–33**
 - Quá trình thay đổi 32–35**
 - Quy tắc CUTFIT 45–47**
 - Quy tắc nghiệp vụ trong ước tính điểm Story 57–58**
 - Quy tắc SMART (Rõ ràng, Đòi được, Có thể đạt được, Thực tế, Dựa trên thời gian) 42–43, 131**
 - Quy trình làm phần mềm thác nước 105–106, 143**
 - Quy trình Scrum 5–10**
 - Quy trình thu thập yêu cầu 41–47**
 - Conferous, Case Study 225–226*

Chỉ mục

- Kỹ năng của Product Owner
102
- Noshster, Case Study 177
- Quy tắc CUTFIT 45–47
- Quy tắc SMART 42–43
- Tầm nhìn kiến trúc 70
- Thích ứng với Scrum 143
- Tổng quan 43–47, 164
- Xác định bên liên quan và
mục tiêu 41–42
- Quy trình ước tính dựa theo
tiêu chí**
- Conferous, Case Study
227–229
- Noshster, Case Study 182
- Thảo luận chung 56–65
- Tự đánh giá độ sẵn sàng
154–155
- Ví dụ 63–65
- R**
- Review partial, Noshster
219–220
- ROI (Tỷ lệ hoàn vốn đầu tư)
19
- Room partial, Conferous
242–243
- Ruby on Rails (RoR)
- Kiểm thử 174–176
- Là khung làm việc Web
171–174
- MVC 171–174
- Phát triển Web bằng RoR
174–176
- Quản lý phiên bản Git 174
- S**
- Schwaber, Ken 4–5, 27, 125,
136
- Scrum 1–14**
- Áp dụng vào các dự án thực
tế 163–165
- Hiệu quả trong quản lý dự
án 10–13
- Nguồn gốc 4
- Thảo luận chung 4–10
- Scrum and the Perfect Storm,
Bài viết 5**
- Scrum and XP from the
Trenches 6, 107**
- ScrumBut 136–137, 142.
Xem Thích ứng với Scrum
- ScrumBut tích cực 136–137
- ScrumBut tiêu cực 136–137
- ScrumBut tốt 136–137
- ScrumBut xấu 136–137
- Scrum in Church 136, 140**
- Scrum Master 159–162**
- An toàn công việc của các
thành viên nhóm 123
- Biểu đồ Burndown 8
- cộng tác trong việc lập kế
hoạch phát hành và lập
kế hoạch Sprint 97
- đồng thời là Product Owner
141–142
- Giải quyết xung đột 122,
162
- Kiến thức về Scrum 160
- Kỹ năng giao tiếp 161
- Kỹ năng mặt tổ chức 161
- Kỹ năng phát triển nhân
lực 162
- Kỹ năng thuyết trình
161–162
- Quản lý chức năng chính là
Scrum Master 139–140
- Quản lý dự án 27, 29
- Quản lý và Lãnh đạo
125–130
- Sơ kết Sprint 9
- Thích ứng với Scrum khi
thiếu Scrum Master 139
- Tố chất lãnh đạo kiểu phục
vụ 161
- Tổng quan 165
- SDLC (Vòng đời phát triển
phần mềm) 143**
- SOA (Kiến trúc hướng dịch
vụ) 35–36**
- Sơ kết Sprint 9
- Số nguyên trong ngôn ngữ
Ruby 170**
- Số trong ngôn ngữ Ruby 170**
- Sprint. Xem** Conferous, Case
Study; Noshster, Case
Study
- Định nghĩa 7–8
- Độ dài 142
- Sprint 4 tuần 142**
- Sprint Backlog 6–7, 92, 94**
- String (Chuỗi) trong ngôn ngữ**
- Ruby 169
- Succeeding with Agile Using
Scrum 128**
- Supervisor (Giám sát viên),
Các loại tính cách Keirsey
120**
- Sự sẵn sàng**
- của Lãnh đạo/Quản lý 133
- Product Owner 102
- Sutherland, Arline C. 136, 140**
- Sutherland, Jeff 5, 136, 140**
- Symbol trong ngôn ngữ Ruby
169**
- T**
- Tài chính 15–26**
- Chi phí thực hiện 21–22
- Chọn dự án đầu tư 16–20
- Dự toán ngân sách dự án
23–24
- Giá trị hiện tại thuần 18–19
- Mua so với Xây dựng
17–18
- Theo dõi hiệu suất dự án
20–24
- Thời gian hoàn vốn 16–17
- Tiến độ thực hiện 22–23
- Tính chi phí dự án 15–16
- Tỷ lệ hoàn vốn đầu tư
19–20
- Tái sử dụng mã nguồn trong
Ruby on Rails 171**
- Takeuchi, Hirotaka 4**
- Tầm nhìn kiến trúc 67–84**
- Conferous, Case Study
225–227
- Lập kế hoạch phát hành
và lập kế hoạch Sprint
85–95
- Lợi ích 73–83
- Noshster, Case Study
176–181
- Phát triển phần mềm song
song 95–97
- Tầm quan trọng 69
- Tổng quan 164
- Xác định 70–73
- Tầm nhìn sản phẩm**
- Conferous, Case Study 225
- Noshster, Case Study 176
- Product Owner 101–102,

