

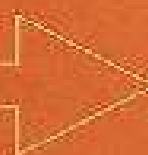


# CẨM NANG SCRUM

— cho người mới bắt đầu —



Tập thể tác giả thuộc Agile



# Mục lục

1. [Lời nói đầu](#)
2. [Lời cảm ơn](#)
3. [Cùng bạn dùng sách](#)
4. [1. Tổng quan Agile](#)
5. [2. Khái lược Scrum](#)
6. [3. Nhóm Scrum](#)
7. [4. Các sự kiện Scrum](#)
8. [5. Các tạo tác và công cụ](#)
9. [6. Scrum trong tổ chức](#)
10. [7. Kỹ thuật Agile](#)
11. [8. Scrum phân tán](#)
12. [9. Scrum với quy mô lớn](#)
13. [Tài liệu tham khảo và đọc thêm](#)
14. [Phụ lục 1: Thuật ngữ](#)
15. [Phụ lục 2: Danh mục kiểm tra](#)
16. [Bảng chỉ mục](#)

# Lời nói đầu

Bạn đang cầm trên tay cuốn sách về Scrum đầu tiên do người Việt biên soạn. Sau nhiều năm hoà nhập cùng cộng đồng Agile thế giới trong việc học hỏi, ứng dụng, và xây dựng cộng đồng Agile tại Việt Nam, cuốn sách này đánh dấu một bước tiến mới trong việc phát triển một hệ sinh thái Agile mạnh tại Việt Nam.

Kể từ hội nghị tại Utah, Hoa Kỳ, năm 2001, khai sinh ra đường lối Phát triển Phần mềm Linh hoạt (Agile Software Development) mà chúng ta gọi tắt là Agile, chúng ta đã chứng kiến những thay đổi lớn lao của thế giới phần mềm nói riêng, và thế giới công nghệ nói chung. Đó là thời gian đủ để những Google, iPhone, iPad, Android, Facebook, Twitter... ra đời và đảo lộn trật tự thế giới.

Agile, từ chỗ là một trào lưu phản kháng trong đợt khủng hoảng các phương pháp phát triển phần mềm, đã trở thành “tiêu chuẩn hờ” cho việc phát triển phần mềm, lan sang phát triển sản phẩm công nghệ nói chung, và tiện thể loang cả sang những địa hạt mà những người kí vào bản Tuyên ngôn Agile năm 2001 không thể ngờ tới: Marketing, giáo dục, quản trị, trong gia đình, nông nghiệp hay thậm chí cả trong... nhà thờ.

Nhóm tác giả Học viện Agile hân hạnh đồng hành cùng các bạn học Agile, thực hành Agile để kiến tạo những giá trị mới, tạo những đổi thay tích cực. Thông qua việc biên soạn tài liệu này, chúng tôi mong muốn mang đến cho bạn cơ hội học hỏi, thực hành Agile với chất lượng cao và hiệu quả tốt.

Dù rất nỗ lực, cuốn sách chắc chắn không khỏi mắc phạm phải những thiếu sót. Chúng tôi luôn mong chờ những phản hồi của bạn để cải tiến liên tục tài liệu này tại thư điện tử [contact@hocvienagile.com](mailto:contact@hocvienagile.com).

**Học viện Agile**

# Lời cảm ơn

Cuốn sách này được viết dựa trên một tài liệu giảng dạy Scrum do chúng tôi khởi thảo từ cuối 2011, và liên tục cải tiến để giúp cho việc học Scrum ngày càng dễ dàng hơn. Phần nhiều những cải tiến đó đến từ người học, mà chúng tôi gọi là những “đối tác phát triển” – những người giúp nhau cùng tiến bộ. Chúng tôi vô cùng biết ơn những “đồng tác giả” vô danh này.

Trong khi tổ chức lại thành một cuốn cẩm nang có chất lượng cao như phiên bản bạn cầm trên tay, rất nhiều người đã góp ý trực tiếp về nội dung, bố cục cũng như cách thức trình bày hiệu quả. Chúng tôi xin chân thành cảm ơn bạn bè khắp nơi đã cung cấp những lời khuyên quý báu. Trong đó xin đặc biệt cảm ơn chị Nguyễn Phương Anh – Giám đốc Khối đào tạo liên kết quốc tế thuộc Đại học FPT, anh Nguyễn Tuấn – Giám đốc đào tạo FPT Aptech Hà Nội, anh Phạm Mạnh Lân – Co-founder NAL Group, anh Hán Văn Thắng – CEO DEHA, anh Nguyễn Ngọc Tú và Nguyễn Minh Tâm từ NAL Vietnam, anh Nguyễn Văn Hiền từ Agile Vietnam, anh Tô Hải Sơn – Giám đốc vận hành NTQ Solution, và những người bạn khác nữa chưa kịp kể tên.

Xin cảm ơn anh Đoàn Xuân Trường đã thiết kế bìa; xin cảm ơn chị Nguyễn Thị Phương – đồng nghiệp đã lao tâm khổ tứ để cuốn sách này đến được với bạn đọc. Cảm ơn Kim Cúc đã hiện thực hoá những ý tưởng trình bày khác thường của nhóm tác giả.

Cuối cùng, chúng tôi xin vô cùng cảm ơn bạn – người đang đọc cuốn sách này, vì chính bạn là người cùng chúng tôi tạo ra giá trị cho cuốn sách và lan toả những giá trị Scrum.

# Cùng bạn dùng sách

Cuốn sách này được viết ra để cho những nhà phát triển và quản lí bước đầu sử dụng Scrum. Vì thế nó được thiết kế tối ưu cho việc học và tra cứu. Chúng tôi nghĩ rằng 80% giá trị của cuốn sách nằm ở cách bạn đọc vận dụng nó. Do đó, chúng tôi mong muốn thảo luận đôi chút về cấu trúc của cuốn sách cũng như cách đọc, cách dùng cuốn sách này.

Chương đầu tiên rất lí thuyết nhưng là nền tảng. Có thể là chương khó đọc nhất, nhưng lại quan trọng nhất vì nó giải thích cặn kẽ triết lí đứng đằng sau Scrum. Bạn không thể thực hành tốt Scrum mà không nắm rõ những nguyên lí Agile có tính dẫn đường. Do vậy hãy đọc qua Chương 1 trước khi đọc các chương tiếp theo, và thỉnh thoảng đọc lại chương này để suy ngẫm thêm. Nếu bạn nóng ruột muốn tìm hiểu Scrum thì có thể chỉ cần lướt qua Chương 1, nhưng hãy quay trở lại sau khi đã đọc xong cả cuốn sách.

Chương 2, 3, 4, 5 mô tả chi tiết khung Scrum và cách vận hành trong thực tiễn. Cuốn sách này hướng đến việc thực hành “đúng” Scrum. Vì thế hãy đọc kĩ, và tự trả lời các câu hỏi cuối chương. Việc trả lời các câu hỏi này giúp bạn đưa kiến thức vào thực tiễn, bối cảnh đặc thù của bạn. Ngoài ra, việc trả lời các câu hỏi này giúp cho bạn bổ khuyết được một khía cạnh không được đề cập trong khuôn khổ một cuốn cẩm nang ngắn gọn: vận dụng Scrum “hợp lí”.

Chương 6 thảo luận về các tình huống sử dụng Scrum và những điều ghi nhớ khi lần đầu tiên đưa Scrum vào tổ chức. Bạn có thể nhảy cóc đến chương này nếu tò mò về các bước cụ thể áp dụng Scrum.

Các chương 7, 8, 9 có thể coi là phần mở rộng của cuốn sách nhằm cung cấp một bức tranh lớn hơn khi vận dụng Scrum. Người mới bắt đầu với Scrum, đối tượng chính của cuốn sách này, có vẻ ít khi vận dụng những nội dung ở các chương này, nhưng bạn cần nắm

được một bức tranh lớn để hình dung ra một lộ trình dài hơi cho việc biến Scrum thành một đường lối làm việc nhất quán, cải tiến liên tục, hiệu quả và thành công bền vững.

Riêng Chương 7 có đề cập nhiều kĩ thuật đặc thù Agile mà nhiều lập trình viên thấy lạ lẫm và rất thích thú. Bạn hãy lướt qua, tìm hiểu kĩ thêm (tham khảo danh mục đọc thêm ở cuối sách) nếu muốn đi xa hơn.

Nếu bạn để ý, đây là cuốn sách kĩ thuật bằng tiếng Việt hiếm hoi có Bảng chỉ mục. Hãy tận dụng nó cho việc học và tra cứu. Hãy dùng Bảng chỉ mục (Index) ở cuối sách, Mục lục chi tiết, và phần Phụ lục 1 (Thuật ngữ Scrum) để tra cứu, chuyển qua chuyển lại các phần của cuốn sách, hoặc để tra cứu thêm thông tin trên Internet về một khái niệm được đề cập trong sách.

Trong sách, thỉnh thoảng có những ô “Dừng và nghĩ” với một số câu hỏi hướng dẫn, như những nốt lặng trong các bản nhạc. Bạn hãy dừng lại một chút, động não trong phút chốc để tiêu hoá nội dung của cuốn sách. Ngoài ra, các Danh mục Kiểm tra được đặt vào một số phần quan trọng cũng có thể giúp bạn trong việc tự đánh giá được mức độ đúng đắn và thành thực trong việc vận dụng Scrum trong nhóm của mình.

Một số tài nguyên và đọc thêm cho cuốn sách được chúng tôi cập nhật tại [www.hocvienagile.com/cam-nang-scrum](http://www.hocvienagile.com/cam-nang-scrum).

Chúc bạn dùng sách thật hiệu quả!



# 1 TỔNG QUAN AGILE

Trong chương này...

- Những vấn đề nổi cộm trong phát triển sản phẩm
- Sự ra đời của Agile
- Tuyên ngôn Phát triển Phần mềm Linh hoạt
- Cách tiếp cận lặp tăng trưởng và truyền thống
- Khi nào Agile, khi nào không?

*Các phương pháp Agile tối ưu các tri thức ẩn trong đội ngũ phát triển, thay vì lệ thuộc vào những thông tin được viết ra dưới dạng kế hoạch hay tài liệu.*

**Barry Boehm**

## **NHỮNG VẤN ĐỀ PHỔ BIẾN TRONG PHÁT TRIỂN SẢN PHẨM VÀ QUẢN TRỊ DỰ ÁN PHẦN MỀM**

---

Ngành phát triển phần mềm đã có hơn nửa thế kỷ hình thành và phát triển, tuy nhiên việc phát triển phần mềm và quản trị các dự án phần mềm chưa bao giờ hết thử thách. Rất nhiều những vấn đề cứ không ngừng lặp đi lặp lại, làm đau đầu đội ngũ phát triển và các nhà quản trị. Dưới đây là một số vấn đề tiêu biểu.

Theo cách thức phát triển truyền thống, dự án được lập kế hoạch cẩn thận, được tiến hành với rất nhiều khâu trung gian, và khách hàng ngồi chờ. Các bên liên quan hầu như chỉ nhận được báo cáo, tài liệu chứ không nhận được phần mềm để dùng. Thông tin về phần mềm rất mù mờ. Lâu tới ngày phát hành và không minh bạch về tiến độ khiến khách hàng và các bên liên quan sốt ruột, lo lắng và



không làm cách nào để cung cấp các phản hồi có ý nghĩa, khó hợp tác với đội phát triển cho ra sản phẩm tốt nhất.

Kế hoạch đã định, nhưng nhiều rủi ro xuất hiện: lỗi xuất hiện ngày càng nhiều, hiểu sai yêu cầu, những khó khăn về mặt công nghệ, một vài thành viên có vấn đề và không làm việc như kỳ vọng và sản phẩm chậm ngày phát hành.

Mọi thứ đã được tích hợp, nhưng sản phẩm thiếu ổn định do không kiểm soát được chất lượng. Ở rất nhiều dự án, mọi công việc đã hoàn thành nhưng giai đoạn tích hợp và ổn định hệ thống thật là thảm họa, không biết khi nào kết thúc.

Kế hoạch thường xuyên sai sót nên chúng ta phải đầu tư nhiều thời gian hơn vào xây dựng kế hoạch. Nhưng dù cố gắng đến mấy thì kế hoạch vẫn không đâu vào đâu và chúng ta lại bị phàn nàn là làm kế hoạch chưa kỹ. Lập kế hoạch thật là một ác mộng, nhiệm vụ không thể hoàn thành.

Khi đã thực hiện xong phân tích yêu cầu, thiết kế và bắt đầu lập trình, khách hàng đổi yêu cầu. Vì đó là yêu cầu rất quan trọng, bạn không thể từ chối. Nhóm rất khó xử. Họ không biết làm thế nào bởi việc thay đổi rất khó.

Ban đầu sản phẩm còn chạy tốt, nhưng lượng mã nguồn ngày một tăng lên, áp lực thời gian, v.v. nên chất lượng sản phẩm cứ giảm sút. Kế hoạch đã bị vỡ, chất lượng ngày một giảm, tất cả mọi người phải làm thêm giờ với áp lực rất cao, thành viên không hạnh phúc.

**⇒ Dừng và Nghĩ**

**Bạn và nhóm của mình đang gặp phải những vấn đề nổi cộm nào?**

**KHỦNG HOẢNG PHƯƠNG PHÁP LUẬN, VÀ SỰ RA ĐỜI CỦA AGILE**

Thập kỉ 80 của thế kỉ XX chứng kiến một thời kì khủng hoảng phương pháp luận phát triển phần mềm, do tỉ lệ thất bại của các dự án phần mềm quá cao. Hàng loạt nỗ lực của các nhà thực hành cũng như giới hàn lâm đã cố gắng tìm ra phương pháp hữu hiệu để đảm bảo tính hiệu quả trong phát triển phần mềm.

Thập kỉ 90, nhiều phương pháp phát triển phần mềm với quy trình nhẹ (light weight) và linh hoạt ra đời, như XP, Scrum, FDD, Crystal, ... và nhanh chóng được lan rộng.



Từ 11 - 13 tháng 2 năm 2001, 17 nhà phát minh và nhà thực hành đã họp nhau tại bang Utah, Hoa Kỳ, để thảo luận về hướng đi mới trong phương pháp luận phát triển phần mềm. Họ đã đi đến thống nhất và cho ra đời bản Tuyên ngôn Agile (The Manifesto for Agile Software Development) và đánh dấu một xu thế mới trong phát triển phần mềm. Ngày nay chúng ta gọi Agile, tức là chỉ chung một họ phương pháp phát triển phần mềm có chung sự chia sẻ các giá trị và nguyên tắc được phát biểu trong Tuyên ngôn Agile. Biện pháp thực hành có thể rất khác nhau, nhưng triết lí chung thì giống nhau.



*Sự ra đời Tuyên ngôn Agile. Ảnh: [agilemanifesto.org](http://agilemanifesto.org)*

Sau một thập kỉ rưỡi ra đời. Agile đã cải thiện đáng kể khả năng thành công của các dự án.

Báo cáo CHAOS của Standish Group năm 2015 cho thấy, các dự án Agile có tỉ lệ thành công cao hơn 3 lần so với dự án truyền thống.



**Dự án thành công:** đúng thời gian, đúng ngân sách với mọi tính năng và kết quả thoả mãn.

**Dự án thử thách:** hoàn thành nhưng không đạt một trong các tiêu chí đúng ngân sách, đúng thời gian, hoặc kết quả không thỏa mãn

**Dự án thất bại:** dự án bị cắt ở một thời điểm nào đó trong quá trình phát triển



Những nghiên cứu chi tiết hơn định lượng được tác động của những phương pháp Agile lên năng suất lao động.

Như trong nghiên cứu của Sutherland và đồng nghiệp năm 2008, năng suất của nhóm Scrum cao hơn tới gần 9 lần so với nhóm sử dụng phương pháp truyền thống.

Trong nghiên cứu này, kích thước phần mềm được đo bằng hai thước đo phổ biến trong phát triển phần mềm là số dòng mã và điểm chức năng (function point).

Năng suất tính theo điểm chức năng của nhóm Scrum cao hơn nhóm truyền thống  $15,3/1,7 = 8,88$  lần.



\* So sánh với phương pháp truyền thống waterfall

\*\* KLOC: 1000 dòng mã

⇒ Theo Sutherland, J., Schoonheim, G., Rustenburg, E., & Rijk, M. (2008, August). Fully distributed scrum: The secret sauce for hyperproductive offshored development

## **TUYÊN NGÔN PHÁT TRIỂN PHẦN MỀM LINH HOẠT**

Chúng tôi đã tìm ra cách tốt hơn để phát triển phần mềm thông qua việc thực hành và giúp đỡ người khác thực hiện. Qua đó, chúng tôi đánh giá cao:



*Mặc dù các thứ bên phải vẫn còn giá trị, chúng tôi đánh giá cao hơn các mục ở bên trái.*

## **12 NGUYÊN LÝ PHÍA SAU TUYÊN NGÔN NGÔN AGILE**

---

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 việc thay đổi yêu cầu, thậm chí rất muộn trong quá trình phát triển. Các quy trình linh hoạt 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 cho các 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. Xây dựng các dự án xung quanh những cá nhân có động lực. Cung cấp cho họ môi trường và 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 linh hoạt thúc đẩy 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 các kỹ thuật và thiết kế tốt để gia tăng sự linh hoạt.
10. Sự đơn giản – nghệ thuật tối đa hóa lượng công việc chưa xong – là căn bản.

11. Các kiến trúc tốt nhất, yêu cầu tốt nhất, và thiết kế tốt nhất sẽ được làm ra bởi các nhóm tự tổ chức.

12. Nhóm phát triển sẽ thường xuyên suy nghĩ về việc làm sao để trở nên hiệu quả hơn, sau đó họ sẽ điều chỉnh và thay đổi các hành vi của mình cho phù hợp.

**Theo AgileManifesto.org**

## **Ý NGHĨA CỦA TUYÊN NGÔN AGILE**

*theo Jeff Sutherland\**

Những giá trị đó không chỉ là thứ mà các tác giả của Tuyên ngôn Agile dự định cung cấp ra để phục vụ cho tuyên ngôn và rồi sau đó đi vào quên lãng. Chúng là các giá trị căn cứ vào để làm việc. Bản thân mỗi phương pháp linh hoạt đều tiếp cận các giá trị trên theo các cách thức khác nhau, nhưng tất cả các phương pháp này đều có tiến trình và biện pháp thực hành cụ thể để nuôi dưỡng một hoặc nhiều trong số các giá trị đó.



### **Cá nhân và tương tác**

Cá nhân và sự tương tác giữa họ là cốt yếu để nhóm đạt được hiệu suất cao. Các nghiên cứu về sự “bão hòa giao tiếp” (communication saturation) trong dự án cho thấy rằng, khi không có vấn đề trong giao tiếp, nhóm có thể thực hiện công việc tốt hơn 50 lần so với mức trung bình trong lĩnh vực của mình. Để tạo điều kiện cho giao tiếp, các phương pháp linh hoạt thường xuyên sử dụng chu trình thanh- tra-và-thích-nghi. Chu trình này có thể diễn ra hằng phút với lập trình cặp (pair-programming), hằng giờ với tích hợp liên tục (continuous integration), hằng ngày với standup meeting (họp đứng), hằng phân đoạn với các buổi họp sơ kết và cải tiến.