- 128
- Tầm nhìn kiến trúc 70
- Tầm nhìn, trong vai trò lãnh đạo 128
- Tầng Quản lý Dữ liệu Chủ (MDM) 36, 37–39
- Tập hợp, làm việc nhóm 117
- Task Board 7
- TDD (Phát triển Hướng Kiểm thử) 141
- Test case, trong TDD 141
- Tham gia, Product Owner 103
- Thành thật, Lãnh đạo 132
- Tháp nhu cầu của Maslow 116–117
- Thất bại với Scrum 2
- Thay đổi mức ưu tiên sản phẩm 72, 77
- Thẻ điểm cân bằng 30
- The New New Product Development Game 4
- Theo dõi hiệu suất dự án 20–24
- Chi phí thực hiện 21–22
 - Dự toán ngân sách dự án 23–24
 - Tiến độ thực hiện 22–23
 - Tổng quan 20
- Thẻ story 51–52, 63, 65
- Thích ứng, Quản lý dự án 12
- Thích ứng với Scrum 135–144
- Khía cạnh công nghệ 142
 - Khía cạnh hạ tầng 140–141
 - Khía cạnh nghiệp vụ 143
 - Khía cạnh nhóm 141–142
 - Khía cạnh quy trình 142–143
 - Khía cạnh tổ chức 137–140
 - Không triển khai “ScrumBut” tiêu cực 136–137
 - Ví dụ về các tình huống thích ứng Scrum 137–143
- Thời gian hoàn vốn 16–17
- Thomas, Kenneth 121
- Thừa nhận lỗi lầm, Phẩm chất của lãnh đạo 132
- Thực tại trong mô hình GROW 131
- Thực thể dữ liệu trong Ước tính điểm Story 58–59
- Thực thể nghiệp vụ trong Ước tính điểm Story 59
- Thước đo về khả năng lãnh đạo kiểu phục vụ của ScrumMaster và Product Owner với nội bộ nhóm 128–129
- Thu thập yêu cầu 41–47
- Conferous, Case Study 225–226
 - Kỹ năng thu thập yêu cầu của Product Owner 102
 - Noshster, Case Study 177
 - Quy tắc CUTFIT 45–47
 - Quy tắc SMART 42–43
 - Tầm nhìn kiến trúc 70
 - Thích ứng với Scrum 135–144
 - Thu thập yêu cầu cho Product Backlog 43–45
 - Tổng quan 164
 - Xác định các bên liên quan và mục tiêu 41–43
- Tiến độ thực hiện 22–23
- Chênh lệnh tiến độ 22–23
 - Chỉ số tiến độ thực hiện 23
 - Công thức 20
- Tính cách Keirsey 118–120
- Architect 119
 - Champion 119
 - Composer 119
 - Counselor 118
 - Crafter 119
 - Field Marshal 120
 - Healer (Người hòa giải) 119
 - Inspector (Thanh tra) 118
 - Inventor (Nhà sáng chế) 119
 - Mastermind (Quân sư hay người vạch ra kế hoạch) 118
 - Performer (Diễn viên) 119
 - Promoter (Người kích động) 119
 - Protector (Người bảo vệ) 118
 - Provider (Nhà cung cấp) 120
 - Teacher (Giáo viên) 120
- Tính chi phí dự án 15–16
- Tính năng Thêm món ăn, Noshster
- Kiểm thử 222–223
- Tổng quan 197–198
- Tốc độ nhóm
- Định nghĩa 15–16
 - Kiểm thử tự động 105, 107
 - Tầm nhìn kiến trúc 68–69, 164
- Tốc độ, trong ROI 19
- Tôn trọng
- Lãnh đạo 132
 - Trong làm việc nhóm 122
- Trách nhiệm, ở cấp độ tập hợp (group) 117
- Trang Cảnh sửa đánh giá, Noshster 218–219
- Trang Đăng xuất, Noshster 187
- Trang Duyệt món ăn, Noshster 201
- Trang Duyệt nhà hàng, Noshster 210–211
- Trang Sửa món ăn, Noshster 198–199
- Trang Sửa nhà hàng, Noshster 207–209
- Trang Thêm đánh giá, Noshster 216–218
- Trang Thêm nhà hàng, Noshster 206
- Trang Xem món ăn, Noshster 200
- Trang Xem nhà hàng, Noshster 209–210
- Trang Xem/Xóa hồ sơ, Noshster 188
- Trang Xóa đánh giá, Noshster 219–220
- Trang Xóa hồ sơ, Noshster 190
- Trạng thái dự án, Báo cáo 137–138
- Triển khai toàn doanh nghiệp 55–56
- Quy trình Ước tính dựa theo tiêu chí khách quan 56–65, 182, 227–229
 - Vấn đề không so sánh được 55–56
 - Vấn đề về Văn hóa, Planning Poker 56
 - Ví dụ 63–65
- Trò ngại, được các nhà lãnh đạo loại bỏ 128

Chỉ mục

Tuckman, Bruce 120

Tự đánh giá độ sẵn sàng của dự án 145–157

Cải thiện điểm 155–156

Công cụ đơn giản 145–151

Khía cạnh công nghệ 148

Khía cạnh hạch tầng 148

Khía cạnh nghiệp vụ 149

Khía cạnh nhóm 148

Khía cạnh quy trình 149

Khía cạnh tổ chức 147

Tổng quan 163, 165

Ví dụ 151–156

Tư duy cởi mở

của lãnh đạo 132

trong làm việc nhóm 123

Tự khẳng định bản thân 116

Tự tổ chức, Nhóm 118, 139

Tuyên ngôn Agile 2–3

Tuyên ngôn Tương hỗ 4

Tỷ lệ hoàn vốn đầu tư (ROI) 19–20

Tỷ lệ hoàn vốn nội bộ (IRR) 19–20

Tỷ lệ lạm phát 18

Tỷ suất lợi nhuận biên trong ROI 20

U

Üng hộ, Product Owner 103

UP (Điểm chưa hiệu chỉnh) 60

User partial, Noshster 191

User Story. Xem **Ước tính điểm Story; Quy trình thu thập yêu cầu trực quan**

Báo cáo tiến độ 137–138

Conferous, Case Study 229–230

Conferous, Case Study

240–241

Hợp lập kế hoạch Sprint 7

Kỹ năng thu thập của Product Owner 102

Noshster, Case Study 183, 195–196, 204, 213

Phát triển phần mềm song song 95–96

Quy tắc CUTFIT 45–47

Tầm nhìn kiến trúc 69–77, 86–88

Thu thập 43–45

Ví dụ 51–52

Vấn đề về Văn hóa, Planning Poker 56

Văn hóa Châu Á, Planning Poker 56

Ví dụ về hệ thống thư viện trung tâm 86–94

Ví dụ về phần mềm đặt phòng 48–52

Ví dụ về phát triển phần mềm, Kỹ thuật rừng và cây 48–52

Vòng đời phát triển phần mềm (SDLC) 11, 143

U

Üng dụng, Khía cạnh 57

Ước tính điểm Story so sánh được 55–66

Conferous, Case Study 227–229

Noshster, Case Study 182

Quy trình ước tính dựa theo tiêu chí khách quan 56–65, 182, 227–229

Tổng quan 164

Tự đánh giá độ sẵn sàng 154–155

Vấn đề không so sánh được 55–56

Vấn đề về Văn hóa, Planning Poker 56

Ví dụ 63–65

V

VAC (Chênh lệch lúc hoàn thành) 24

Vai trò của Scrum Master trong suốt Sprint 161

X

Xóa bằng cách dùng ActiveRecord 172

Y

Ý chí, trong mô hình GROW 132

Yêu cầu

độc lập 46

khả thi 46

không mơ hồ 46

kiểm thử được 46

nghiệp vụ 58

nhất quán 45

theo dõi được 46

ĐÔI NÉT VỀ FPT POLYTECHNIC

Thành lập tháng 07/2010, FPT Polytechnic thuộc Đại học FPT, đào tạo Hệ Cao đẳng thực hành và cấp bằng Cao đẳng nghề theo Quyết định của Tổng cục dạy nghề.