Tuy nhiên, chỉ tăng tần suất phản hồi và giao tiếp là không đủ để loại bỏ các vấn đề về giao tiếp. Chu kỳ thanh-tra-và-thích-nghi hoạt

động tốt chỉ khi các thành viên trong nhóm thể hiện những hành vi quan trọng sau:

- Tôn trọng giá trị của mỗi cá nhân
- Trung thực trong giao tiếp
- Minh bạch về dữ liệu, hoạt động, và quyết định
- Tin tưởng vào sự hỗ trợ của mỗi cá nhân với nhóm
- Cam kết với nhóm và các mục tiêu của nhóm



Để thúc đẩy các hành vi này, nhà quản lí linh hoạt phải cung cấp một môi trường hỗ trợ, các nhà huấn luyện phải tạo điều kiện thuận lợi, và các thành viên của nhóm phải thể hiện chúng. Chỉ khi đó nhóm mới có thể phát huy được hết tiềm năng của mình.

Đạt tới các hành vi đó khó khăn hơn rất nhiều so với việc hình thành chúng. Hầu hết các nhóm tránh né sự thật, sự minh bạch, và tin tưởng vào các chuẩn mực văn hóa hoặc có những kinh nghiệm tiêu cực từ các xung đột xuất phát từ truyền thống trước đây. Để chống lại những khuynh hướng này, ban lãnh đạo và thành viên nhóm phải tạo điều kiện cho những xung đột tích cực. Làm như vậy không chỉ giúp tạo ra hành vi sinh lợi mà còn đem lại các lợi ích khác:



- Cải tiến quy trình phụ thuộc vào nhóm trong việc tạo ra danh sách các trở ngại hoặc vấn đề trong tổ chức, đối mặt với chúng một cách trung thực, và sau đó loại bỏ chúng một cách có hệ thống tùy theo độ ưu tiên.
- Đổi mới chỉ xảy ra với việc tự do trao đổi các ý tưởng mâu thuẫn nhau, như Takeuchi và Nonaka đã từng nghiên cứu và chỉ ra.

- Việc hướng các nhóm tới mục tiêu chung đòi hỏi nhóm phải đổi mới và giải quyết các vấn đề về xung đột.
- Cam kết làm việc cùng nhau sẽ xảy ra chỉ khi mọi người đồng ý với mục tiêu chung và sau đó đấu tranh cho sự tiến bộ của bản thân cũng như của nhóm.

Ý cuối cùng ở trên nói về cam kết là đặc biệt quan trọng. Đó là khi mà các cá nhân và các nhóm được cam kết và cảm thấy có trách nhiệm với việc cung cấp các giá trị cao, đó là điểm mấu chốt đối với các nhóm phát triển phần mềm. Các phương pháp linh hoạt tạo điều kiện cho việc cam kết bằng cách khuyến khích nhóm đưa ra một danh sách công việc được ưu tiên hóa, để họ tự quản lý công việc của mình, và tập trung vào cải tiến về cách thực hiện các công việc đó.

## **Phần mềm chạy tốt**

Phần mềm chạy tốt là một trong những khác biệt lớn mà phát triển linh hoạt mang lại. Tất cả các phương pháp linh hoạt thể hiện Tuyên ngôn Agile bằng cách nhấn mạnh việc cung cấp một phần nhỏ của phần mềm chạy tốt tới khách hàng sau một khoảng thời gian nhất định.

Tất cả các nhóm Agile phải xác lập những gì họ muốn nói là “phần mềm chạy tốt”, cái thường được biết như là Định nghĩa Hoàn thành. Ở mức độ cao, một phần của chức năng hoàn thành chỉ khi các tính năng của chúng vượt qua tất cả các kiểm thử và có thể được vận hành bởi người dùng cuối. Ở mức thấp nhất, các nhóm phải vượt qua được kiểm thử đơn vị (unit test) và kiểm thử hệ thống. Các nhóm tốt nhất còn bao gồm việc kiểm thử tích hợp, kiểm thử hiệu năng, và kiểm thử chấp nhận của khách hàng trong định nghĩa hoàn thành đối với một phần chức năng. Thông qua nguồn dữ liệu phong phú từ các dự án, một công ty CMMI cấp độ 5 cho thấy việc xác định kiểm thử chấp nhận cùng với các tính năng, triển khai một loạt các tính năng và theo độ ưu tiên, ngay lập tức chạy các kiểm thử chấp nhận với mỗi tính năng, và sửa bất cứ một lỗi nào có độ ưu tiên cao nhất sẽ tăng gấp đôi tốc độ sản xuất và giảm các sai sót

đến 40%. Điều này có được từ một công ty có tỉ lệ sai sót thấp nhất thế giới.

Tuyên ngôn Agile khuyến nghị các nhóm cung cấp phần mềm chạy tốt sau một khoảng thời gian nhất định. Đồng thuận với Định nghĩa Hoàn thành là một trong những cách thực tế để nhóm Agile mang lại hiệu suất và chất lượng cao, cái cần thiết để hoàn thành mục tiêu này.



## **Cộng tác với khách hàng**

Trong hai thập kỷ qua, tỉ lệ thành công của các dự án tăng hơn hai lần trên toàn thế giới. Điều này được cho là vì các dự án nhỏ hơn và mức độ chuyển giao thường xuyên đã cho phép khách hàng cung cấp các thông tin phản hồi về phần mềm hoạt động một cách đều đặn hơn. Các tác giả của bản Tuyên ngôn Agile đã làm sáng tỏ điều này khi họ nhấn mạnh rằng việc khách hàng tham gia vào quá trình phát triển phần mềm là hết sức cần thiết để dẫn tới thành công.

Các phương pháp phát triển linh hoạt đã thúc đẩy giá trị này bằng cách đưa vào một đồng minh tích cực của khách hàng làm việc sát cánh với đội phát triển. Ví dụ, một nhóm Scrum đầu tiên có hàng ngàn khách hàng. Sẽ là không khả thi nếu cho phép tất cả khách hàng tham gia vào quá trình phát triển sản phẩm, vì vậy họ chọn ra một vị đại sứ của khách hàng, được gọi là Product Owner (chủ sản phẩm), để đại diện cho không chỉ tất cả khách hàng trong trường hợp này mà còn bao gồm cả quản lí, dịch vụ khách hàng, và các bên liên quan khác. Product Owner có trách nhiệm cập nhật danh sách yêu cầu về sản phẩm sau mỗi bốn tuần thời điểm mà nhóm Scrum phát hành phiên bản sản phẩm chạy tốt, có tính đến yếu tố thực tế cùng phản hồi của khách hàng và các bên liên quan. Điều này cho phép khách hàng có thể định hình sản phẩm phần mềm đang được tạo ra.

Một nhóm XP đầu tiên đã bắt đầu với một dự án CNTT nội bộ. Họ có thể có sẵn người sử dụng đầu cuối của công ty trong nhóm làm



việc với họ hằng ngày. Sử dụng khoảng 10% thời gian, các tư vấn viên và nhóm nội bộ có thể tìm được một người dùng cuối có thể làm việc với nhóm từng ngày. 90% thời gian còn lại, họ phải cử ra người đại diện cho khách hàng. Người này, được nhóm XP gọi là Customer (khách hàng), làm việc trực tiếp với người dùng cuối để cung cấp một danh sách các tính năng rõ ràng cùng độ ưu tiên cho phép đội phát triển có thể thực hiện.

Cộng tác với khách hàng (hoặc đại diện của khách hàng) trên cơ sở hằng ngày là một trong những lý do lý giải tại sao các dữ liệu trong ngành công nghiệp cho thấy rằng các dự án linh hoạt có tỉ lệ thành công cao hơn gấp đôi so với các dự án truyền thống tính trung bình trên toàn thế giới. Các phương pháp phát triển linh hoạt đã nhận ra điều đó, và do vậy, chúng đã tạo ra một vị trí đặc biệt trong đội hình phát triển dành riêng cho vị khách hàng đại diện này.

### **Phản hồi với thay đổi**

Phản hồi với thay đổi là điều cần thiết cho việc tạo ra một sản phẩm làm hài lòng khách hàng cũng như mang lại những giá trị kinh doanh. Dữ liệu ngành công nghiệp cho thấy hơn 60% các yêu cầu về sản phẩm hay dự án bị thay đổi suốt quá trình phát triển phần mềm. Ngay cả khi các dự án truyền thống kết thúc đúng thời gian, trong giới hạn kinh phí, với tất cả các tính năng theo kế hoạch, nhưng khách hàng thường không hài lòng vì những gì họ thấy không thật sự đúng như những gì họ muốn. Luật Humphrey nói rằng khách hàng không bao giờ biết những gì họ muốn cho đến khi họ thấy phần mềm hoạt động. Nếu khách hàng không nhìn thấy phần mềm hoạt động cho đến khi kết thúc dự án, sẽ là quá muộn cho việc kết hợp các thông tin phản hồi của họ ở thời điểm này. Các phương pháp phát triển linh hoạt tìm kiếm sự phản hồi của khách hàng trong suốt dự án để có thể kết hợp thông tin phản hồi và thông tin mới ngay khi sản phẩm đang được phát triển.



Tất cả các phương pháp phát triển linh hoạt đều được tích hợp sẵn những tiến trình thay đổi các kế hoạch trong một khoảng thời gian

đều dựa trên những thông tin phản hồi từ phía khách hàng cũng như bên đại diện của khách hàng. Các kế hoạch được thiết kế để sao cho luôn cung cấp giá trị kinh doanh cao nhất trước hết. Bởi vì 80% giá trị nằm trong 20% các tính năng, một dự án phát triển linh hoạt chạy tốt có xu hướng kết thúc sớm, trong khi hầu hết các dự án truyền thống thường kết thúc trễ. Kết quả là, khách hàng thì vui vẻ hơn, và các nhà phát triển thì thích thú với công việc của họ hơn. Các phương pháp phát triển linh hoạt dựa trên những hiểu biết đó, để thành công hơn chúng phải có kế hoạch để thay đổi. Đó là lý do tại sao chúng thiết lập các quy trình, chẳng hạn như Sơ kết và Cải tiến, được thiết kế đặc biệt để thay đổi các ưu tiên thường xuyên dựa trên thông tin phản hồi của khách hàng và giá trị kinh doanh.



⇒ **Dừng và Nghĩ**

**Bạn có đồng tình với các điểm chính của Tuyên ngôn Agile không? Bạn có muốn bổ sung gì thêm không?**



Agile chỉ là tập hợp những giá trị được nêu trong Tuyên ngôn Phát triển Phần mềm Linh hoạt (gọi tắt là Tuyên ngôn Agile).

Có rất nhiều phương pháp chia sẻ những giá trị của Tuyên ngôn Agile, chúng đều được gọi là phương pháp Agile Scrum là một trong số các phương pháp Agile phổ biến.

### **Các phương pháp Agile và mức độ phổ biến**

Có rất nhiều phương pháp Agile khác nhau và những biến thể từ sự kết hợp giữa các phương pháp Agile ban đầu.

Theo khảo sát của VersionOne năm 2015, Scrum là phương pháp phổ biến nhất. Scrum và các phương pháp lai với Scrum như Scrumban, Scrum và XP chiếm gần  $\frac{3}{4}$  mức độ phổ biến.



## **Đặc trưng quan trọng của Agile: Tiếp cận tăng trưởng lặp**

Nhóm Phát triển sẽ hoàn thành một tập những tính năng ở phần trên của Product Backlog trong một khoảng thời gian đóng khung (Sprint) tạo ra một phần tăng trưởng cho khách hàng.

Khách hàng sẽ nhận được phần tăng trưởng cuối mỗi Sprint.

Sản phẩm sẽ tăng trưởng dần cùng với thời gian, tuy nhiên tại bất cứ thời điểm nào họ cũng có phần mềm chạy tốt.



Yêu cầu sản phẩm được sắp xếp theo đội ưu tiên. Những hạng mục đem lại nhiều giá trị trên vốn đầu tư (ROI) sẽ ở trên.

Product Backlog chứa các yêu cầu về sản phẩm được cập nhật liên tục để thích ứng với những phản hồi với những thay đổi cần thiết.

Cách tiếp cận này, cùng với bộ nguyên lí Agile có thể giúp đội phát triển và nhà quản trị dự án vượt qua được hầu hết những vấn đề được nhắc tới trong phần đầu của chương này.

**Trước Agile, phần mềm cơ bản được phát triển theo mô hình thác nước (Waterfall), hoặc dựa theo kế hoạch (plan-driven)**



Các công việc được làm một cách tuần tự phụ thuộc rất nhiều vào những điều đã được tiên lượng trước đó.

Mô hình này sẽ phát huy tác dụng nếu như chúng ta có thể xác định được chính xác yêu cầu khách hàng ngay từ đầu, có khả năng tiên lượng được công việc tương lai và không có thay đổi nào về công nghệ, môi trường kinh doanh, con người, v.v., điều rất hiếm gặp trong thế giới công nghệ ngày nay.

## KHI NÀO AGILE, KHI NÀO KHÔNG


---

Các dự án có quy mô, đặc thù và độ phức tạp không giống nhau, do vậy chúng ta cần phải có những phương pháp phát triển phù hợp.

Để xác định được khi nào thì ra quyết định kiểu gì, làm việc như thế nào, chúng ta có thể sử dụng Mô hình Cynefin cho việc phân loại tình huống và dự án, rồi ra các quyết định quan trọng.

Theo đó có các vùng chính mô tả các đặc trưng khác nhau bao gồm Hiện nhiên, Rắc rối, Phức hợp, Hỗn độn.

Bảng sau đây liệt kê các đặc tính từng vùng và vai trò của người ra quyết định.

img360

*Mô hình Cynefin cho việc ra quyết định chiến lược*

Đối với các dự án phát triển sản phẩm mới, thông thường chúng rơi vào vùng Phức hợp. Đó là khu vực mà phương pháp truyền thống sẽ gặp khó khăn, và cách tiếp cận Agile sẽ phù hợp hơn.

18

## NHỮNG CÂU HỎI THƯỜNG GẶP

---

### Agile là gì?

Phát triển Linh hoạt (Agile Development, gọi tắt là Agile) là một thuật ngữ có nguồn gốc từ Tuyên Ngôn Phát triển Phần mềm Linh hoạt (The Manifesto for Agile Software Development). Tuyên ngôn này được soạn thảo năm 2001 bởi một nhóm các tác giả, chuyên gia của các phương pháp phát triển phần mềm Scrum, Extreme Programming (XP), DSDM (Dynamic Systems Development Method), Crystal, FDD (Feature-Driven Development), và một vài nhà lãnh đạo khác.

Tuyên ngôn Agile đã tổng kết ra một số giá trị và nguyên tắc chung, phổ quát cho tất cả các phương pháp luận về linh hoạt đang tồn tại độc lập tại thời điểm đó.

Khi dùng chữ “agile” với tư cách là một tính từ, thì nó mang nghĩa là “linh hoạt”. Nhưng khi ta dùng chữ Agile (A viết hoa), tức là chỉ triết lí và tất cả những phương pháp phát triển phần mềm, phát triển sản phẩm và quản lí dựa trên triết lí được mô tả trong Tuyên ngôn Agile.

Agile là triết lí đã làm thay đổi diện mạo nền công nghệ thế giới trong những thập kỉ đầu thế kỉ XXI. Các phương pháp Agile đã được thực tế chứng minh là công cụ mạnh của các cá nhân, tổ chức trong việc đổi mới, sáng tạo, làm nên những sản phẩm chất lượng cao, đột phá, cũng như gia tăng hiệu quả hoạt động và năng lực cạnh tranh, phát triển văn hóa tổ chức. Những nghiên cứu thực nghiệm và thống kê từ ngành công nghiệp phần mềm cho thấy Agile có thể giúp cho các doanh nghiệp gia tăng năng suất và hiệu quả lên gấp nhiều lần, nâng tỉ lệ thành công của dự án lên gấp đôi và gia tăng năng lực cạnh tranh bền vững cho các doanh nghiệp. Cùng với Tư duy Tinh gọn (Lean Thinking), Khởi nghiệp Tinh gọn (Lean Startup), và Tư duy Thiết kế (Design Thinking), Agile vẫn đang tiếp tục tạo nên những làn sóng đổi thay lớn trên thế giới.

## **Scrum có phải là Agile không?**

Scrum là một phương pháp Agile (phổ biến nhất) nhưng không phải là Agile. Agile định nghĩa các giá trị cốt lõi và nguyên tắc định hướng, còn Scrum là một phương pháp cụ thể chia sẻ các nguyên tắc đó.

## **Vậy Scrum là gì?**

Tài liệu Hướng dẫn Scrum định nghĩa “Scrum là một khung làm việc để phát triển bền vững các sản phẩm phức tạp”. Có thể hiểu đây là khung tổ chức công việc tổng quát hướng đến phát triển các sản phẩm phức tạp, chủ yếu là phần mềm. Ngày nay Scrum có thể được dùng như là nền tảng tổ chức các công việc khác nhau, từ quản trị dự án linh hoạt nói chung, đến phát triển sản phẩm, thực

hiện các chiến dịch marketing, tổ chức dạy học, sản xuất ô tô module hóa hoặc những công việc cá nhân khác.

Dựa trên lý thuyết quản lý thực nghiệm (empirical process control), Scrum sử dụng cơ chế lặp (iterative) và tăng trưởng (incremental) để tối ưu hóa tính hiệu quả và kiểm soát rủi ro. Scrum rất đơn giản, dễ học và có khả năng ứng dụng rất rộng. Để có thể dùng Scrum,

chúng ta cần hiểu rõ và vận dụng đúng các thành tố tạo nên Scrum bao gồm các giá trị cốt lõi (còn gọi là “ba chân”, hay ba trụ cột của Scrum), các vai trò, các sự kiện, các tạo tác (artifact) và những quy tắc đặc thù của Scrum.

Hiện nay, định nghĩa Scrum là gì được ghi trong tài liệu Scrum Guide (Hướng dẫn Scrum) và được cập nhật thường xuyên bởi chính các tác giả tại <http://scrumguides.org>.

Toàn bộ cuốn sách này sẽ là về Scrum. Hãy đọc tiếp các chương sau.

## **Tại sao chọn Agile?**

Trong nền kinh tế sáng tạo đầy biến động hiện nay, yêu cầu cấp thiết đối với các cá nhân hay tổ chức là phải đạt được năng lực linh hoạt (Agility) cao độ để thích ứng và dẫn đầu trong các cuộc cạnh tranh. Agile là nền tảng để đạt được Agility và duy trì lợi thế cạnh tranh bền vững.

Triết lý Agile cùng với các framework, phương pháp cũng như các công cụ đặc thù có thể giúp các cá nhân, đội nhóm và doanh nghiệp trở nên hiệu quả hơn, năng suất hơn và đổi mới-sáng tạo tốt hơn.

## **Muốn triển khai Agile thì bắt đầu từ đâu?**

Các cá nhân có thể bắt đầu với việc áp dụng những kỹ thuật cơ bản của Personal Kanban cùng với tư duy Agile để nâng cao năng suất cá nhân, làm việc hiệu quả, kiểm soát tốt stress.

Các lập trình viên có thể bắt tay với những kĩ thuật lập trình Agile cơ bản như TDD, Pair-Programming để nâng cao chất lượng phần mềm trước khi đi xa hơn với đầy đủ bộ kĩ năng để trở thành Nghệ nhân Phần mềm (Software Craftsman) đích thực.

Các nhóm có thể ứng dụng Scrum để tổ chức lại công việc, nâng cao hiệu quả giao tiếp và hiệu suất nhóm.

Các công ty có thể bắt đầu với việc tái tổ chức công việc và quy trình nghiệp vụ với Scrum, Lean để giảm thiểu lãng phí, nâng cao hiệu quả hoạt động, tối ưu hóa nguồn lực và gia tăng năng lực cạnh tranh bền vững.

### **Triển khai Agile có tốn kém không? Có đòi hỏi điều kiện gì đặc biệt về trình độ không?**

Phụ thuộc vào nguồn lực chúng ta có, việc thực hành Agile có thể đơn giản là sắp xếp lại công việc mà không cần thêm chi phí. Đối với các chương trình chuyển đổi căn bản và toàn diện, sự đầu tư có thể sẽ phải tương xứng với mục tiêu do tổ chức đề ra.