SỨ MỆNH

FPT Polytechnic ra đời với sứ mệnh cung cấp dịch vụ đào tạo tốt trên các tiêu chí: phù hợp với năng lực học tập của sinh viên; đáp ứng nhu cầu lớn của doanh nghiệp và cung cấp dịch vụ đào tạo chuẩn mực dựa trên các chuẩn đã được công nhận.

Phù hợp với năng lực học tập của sinh viên

Tất cả sinh viên đều có quyền học tập và có quyền được cung cấp một học vấn, kỹ năng phù hợp với năng lực. FPT Polytechnic cung cấp một chương trình học tập thiên về thực hành với mục tiêu chỉ cần sinh viên chăm chỉ, có ý thức học hỏi, cầu tiến thì sẽ đáp ứng được nhu cầu việc làm của doanh nghiệp.

Đáp ứng nhu cầu lớn của doanh nghiệp

Các chuyên ngành đào tạo của FPT Polytechnic đều nhắm tới nhu cầu xã hội lớn. Với nhận định Việt Nam có hàng trăm nghìn doanh nghiệp, mỗi doanh nghiệp đều cần có nhân viên kế toán, nhân viên tiếp thị và bán hàng. Các doanh nghiệp đều cần có website quảng bá sản phẩm và giao dịch do đó đều cần một nhân viên thiết kế, quản trị website. Ngoài ra, doanh nghiệp có hệ thống máy tính đều cần nhân viên để xây dựng các ứng dụng hữu ích và đảm bảo hệ thống được vận hành hiệu quả.

Cung cấp dịch vụ đào tạo chuẩn mực

Chương trình đào tạo của FPT Polytechnic tuân theo chuẩn khung chương trình của BTEC, Vương quốc Anh. Sách giáo trình, tài liệu học tập được chuyển ngữ sang Tiếng Việt từ các bộ sách uy tín của các Nhà xuất bản lớn trên thế giới như Pearson, Cengage, McGraw-Hill, ... Học liệu được các cán bộ có uy tín của Trường Đại học FPT thiết kế, biên tập và chuyển tải trên việc áp dụng hệ thống công nghệ thông tin.

TRIẾT LÝ ĐÀO TẠO

FPT Polytechnic áp dụng triết lý đào tạo “Thực học – Thực nghiệp” với việc coi người học như là nhân viên. Trải nghiệm học tập sẽ được coi như trải nghiệm làm việc nhằm giúp cho doanh nghiệp làm quen được với công việc thực tiễn và phương pháp giải quyết vấn đề dựa trên công việc. Phương pháp đào tạo qua dự án “project based training” sẽ đưa các yêu cầu thực tế của công việc truyền đạt dưới dạng dự án giao cho sinh viên. Việc kỷ luật học tập cũng được áp dụng chặt chẽ cùng với các khóa đào tạo kỹ năng mềm nhằm đảm bảo sinh viên ra trường có được ý thức cũng như kỹ năng mềm trong công việc.

CHIẾN LƯỢC TRONG VIỆC PHÁT TRIỂN GIÁO TRÌNH

Hiện nay FPT Polytechnic đã có bản quyền dịch và phát hành nhiều bộ sách giáo trình cho các chuyên ngành liên quan đến Công nghệ thông tin, Kế toán và Quản trị kinh doanh. Ngoài ra, FPT Polytechnic cũng đã phát triển các gói học liệu cho các giáo trình này. FPT Polytechnic sẵn sàng chia sẻ và cùng phát triển giáo trình, tài liệu để cung cấp chất lượng đào tạo tốt hơn cho sinh viên Việt Nam.