### **Có bộ công cụ hay phần mềm nào để thực hành Agile?**

Agile quy trình tư duy, trong khi phương pháp sẽ định hình cách triển khai cụ thể, và các công cụ sẽ trợ giúp một phần nào đó. Công cụ và quy trình cần được lựa chọn kĩ để phục vụ các cá nhân làm việc hiệu quả và tương tác nhóm tốt hơn, không nên là lựa chọn để “đánh dấu” rằng “tôi đã sử dụng Agile”.

Các nhóm Scrum nhỏ, làm việc tập trung có thể chỉ cần các đồ chi phí thấp để có thể vận hành tốt Scrum: bảng trắng, giấy dán, bút viết bảng.

Chúng ta cũng có thể sử dụng hàng loạt các hệ thống phần mềm, cloud hỗ trợ tương tác nhóm theo triết lí Agile như Trello, Asana, Jira, TFS. Chúng có thể miễn phí hoặc tính phí tùy nhu cầu sử dụng của mỗi nhóm và tổ chức.

Đối với các tổ chức lớn, việc đầu tư cho các công cụ để gia tăng hiệu quả tổng thể luôn là một trong những ưu tiên trong danh mục đầu tư cho công nghệ để phát triển.

## **CÂU HỎI ỨNG DỤNG**

1. Tuyên ngôn Agile giúp bạn nhận ra giá trị gì cho mình?
2. Có người diễn dịch ý thứ hai trong Tuyên ngôn Agile là “phần mềm chạy tốt thay vì tài liệu đầy đủ”, theo bạn diễn dịch như vậy có gì không đúng? Tại sao?
3. Nhiều người cho rằng Tuyên ngôn Agile có 5 điều, theo bạn điều thứ 5 của bản tuyên ngôn này là gì?
4. Căn cứ vào đâu để nói rằng một phương pháp phát triển phần mềm là Agile?
5. Bạn nhận định xem những lý do nào khiến các phương pháp phát triển phần mềm truyền thống không còn phù hợp với thực tế?
6. Phân loại dự án của bạn theo mô hình Cynefin?
7. Tại sao cách thức phát triển lặp (iterative) và tăng trưởng (incremental) lại giúp các phương pháp Agile thích ứng với những thay đổi từ khách hàng, môi trường kinh doanh, công nghệ, v.v.?
8. Bạn sẽ giải thích như thế nào với đồng nghiệp cho rằng làm Agile là không cần có kế hoạch?
9. Khi thực hiện một dự án cụ thể của bạn thì cần tạo bao nhiêu tài liệu là đủ?
10. Hãy phân tích những yếu tố công cụ, quy trình nào đang cản trở động lực làm việc của cá nhân và sự cộng tác trong nhóm/tổ chức của bạn?



# 2 KHÁI LƯỢC SCRUM

Trong chương này...

- Định nghĩa Scrum
- Lịch sử Scrum
- Tổng quan khung làm việc Scrum
- Các ứng dụng của Scrum
- Tại sao dùng Scrum?
- Sprint
- Gói tăng trưởng

*Năng suất không phải là tất cả, nhưng về lâu dài thì nó hầu như là tất cả.*

**Paul Krugman - Nobel kinh tế 2008**

## SCRUM LÀ GÌ ?

---

Đôi khi được gọi là một phương pháp Agile, Scrum là khung làm việc linh hoạt được sử dụng phổ biến nhất đến nỗi nhiều người lầm tưởng Agile chính là Scrum hay Scrum chính là Agile.

Nhưng Scrum chỉ là một trong số hơn chục phương pháp cụ thể chia sẻ các giá trị được phát biểu trong Tuyên ngôn Agile.

Theo tài liệu Hướng dẫn Scrum, Scrum là khung làm việc (framework) để phát triển sản phẩm hoặc quản lí các dự án phức tạp theo cách thức lặp (iterative) và tăng trưởng (incremental). Quá trình phát triển được thực hiện thông qua các phân đoạn nối tiếp nhau. Khung

làm việc Scrum bao gồm các giá trị cốt lõi, vai trò, sự kiện, tạo tác và các quy tắc để gắn kết tất cả thành một thể thống nhất.

Scrum là khung làm việc linh hoạt, “dễ hiểu nhưng khó tinh thông”.

Một số hiểu lầm Scrum là một bộ công cụ hay biện pháp thực hành nên cho rằng mình cứ có Sprint Backlog hay thực hiện Scrum Hằng ngày thì gọi là Scrum. Một hiểu lầm khác, Scrum là phép thần để giải quyết tất cả các vấn đề của tổ chức hay dự án. Để sử dụng Scrum cần sự thay đổi sâu sắc trong cách nghĩ, cách làm chứ không dừng lại ở việc có một số công cụ hay thực hiện một số sự kiện. Chương này sẽ tìm hiểu toàn cảnh khung làm việc Scrum. Riêng chương 6 sẽ thảo luận cách đưa Scrum vào tổ chức.



## **Scrum – một cách diễn giải khác**

Trong một bài viết năm 2011, Denning tóm tắt 10 đặc điểm của Scrum:

1. Tổ chức công việc theo các chu trình ngắn (gọi là phân đoạn)
2. Khi nhóm làm việc trong các chu trình ngắn này, cấp quản lý không can thiệp (tức nhóm được trao quyền tối đa)
3. Nhóm báo cáo trực tiếp cho khách hàng, không phải cho nhà quản lý
4. Nhóm ước tính thời gian để hoàn thành công việc
5. Nhóm quyết định khối lượng công việc để làm trong phân đoạn
6. Nhóm quyết định cách hoàn thành công việc trong phân đoạn
7. Nhóm tự đánh giá hiệu suất của mình
8. Xác định mục đích của phân đoạn trước khi bắt đầu

9. Xác định mục tiêu thông qua các câu chuyện người dùng (user stories)

10. Loại bỏ các trở ngại cho công việc một cách có hệ thống

Theo: Steve Denning (2011). Scrum is a major management discovery, Forbes.

## **Lịch sử Scrum**

Takeuchi và Nonaka là những người có ảnh hưởng rất lớn tới sự ra đời của Scrum với bài viết “The New New Product Development Game” trên Harvard Business Review năm 1986.

Jeff Sutherland lần đầu giới thiệu Scrum tại công ty Easel vào năm 1993.

Ken Schwaber cùng Jeff Sutherland thuyết trình về Scrum tại sự kiện OOPSLA năm 1995.

Năm 2001: Tuyên ngôn Phát triển Phần mềm Linh hoạt và Liên minh Agile (Agile Alliance) ra đời. Năm 2002: Liên minh Scrum (Scrum Alliance) được thành lập.

Ken Schwaber và Jeff Sutherland cùng xây dựng định nghĩa Scrum tại ScrumGuides.org với phiên bản đầu năm 2010.

img382

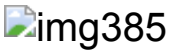
img383

## **Scrum được dùng để làm gì?**

Nhiều báo cáo cho thấy Scrum được sử dụng rộng rãi, không chỉ trong lĩnh vực phát triển phần mềm, mà còn loang ra cả những dự án sản xuất phần cứng, các đội nhóm marketing hay giáo dục.

- Phát triển phần mềm thương mại

- Phát triển ứng dụng nội bộ
- Phát triển theo đơn đặt hàng
- Các dự án mà giá đã được chốt
- Các ứng dụng tài chính
- Các ứng dụng tuân thủ chuẩn ISO 9001
- Các hệ thống nhúng
- Các hệ thống hoạt động 24x7 với yêu cầu 99,999% thời gian hoạt động.
- Hệ thống điều khiển máy bay Joint Strike Fighter
- Phát triển video game
- Phần mềm Điều khiển-Vệ tinh
- Làm ra các Website
- Phần mềm cho thiết bị cầm tay
- Điện thoại di động
- Các ứng dụng chuyển mạng
- Các dự án học tập
- Các chiến dịch marketing
- Quản lí các sự kiện công nghệ
- Sản xuất ô tô
- Quản lí lớp học hiện đại (eduScrum)
- Và nhiều nữa



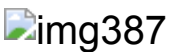
*Một cuộc họp DailyScrum ở Microsoft,*

*<ảnh Wikipedia>*



*Đội wikispeed sử dụng Scrum để cộng tác và sản xuất ô tô.*

*<ảnh: wikispeed>*



*Một lớp học dùng eduScrum ở Hà Lan,*

*<ảnh eduScrum.nl>*

## **LỢI ÍCH KHI DÙNG SCRUM**

---



Với phương pháp truyền thống, phải đến giai đoạn cuối mới thấy được sản phẩm, còn trước đó không có sản phẩm thật. Từ khi bắt đầu phát triển tính trực quan ngày càng giảm.

Với Scrum, ngay từ đầu chúng ta đã chuyển giao những phần tăng trưởng hoạt động tốt của sản phẩm và việc chuyển giao là liên tục và đều đặn, độ trực quan được duy trì ở mức cao.

Với phương pháp truyền thống, khách hàng phải đợi đến cuối để bắt đầu nhận được giá trị từ sản phẩm bởi việc dành nhiều thời gian để lập kế hoạch và thi công dự án mà không đưa ra được một sản phẩm thực sự. Do đó giá trị họ nhận được ban đầu là rất thấp.

Ngược lại, Scrum tập trung chuyển giao các tính năng hoàn chỉnh của sản phẩm ngay từ đầu. Do đó, ngay trong giai đoạn sản xuất,

khách hàng đã nhận được các giá trị. Những giá trị này ngày càng được tích lũy và lớn dần lên theo tiến độ dự án.

Ở phương pháp quản lý truyền thống, chúng ta đặt cược vào độ chính xác của kế hoạch ban đầu mặc dù khi đó chúng ta có nhiều giả định như con người, công nghệ, môi trường kinh doanh không thay đổi hoặc chúng ta biết đủ những yếu tố này, do đó rủi ro là rất cao cho tới khi kết thúc.

Đối với Scrum, mức độ rủi ro giảm rất nhanh theo cấp số mũ. Chìa khóa ở đây chính là làm việc lặp và tăng trưởng cùng với sự thanh tra và thích nghi liên tục. Những yếu tố tiềm ẩn rủi ro liên quan đến sản phẩm, công nghệ, con người, v.v., đều được phát hiện và điều chỉnh từ sớm.



Các phương pháp truyền thống không sẵn sàng cho việc thay đổi bởi, chúng ta phải chuẩn bị một kế hoạch chi tiết và cụ thể từ giai đoạn đầu, sau đó phải bám sát từng kế hoạch. Chi phí cho sự thay đổi là rất lớn, đặc biệt ở giai đoạn sau.

Ngược lại, Scrum luôn chủ động đón nhận thay đổi. Chúng ta không dành nhiều công sức để lập kế hoạch cho những giả định mà luôn chuẩn bị cho mọi sự phía trước, do đó chi phí cho thay đổi thấp hơn.

### **Những lợi ích khác...**

Mọi hoạt động đều hướng giá trị (Value-Oriented) do đó Scrum giúp chúng ta tối ưu hóa giá trị trên vốn đầu tư (ROI).

Định hướng khách hàng (Customer-Centric): tăng độ hài lòng của khách hàng bởi tất cả những gì chúng ta làm là để phục vụ khách hàng.

Giảm thiểu các “nợ kỹ thuật” bởi cách làm là luôn hoàn thành mọi thứ với chất lượng đảm bảo và linh hoạt cao để chấp nhận sự thay

đổi. Bởi thế khi phát triển, các lập trình viên cần sử dụng các phương pháp để tăng khả năng linh hoạt như mã sạch, v.v..

Chất lượng sản phẩm cao. Nhóm luôn bàn giao sản phẩm hoàn thành và thực hiện việc thanh tra, nên nếu có những vấn đề chất lượng thì nó sẽ được phát hiện và nhóm sẽ thích nghi để đưa mọi thứ trở về tiêu chuẩn.

Giảm thiểu rủi ro khi ứng dụng gặp vấn đề. Việc bàn giao sản phẩm sớm, khách hàng có thể xem và sử dụng sản phẩm, nên việc phát hiện các rủi ro sớm hơn. Thậm chí khi ứng dụng gặp vấn đề thì nhóm nhanh chóng có thể thích nghi.

Tăng năng suất và hiệu quả lao động. Scrum giúp nhóm liên tục phát hiện ra những tính năng đem lại giá trị trên vốn đầu tư cao, hoàn thành những tính năng có chất lượng và tốc độ cao hơn như trình bày ở Chương 1.

Phát triển bền vững cá nhân và đội nhóm. Việc nhóm được tự tổ chức giúp họ có cơ hội làm việc cùng với nhau, qua đó mỗi cá nhân trưởng thành hơn ở rất nhiều yếu tố như tính chủ động hơn, cam kết hơn, động lực, tri thức và kỹ năng phát triển sản phẩm. Tri thức trong nhóm được lan tỏa giữa các thành viên, do đó bản thân nhóm cũng phát triển.

## **KHUNG LÀM VIỆC SCRUM**

---

Khung làm việc Scrum gồm các vai trò của Nhóm Scrum, các Tạo tác (Artifact) được làm ra, các Sự kiện Scrum, cùng với các quy tắc ràng buộc cách thức làm việc. Tất cả chúng được xây dựng dựa trên một hệ giá trị nhất quán với các giá trị Agile và các giá trị cốt lõi của Scrum.

### **Vai trò**

Định ra trách nhiệm của từng thành viên trong nhóm Scrum tự tổ chức

- Product Owner
- ScrumMaster
- Nhóm Phát triển

## **Tạo tác**

Các công cụ hoặc sản phẩm do nhóm Scrum làm ra

- Gói tăng trưởng
- Product Backlog
- Sprint Backlog

## **Sự kiện**

Các cuộc họp và cơ chế cộng tác có cấu trúc để cộng tác hiệu quả hướng đến mục tiêu chung

- Sprint
- Lập kế hoạch Sprint
- Scrum Hằng ngày
- Sơ kết Sprint
- Cải tiến Sprint

**Ba trụ cột: Thanh tra – Minh bạch - Thích nghi**

**Năm giá trị: Dũng cảm, Tập trung, Cam kết, Cởi mở, Tôn trọng**

## **SCRUM - Một mô tả tổng quát**

Giả sử bạn đang là thành viên của một nhóm Scrum. Sáng thứ Hai, bạn sẽ cùng nhóm làm việc ngồi với nhau, họp lập kế hoạch để



quyết định cả đội sẽ làm gì trong tuần và làm như thế nào. Sau đó các công việc được chia nhỏ ra thành công việc cụ thể mà ai cũng hiểu được, dán lên bảng công việc chung. Hằng ngày từ thứ Ba đến thứ Năm, bạn sẽ tự lựa chọn việc làm phù hợp rồi bắt tay thực hiện. Vào đầu giờ sáng, trước khi làm việc, bạn cùng cả nhóm đứng trước bảng công việc chung, rà soát lại tiến độ của từng người, thông báo về một số khó khăn gặp phải trong khi làm việc. Cuộc họp đứng đó diễn ra hằng ngày, trong vòng 15 phút. Cuối mỗi ngày, bạn cố gắng hoàn thành thật tốt những việc đã lựa chọn. Vào chiều thứ 6, cả đội ngồi rà soát lại công việc xem kế hoạch đề ra từ đầu tuần có được hoàn tất không. Ngay sau đó, cả đội ngồi với nhau để đề ra cải tiến cách làm việc cho tuần sau để hiệu quả và vui vẻ hơn. Tuần sau cả nhóm lặp lại nhịp làm việc như vậy, nhưng với lưu ý về một số cải tiến sẽ được mọi người tuân thủ.

Đó là hình dung đơn giản về một nhịp làm việc của một nhóm Scrum nhỏ suốt một tuần. Trong mô tả trên, Nhóm Scrum đóng khung hoạt động của mình trong vòng 1 tuần, Scrum gọi khung thời gian này là Sprint. Buổi họp kế hoạch đầu tuần được gọi là buổi Lập kế hoạch Sprint. Buổi rà soát thành quả công việc vào thứ Sáu gọi là hoạt động Sơ kết Sprint. Hoạt động họp để tìm kiếm cách làm việc tốt hơn được gọi là họp Cải tiến Sprint. Buổi họp đứng kéo dài 15 phút mỗi ngày được gọi là Scrum Hằng ngày. Bảng công việc chứa những việc cần làm cho một tuần đó được gọi là Sprint Backlog.

Trong Nhóm Scrum kia còn có các thành viên giữ các vai trò khác nhau: một Product Owner chịu trách nhiệm quản lý danh sách những yêu cầu công việc (được gọi là Product Backlog), một ScrumMaster chịu trách nhiệm tổ chức cách thức làm việc theo Scrum cho cả nhóm, và một Nhóm Phát triển gồm những người có đủ chuyên môn để hoàn thành mục tiêu mà cả nhóm đề ra.

Trên đây là cách chúng ta hình dung đơn giản hoá về một nhóm làm việc theo Scrum trong vòng một tuần. Ở mức độ tổng quan, chúng ta thấy nhịp làm việc này rất gần gũi với cách làm việc của những nhóm làm việc gắn kết và chuyên nghiệp.

Nói theo cách của Scrum, chúng ta sẽ có một mô tả khác về cách một nhóm Scrum làm việc như dưới đây.

Product Owner tạo ra Product Backlog chứa các yêu cầu của dự án với các hạng mục được sắp theo thứ tự ưu tiên. Đội sản xuất sẽ hiện thực hóa dần các yêu cầu của Product Owner với sự lặp đi lặp lại các giai đoạn nước rút từ 1 đến 4 tuần làm việc (gọi là Sprint) với đầu vào là các hạng mục trong Product Backlog, đầu ra là các gói phần mềm hoàn chỉnh có thể chuyển giao được (Potentially Shippable Product Increment). Trước khi cả nhóm cùng đưa nước rút trong Sprint, đội sản xuất cùng họp với Product Owner để lập kế hoạch cho từng Sprint. Kết quả của buổi lập kế hoạch (theo cách làm của Scrum) là Sprint Backlog chứa các công việc cần làm trong suốt một Sprint. Trong quá trình phát triển, nhóm sẽ phải cập nhật Sprint Backlog và họp hằng ngày (Daily Scrum) để chia sẻ tiến độ công việc, tái lập kế hoạch cũng như các vướng mắc trong quá trình làm việc cùng nhau. Nhóm được trao quyền tự quản lí và tổ chức lấy công việc của mình để hoàn thành công việc trong Sprint. Khi kết thúc Sprint, nhóm tạo ra gói phần mềm có chức năng hoàn chỉnh, sẵn sàng chuyển giao (shippable) cho khách hàng. Buổi họp Sơ kết Sprint (Sprint Review) ở cuối Sprint sẽ giúp khách hàng thấy được nhóm đã có thể chuyển giao những gì, còn những gì phải làm hoặc còn gì phải thay đổi hay cải tiến. Sau khi kết thúc việc đánh giá Sprint, ScrumMaster và Nhóm Phát triển cùng tổ chức họp Cải tiến Sprint (Sprint Retrospective) để tìm kiếm các cải tiến trước khi Sprint tiếp theo bắt đầu. Điều này sẽ giúp nhóm liên tục học hỏi và trưởng thành qua từng Sprint.

Các Sprint sẽ được lặp đi lặp lại cho tới khi nào các hạng mục trong Product Backlog đều được hoàn tất hoặc khi Product Owner quyết định có thể dừng dự án căn cứ tình hình thực tế. Do sử dụng chiến thuật “có giá trị hơn làm trước” nên các hạng mục mang lại nhiều giá trị hơn cho chủ dự án luôn được hoàn tất trước. Vì vậy Scrum luôn mang lại giá trị cao nhất cho người đầu tư cho dự án. Do quy trình luôn luôn được cải tiến, nhóm Scrum thường có năng suất lao động rất cao. Đây là hai lợi ích to lớn mà Scrum mang lại cho tổ chức.