FPT POLYTECHNIC

XIN VUI LÒNG LIÊN HỆ

Văn phòng FPT Polytechnic Việt Nam

Địa chỉ: Tòa nhà FPT Polytechnic, Đường Hàm Nghi,
KĐT Mỹ Đình I, Từ Liêm, Hà Nội

Điện thoại: (04) 7 305 9886 - (04) 7 3080898

Email: caodang@fpt.edu.vn

Cơ sở tại Hà Nội

Địa chỉ: Tòa nhà FPT Polytechnic, Đường Hàm Nghi,
KĐT Mỹ Đình I, Từ Liêm, Hà Nội

Điện thoại: (04) 8 582 0808 - (04) 6 287 1911

Email: caodangfpt.hn@fpt.edu.vn

Cơ sở tại Đà Nẵng

Địa chỉ: 137 Nguyễn Thị Thập, Phường Hòa Minh,
Quận Liên Chiểu, TP. Đà Nẵng

Điện thoại: (0511) 3 710 999

Email: caodangfpt.dn@fpt.edu.vn

Cơ sở tại Tây Nguyên

Tầng 2 tòa nhà VIB 27 Nguyễn Tất Thành, Phường
Tân Lợi,

TP. Buôn Ma Thuột, Tỉnh Đăk Lăk

Điện thoại: (0500) 355 5678

Email: caodangfpt.daklak@fpt.edu.vn

Cơ sở tại TP. Hồ Chí Minh

Địa chỉ: 391A Nam Kỳ Khởi Nghĩa, Q. 3, TP. HCM

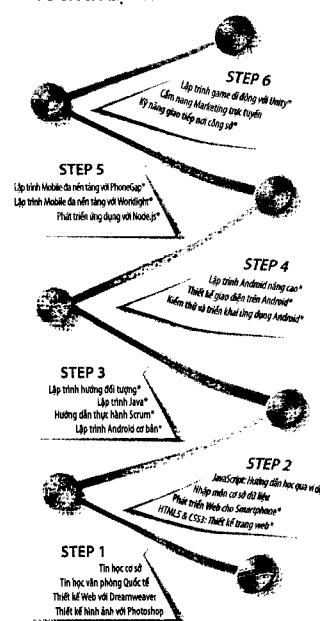
Điện thoại: (08) 3526 8799

Email: caodangfpt.hcm@fpt.edu.vn

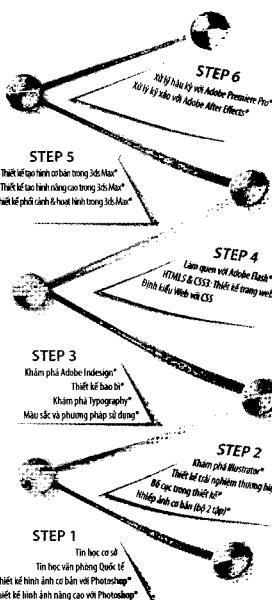
Mời các bạn đón đọc các Tủ sách chuyên nghề Trường Đại Học FPT.
 Lịch phát hành dự kiến chi tiết được cập nhật tại trang web: books.fpt.edu.vn



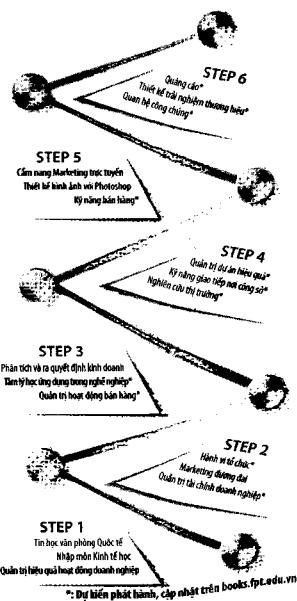
TỦ SÁCH LẬP TRÌNH VIÊN MOBILE



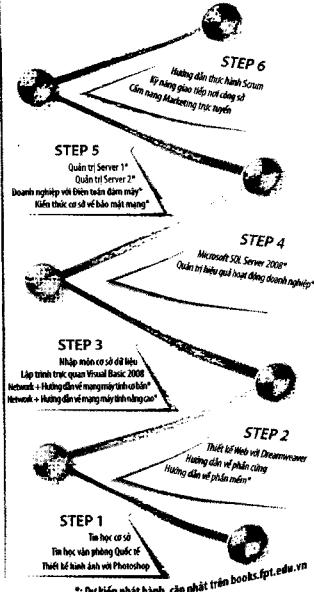
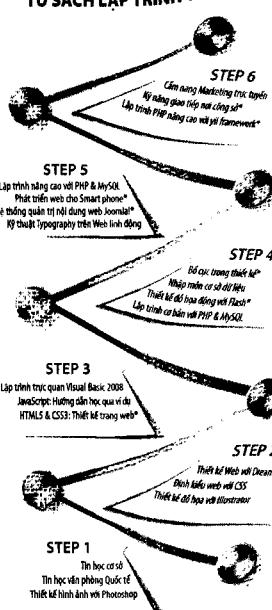
TỦ SÁCH DESIGNER