## BA TRỤ CỘT CỦA SCRUM

---

### **Minh bạch (transparency)**

Muốn thành công với Scrum, thông tin liên quan tới quá trình phát triển phải minh bạch và thông suốt. Các thông tin đó có thể là: tầm nhìn (vision) về sản phẩm, yêu cầu khách hàng, tiến độ công việc, các khúc mắc và rào cản v.v.. Từ đó mọi người ở các vai trò khác nhau có đủ thông tin cần thiết để tiến hành các quyết định có giá trị để nâng cao hiệu quả công việc. Các công cụ và cuộc họp trong Scrum luôn đảm bảo thông tin được minh bạch cho các bên.

### **Thanh tra (inspection)**

Công tác thanh tra liên tục các hoạt động trong Scrum đảm bảo cho việc phát lộ các vấn đề cũng như giải pháp để thông tin đa dạng và hữu ích đến được với các bên tham gia dự án. Truy xét kĩ càng và liên tục là cơ chế khởi đầu cho việc thích nghi và các cải tiến liên tục trong Scrum.

### **Thích nghi (adaptation)**

Dựa trên các thông tin minh bạch hóa từ các quá trình thanh tra và làm việc, Scrum có thể phản hồi các thay đổi một cách tích cực, nhờ đó mang lại thành công cho dự án. Các nỗ lực minh bạch và thanh tra đều để hướng tới hành động thích ứng nhanh chóng, hiệu quả.

## NĂM GIÁ TRỊ CỦA SCRUM

---

### **GIÁ TRỊ SCRUM**

Dũng cảm ⇒ Để một người dám nói vấn đề của mình và chấp nhận rất nhiều loại rủi ro khi thay đổi, cam kết, họ cần là người dũng cảm. Về cơ bản, những giá trị khác không thể có nếu bạn không có sự dũng cảm.

Tập trung ⇒ Mọi người tập trung vào công việc trong Sprint & Mục tiêu Sprint của Nhóm.

Khi nhóm Phát triển đã cam kết với những việc trong Sprint, họ cần phải tập trung để hoàn thành những gì mà mình đã cam kết.

Cam kết ⇒ Mỗi thành viên cam kết với các thành viên khác về những điều mình làm ông việc đã được chọn ở buổi Lập kế hoạch Sprint.

Ngoài ra, chúng ta liên tục cải tiến tức là thay đổi để trở thành một cá nhân tốt hơn, nhóm tốt hơn và tổ chức tốt hơn. Chúng ta luôn phải thay đổi giữ vững lợi thế cạnh tranh và phục vụ khách hàng. Thực hiện thay bao giờ cũng rất khó khăn, do đó chỉ với sự cam kết chúng ta mới có làm được.

Cởi mở ⇒ Mọi thứ cần phải rõ ràng, minh bạch để mọi người có thể làm việc hiệu quả. Phát mềm rất phức tạp, một người không thể nhìn và hiểu được hết tất cả mọi vấn đề. Do đó nếu mọi người không cởi mở với nhau, thông tin bị che giấu rất ệu quả công việc khó lòng có thể nâng cao.

Tôn trọng ⇒ Khi thiếu tôn trọng, mọi người khó thành thật trong chia sẻ. Ví dụ, khi một người không biết một điều gì đó và đi hỏi người khác. Người trả lời thay vì mong muốn giúp đỡ để người hỏi trở nên tốt hơn, độc lập hơn lại phản nản, đánh giá người hỏi thì lần sau người hỏi sẽ khó mà cởi mở và nói sự thật được.

Không có tôn trọng, khó có sự cởi mở. Những công ty có văn hóa đổ lỗi khó có sự cởi mở.

## **SPRINT - TRÁI TIM CỦA SCRUM**

---

Sprint là phân đoạn ngắn để tạo ra phần chức năng và tính năng hoàn chỉnh. Các Sprint diễn ra nối tiếp nhau. Độ dài của Sprint từ một đến bốn tuần. Giữ độ dài Sprint không đổi để tạo nhịp cho nhóm. Sprint càng ngắn, chi phí quản lí càng lớn. Sprint càng dài

càng khó giữ sự tập trung của nhóm cũng như giảm khả năng linh hoạt với sự thay đổi. Mỗi Sprint đều có Mục tiêu Sprint (Sprint Goal) tổng hợp những chức năng và tính năng nhóm cam kết hoàn thành trong Sprint. Sản phẩm được thiết kế, lập trình và kiểm thử trong Sprint.

Cùng công việc nhưng được tổ chức khác và với ít yêu cầu hơn. Những lãng phí lớn sẽ bị phát lộ và có thể được xóa bỏ một cách hệ thống.



*Theo tài liệu thuyết trình “Scrum for Vietnam” của Ken Schwaber, 2012.*

### **Các loại phát triển tuần tự và chồng lấp**

Có nhiều loại phát triển chồng lấp:

- Loại A: Các giai đoạn tách biệt.
- Loại B: Các giai đoạn có sự chồng lấp nhau, nhưng không hoàn toàn
- Loại C: Các giai đoạn hoàn toàn chồng lấp. Mọi việc đều được làm trong một khoảng thời gian ngắn



### **Các loại phát triển tuần tự & chồng lấp**

⇒ Dừng và Nghĩ

**Nhóm của bạn có thể làm việc theo kiểu chồng lấp loại C không? Thử hình dung ai làm gì, như thế nào.**

**Ví dụ về chồng lấp**

Tất cả thành viên ở các vùng kĩ năng khác nhau đồng thời tham gia vào việc phát triển.

img415

Mọi công việc để hoàn thành sản phẩm từ phân tích yêu cầu, viết kiểm thử, thiết kế, lập trình, kiểm thử đơn vị, dựng sản phẩm được thực hiện đồng thời.

### **Đóng khung thời gian cho Sprint**

Không thay đổi yêu cầu sản phẩm trong suốt Sprint nhưng nếu Nhóm Phát triển phát hiện thiếu một công việc để hoàn thành mục tiêu (ví dụ phát sinh bug) thì có thể thêm vào. Nếu một ai đó (ví dụ Product Owner) yêu cầu thay đổi yêu cầu thì Nhóm Phát triển có thể từ chối, ScrumMaster đảm bảo Product Owner không được yêu cầu việc đó với Nhóm Phát triển.

Cần tạo không gian tự chủ cho nhóm. Nhóm được quyết định cách làm để hoàn thành công việc. Không ai được yêu cầu Nhóm Phát triển cách làm việc kể cả Scrum- Master.

Tránh quản lí vật (micro-manage). Không trực tiếp giao việc cho thành viên nhóm. Hãy để họ tự quyết định. Nếu khả năng quyết định của họ chưa thực sự tốt, hãy giúp đỡ họ bằng các đưa ra các quy tắc hoặc đào tạo họ.

ScrumMaster – “chó chặn cừu”: tức là đảm bảo Nhóm Phát triển tập trung làm việc mà không bị làm phiền.

Tránh thay đổi Mục tiêu Sprint, thành viên Nhóm Phát triển, chất lượng mục tiêu, phạm vi của tính năng. Khi có thay đổi rất hệ trọng hoặc bất khả kháng thì có thể hủy Sprint hoặc đàm phán với Product Owner và Nhóm Phát triển. Tuy nhiên nếu việc đó diễn ra liên tục thì vấn đề lớn đang xảy ra. Nhóm Phát triển không thể tập trung để làm việc hiệu quả nhất.

### **Kết quả cuối mỗi Sprint: Gói tăng trưởng**

Gói tăng trưởng (Increment) là cách gọi tắt của thuật ngữ “gói tăng trưởng có khả năng chuyển giao được”, là tổng tất cả những hạng mục đã hoàn thành của Product Backlog của Sprint và các Sprint trước. Tăng trưởng này phải “hoàn thành” có thể chuyển giao được và thỏa mãn Định nghĩa Hoàn thành mà nhóm Scrum đã thống nhất trước đó. Gói tăng trưởng này phải có khả năng sử dụng được bởi khách hàng chứ không phải là sản phẩm alpha hay bản mẫu. Tuy nhiên việc có chuyển giao cho khách hàng hay không tùy thuộc vào nhiều yếu tố như chiến lược phát hành, kế hoạch kinh doanh, v.v.. Gói tăng trưởng phải được trình diễn ở Sơ kết Sprint.



## **Những lợi ích từ gói tăng trưởng**

Việc có một phần tăng trưởng chuyển giao được sau một khoảng thời gian ngắn sẽ giúp nhanh chóng có được phản hồi từ người dùng hay khách hàng. Các phản hồi này sẽ được ghi nhận để đưa ra những điều chỉnh cho sản phẩm nếu cần thiết. Ví dụ một ứng dụng dành cho các thiết bị di động, sau một khoảng thời gian ngắn với một lượng tính năng đủ dùng ban đầu thì chúng ta có thể phát hành tới một lượng người dùng nhất định. Sau đó, chúng ta sẽ lần lượt tích hợp thêm những tính năng khác hoặc có những điều chỉnh hợp lý dựa trên phản hồi của người dùng. Những phản hồi này thực sự giá trị bởi vì chúng xuất phát từ người dùng thật và dựa trên sản phẩm thật.

Việc chuyển giao được những phần sản phẩm nhanh chóng sẽ giúp chúng ta sớm thu được giá trị từ sản phẩm. Ví dụ với một ứng dụng dành cho thiết bị di động như đã nói ở trên, việc sớm mang đến tay người dùng sẽ giúp đẩy nhanh việc phát triển khách hàng lên song song với phát triển sản phẩm. Điều này có nghĩa là chúng ta cũng sớm thấy được giá trị của sản phẩm và giá trị này cũng đến tay người dùng sớm hơn.