TỦ SÁCH MARKETER



TỦ SÁCH KỸ THUẬT VIÊN UD CNTT





NHÀ XUẤT BẢN BÁCH KHOA HÀ NỘI

Ngõ 17, Tạ Quang Bửu – Hai Bà Trưng – Hà Nội

ĐT: 04. 38684569; Fax: 04. 38684570

Website: <http://nxbbk.hust.edu.vn>

Andrew Pham, Phuong-Van Pham

HƯỚNG DẪN THỰC HÀNH SCRUM

QUẢN TRỊ DỰ ÁN PHẦN MỀM THEO TRIẾT LÝ AGILE

Chịu trách nhiệm xuất bản	GĐ - TBT- TS. Phùng Lan Hương
Dịch	Nguyễn Việt Khoa
Biên tập	Khuất Thị Thùy Linh, Nguy Thị Liễu
Sửa bản in	Chu Đình Phú
Vẽ bìa	Nguyễn Thế Hoàng
Trình bày	Nguyễn Thế Hoàng Nguyễn Thị Bích Mai

In 2000 bản, khổ 16 x 24 cm tại Công ty cổ phần In Hà Nội.

Địa chỉ: Lô 6B - CN5 Cụm Công nghiệp Ngọc Hồi - Thanh Trì - Hà Nội

Số đăng ký KHXB: 247-2015/CXB/18-09/BKHN

Quyết định xuất bản số: 10 /QĐ - ĐHBK - BKHN cấp ngày 05 tháng 02 năm 2015

In xong và nộp lưu chiểu Quý I năm 2015

Văn phòng FPT Polytechnic Việt Nam

Địa chỉ: Tòa nhà FPT Polytechnic,

Đường Hàm Nghi, KĐT Mỹ Đình I, Từ Liêm, Hà Nội

Điện thoại: (04) 7 305 1398

Email: caodang@fpt.edu.vn

Hà Nội

Địa chỉ: Tòa nhà FPT Polytechnic,
Đường Hàm Nghi, KĐT Mỹ Đình I, Từ Liêm, Hà Nội

Điện thoại: (04) 8582 0808 – (04) 6287 1911

Email: caodangfpt.hn@fpt.edu.vn

Tây Nguyên

Địa chỉ: Tầng 2 tòa nhà VIB 27 Nguyễn Tất Thành,

Phường Tân Lợi, TP. Buôn Ma Thuột, Tỉnh Đăk Lăk

Điện thoại: (0500) 355 5678

Email: caodangfpt.daklak@fpt.edu.vn

Đà Nẵng

Địa chỉ: 137 Nguyễn Thị Thập, Phường Hòa Minh,

Quận Liên Chiểu, TP Đà Nẵng

Điện thoại: 0511. 3710.999

Email: caodangfpt.dn@fpt.edu.vn

TP. Hồ Chí Minh

Địa chỉ: 391A Nam Kỳ Khởi Nghĩa, Quận 3,

Thành phố Hồ Chí Minh

Điện thoại: (08) 3526 8799

Email: caodangfpt.hcm@fpt.edu.vn

Nếu bạn là nhà quản lý dự án, nhà phát triển phần mềm, nhân viên kiểm thử, quản lý sản phẩm, chuyên gia phân tích nghiệp vụ, hay bất kỳ ai trong lĩnh vực phát triển phần mềm, bạn sẽ thấy cuốn sách rất hữu ích, giúp bạn hiểu mọi thứ về nhóm Scrum với tính thực tiễn cao. Đây là một tác phẩm dễ hiểu và tôi tin rằng bạn sẽ hứng thú đọc, bất kể bạn thuộc vị trí nào.

—Trích từ Lời nói đầu của Sanjiv Augustine,
tác giả cuốn sách *Managing Agile Projects*

Với tất cả những khía cạnh mà chúng ta đã biết, việc triển khai các quy trình Agile, và đặc biệt là Scrum trong một tổ chức là một chiến công không nhỏ. Cuốn sách này là một nguồn tài nguyên có giá trị để giúp thực hiện điều đó.

—Trích từ Lời tựa của Dan Pilone,

Tác giả cuốn *Head First Software Development* và *Head First iPhone Development*

Hướng dẫn thực hành Scrum

Quản trị dự án phần mềm theo triết lý Agile

Scrum in Action: Agile Software Project Management and Development

Cuốn **Hướng dẫn thực hành Scrum – Quản trị dự án phần mềm theo triết lý Agile** là tài liệu hướng dẫn thực tế, cung cấp cho các nhóm dự án phần mềm cách thức để triển khai thành công khung làm việc phát triển phần mềm Agile thông qua việc sử dụng Scrum. Với lối viết rõ ràng, súc tích, đây là cuốn sách hướng dẫn đầu tiên viết về những tinh huống thực tế mà các nhà thực hành Scrum gặp phải trong các công ty. Cuốn sách mô tả phương pháp để nhóm dự án đạt hiệu quả cao nhất, đồng thời xóa nhòa khoảng cách giữa nhiều sách viết về quản lý dự án và về Scrum, thông qua việc chỉ ra phương thức giao tiếp với những nhà điều hành bằng cách sử dụng thuật ngữ tài chính, cách sử dụng kỹ thuật ước tính khách quan, và cách để kiến trúc dự án Scrum phù hợp với kiến trúc phần mềm tổng thể của doanh nghiệp. Bên cạnh đó, sách còn bao hàm một phần phụ lục, cung cấp case study về hai sản phẩm phần mềm đã được xây dựng và triển khai thành công bằng cách sử dụng những kỹ thuật và lời khuyên đã trình bày trong cuốn sách này.

Nội dung cuốn sách:

- Kiến thức cơ bản về tài chính giúp bạn cộng tác tốt hơn với các cấp quản lý kinh doanh
- Phương thức để giành được sự ủng hộ của nhà quản lý
- Kỹ thuật trực quan để thu thập các yêu cầu của dự án Scrum mà bất kỳ ai cũng có thể sử dụng
- Cách sử dụng tầm nhìn kiến trúc để giảm sự biến động về tốc độ nhóm
- Cách sử dụng kỹ thuật ước tính điểm story khách quan để triển khai Scrum trên toàn doanh nghiệp
- Tầm quan trọng của kiểm thử tự động, kiểm thử hồi quy và kiểm thử tích hợp
- Vai trò lãnh đạo nhóm trong môi trường Scrum
- Cách để thích ứng Scrum mà không phá hủy tính linh hoạt cơ bản và triển khai ScrumBut tiêu cực
- Cách để tự đánh giá mức độ sẵn sàng của dự án



FPT Polytechnic Việt Nam

Địa chỉ: Tòa nhà FPT Polytechnic,
Đường Hàm Nghi, KĐT Mỹ Đình I, Từ Liêm, Hà Nội
Điện thoại: (04) 7 305 9886 - (04) 7 3080898
Email: caodang@fpt.edu.vn

Hà Nội

Địa chỉ: Tòa nhà FPT Polytechnic,
Đường Hàm Nghi, KĐT Mỹ Đình I, Từ Liêm, Hà Nội
Điện thoại: (04) 8582 0808 - (04) 6287 1911
Email: caodangfpt.hn@fpt.edu.vn

Đà Nẵng

Địa chỉ: 137 Nguyễn Thị Thập, Phường Hòa Minh,
Quận Liên Chiểu, TP Đà Nẵng
Điện thoại: 0511. 3710.999
Email: caodangfpt.dn@fpt.edu.vn

Tây Nguyên

Địa chỉ: Tầng 2 tòa nhà VIB 27 Nguyễn Tất Thành,
Phường Tân Lợi, TP Buôn Ma Thuột, Tỉnh Đắk Lăk
Điện thoại: (0500) 355 5678
Email: caodangfpt.daklak@fpt.edu.vn

TP. Hồ Chí Minh

Địa chỉ: 391A Nam Kỳ Khởi Nghĩa, Quận 3,
Thành phố Hồ Chí Minh
Điện thoại: (08) 3526 8799
Email: caodangfpt.hcm@fpt.edu.vn

ISBN: 978-604-938-495-0



Giá: 124.000 đồng