Phần tăng trưởng chuyển giao được sẽ gia tăng tính minh bạch trong quá trình sản xuất. Sau những khoảng thời gian ngắn, chúng ta đã có được những phần sản phẩm thực sự “hoàn thành” mà tất

cả mọi người đều có thể nhìn thấy được, tương tác được, thay vì chỉ là những báo cáo, những thiết kế hay bản mẫu vốn tiềm ẩn nhiều rủi ro.

Phần tăng trưởng giúp giảm thiểu rủi ro, xét về cả mặt kinh doanh lẫn kĩ thuật. Về mặt kinh doanh, chúng ta sớm biết được thực sự tính năng đó có mang lại giá trị hay không. Chẳng hạn, chúng ta muốn phát triển một phiên bản của hệ thống hiện tại dành riêng cho các thiết bị di động, nhưng không biết được phản ứng của thị trường. Do vậy, việc chúng ta nhanh chóng đưa ra được một phiên bản dùng được của sản phẩm sẽ giúp thăm dò thị trường và đưa ra các quyết định điều chỉnh hợp lý.

Về mặt kĩ thuật, chúng ta sớm phát hiện được những khó khăn và đưa ra những giải pháp điều chỉnh trước khi quá muộn. Chẳng hạn, bạn muốn sử dụng một công nghệ mới vào trong hệ thống của mình, việc sớm có được tính năng thật sẽ giúp kiểm chứng được liệu công nghệ đó có hỗ trợ tốt cho nhu cầu của mình hay không.

**⇒ Dừng và Nghĩ**

**Hãy nghĩ làm cách nào để nhóm của bạn luôn chuyển giao giá trị cho người dùng trong vòng 30 ngày.**

## **CÂU HỎI ỨNG DỤNG**

1. Hai trụ cột Thanh tra và Thích nghi của quá trình bạch lý thực nghiệm bị ảnh hưởng như thế nào nếu trụ cột Minh bạch không đảm bảo?
2. Mô tả mối quan hệ giữa ba trụ cột của quá trình quản lí thực nghiệm.
3. Phân tích ưu và nhược điểm của việc Nhóm Phát triển tự xác định cách hoàn thành công việc trong phân đoạn thay vì một người thực hiện và truyền đạt lại.



4. Bạn có thể áp dụng Scrum để quản lí những công việc nào của mình?

5. Nhóm Scrum quản lí thay đổi như thế nào?

6. Mô tả lợi ích của bên liên quan (ví dụ: khách hàng, nhà phát triển, quản lí, v.v.) từ việc áp dụng Scrum?

7. Nếu áp dụng Scrum vào một dự án/công việc của bạn thì thành viên ở những chuyên môn khác nhau (kiểm thử, thiết kế, v.v.) sẽ tham gia theo trình tự như thế nào trong một Sprint?

8. Những lợi ích khi Nhóm Phát triển được làm việc trong một Sprint đóng khung?

9. Những khó khăn khi đóng khung một Sprint ở dự án/công việc của bạn? Cách vượt qua?

# 3NHÓM SCRUM

Trong chương này...

- Nhóm liên chức năng và tự tổ chức
- Các giai đoạn phát triển của nhóm làm việc
- Nhóm Scrum
- ScrumMaster
- Product Owner
- Nhóm Phát triển
- Các nghề nghiệp đặc thù Scrum

*Nếu tất cả mọi người cùng nhau tiến về phía trước, thì thành công sẽ tự nó đến.*

**Henry Ford**

## NHÓM TỰ TỔ CHỨC LIÊN CHỨC NĂNG

---



Tuyên ngôn Agile ghi rõ “Cá nhân và tương tác hơn là quy trình và công cụ”. Bí quyết thành công của Scrum nằm ở các cá nhân và sự tương tác qua lại giữa các cá nhân trong nhóm Scrum.

Nhóm Scrum bao gồm Product Owner, Nhóm Phát triển, và ScrumMaster. Nhóm Scrum là nhóm tự tổ chức (self-organizing) và liên chức năng (cross-functional). Nhóm tự tổ chức tự mình chọn cách thức tốt nhất để hoàn thành công việc chứ không bị chỉ đạo bởi ai đó bên ngoài nhóm. Nhóm liên chức năng có đủ kỹ năng cần thiết để

hoàn thành công việc mà không phụ thuộc vào bất kì người ngoài nào khác ngoài nhóm. Mô hình nhóm trong Scrum được thiết kế để tối ưu hóa sự linh hoạt, sáng tạo và năng suất. Nhóm Scrum chuyển giao sản phẩm theo phân đoạn lặp đi lặp lại và tăng dần, tối đa hóa cơ hội cho các phản hồi. Việc chuyển giao tăng dần (incremental) các gói sản phẩm đạt tiêu chuẩn “Hoàn thành” đảm bảo một phiên bản có thể sử dụng được của sản phẩm.

Nhóm Scrum có hai đặc điểm quan trọng là: tự tổ chức (self-organizing) và liên chức năng (cross-functional).

Tự tổ chức có nghĩa là nhóm cùng ra quyết định, cùng tổ chức công việc thông qua bàn bạc, phân công ngang hàng hướng đến mục tiêu chung. Trong nhóm tự tổ chức, không ai giao việc cho ai, mà chủ động lập kế hoạch, phân chia công việc hợp lý dựa trên các vai trò được định nghĩa rõ ràng từ trước. Các vai trò này được trao quyền để hoàn thành công việc.

**Liên chức năng** là một cơ cấu nhóm không quá mới mẻ, và không phải Scrum phát minh ra. Đó là một nhóm bao gồm nhiều cá nhân với các chuyên môn khác nhau đủ năng lực được kết hợp lại cùng làm việc hướng tới một mục tiêu chung.

Trong đội dự án, các cá nhân có thể đến từ nhiều phòng ban chức năng khác nhau, cũng có thể xuất phát từ bên ngoài (khách hàng, các cá nhân có liên quan, chuyên gia tư vấn, v.v.). Nhưng khi đã thành một đội (team), thì các cá nhân làm việc tập trung cho đội như là một đơn vị (unit) để hoàn tất mục tiêu chung. Bên trong nhóm liên chức năng không có các nhóm nhỏ khác. Ví dụ: một Nhóm Scrum “Alpha” được thành lập với 1 Product Owner, 1 Scrum- Master, 2 Tester, 5 Programmer, 1 Architect, 1 UX Designer sẽ không phân chia chức năng thành các nhóm nhỏ khác như nhóm Testing (2 người), nhóm Development (5 người) ... nữa, mà chỉ có một nhóm duy nhất “Alpha” với các cá nhân có chuyên môn khác nhau, hợp thành một đội thống nhất để làm việc hướng tới sản phẩm cần phát triển. Trong cách nói của các tác giả Scrum, tester hay analyst ... đều là developer, họ là nhà phát triển nhưng có chuyên môn đặc thù là kiểm thử hay phân tích; developer trong trường hợp này không có ý

nghĩa là coder/programer. Việc xóa nhòa các định danh công việc (Job Title) này có mục đích là để hướng mọi người vào một mục tiêu chung, không phân biệt “nhãn mác” (title): phát triển phần mềm.

Khác với nhóm liên chức năng, nhóm chức năng (functional team) thường chỉ phụ trách một loại công việc đặc thù. Ví dụ, phòng thiết kế thì không lập trình, phòng test thì không có ai thiết kế. Công việc của nhóm chức năng thường có tính cô lập cao.

Có nhiều hiểu lầm phổ biến đối với người mới nghe tới Agile lần đầu, đơn cử:

*“Cá nhân trong nhóm liên chức năng biết làm tất cả mọi việc, trong nhóm Scrum không có ai là tester hết, không cần designer trong nhóm Scrum nữa, chỉ có developer thôi”.*

Thực chất là mỗi người mỗi việc.

Hay *“Nhóm liên chức năng có nghĩa là ai cũng có thể làm được việc thay thế người khác, không cần phân vai rõ ràng, một người có vấn đề sẽ không ảnh hưởng đến cả nhóm”.* Thực ra đây chỉ là trạng thái lí tưởng, không phải là cái bắt buộc ở một nhóm liên chức năng.

Hay *“Cứ liên chức năng thì siêu năng suất”.* Không hẳn, năng suất còn do cách thức phối hợp trong nhóm để đạt được mục tiêu chứ không đơn giản là chuyện cấu hình.

## **Các giai đoạn phát triển nhóm**

img431

Một nhóm cộng tác trải qua những giai đoạn khác nhau từ khi được bắt đầu thành lập cho đến khi hoạt động ổn định theo thời gian. Việc nhận diện các giai đoạn của nhóm là quan trọng để đưa ra các quyết định chính xác nhằm đảm bảo nhóm đạt được hiệu quả tốt nhất.

Tuckman (một nhà tâm lý học người Mỹ) đã đưa ra một mô hình để giải thích các giai đoạn này. Mô hình Tuckman chia chặng đường của một nhóm thành 4 giai đoạn: Forming (Hình thành), Storming (Sóng gió), Norming (Ổn định) và Performing (Hiệu suất cao). Sau này, Tuckman đã thêm vào một giai đoạn thứ 5 đó là Adjourning (Thoái trào).

ScrumMaster có vai trò rất quan trọng đối với việc xây dựng và phát triển nhóm.

Nhiệm vụ của vai trò này là quan sát để biết nhóm của mình đang ở giai đoạn nào để có những hành động hỗ trợ cần thiết nhằm đẩy nhanh sự phát triển hoặc duy trì sự ổn định của nhóm. Tìm các giải pháp tháo gỡ những trở ngại đối với nhóm trong quá trình phát triển.

## SCRUMMASTER

---



ScrumMaster là một vai trò then chốt trong Nhóm Scrum có trách nhiệm đảm bảo Scrum được vận hành đúng bằng việc tuân thủ nguyên lý, các kỹ thuật và quy tắc Scrum nhằm hướng đến kết quả tốt nhất. ScrumMaster là người tháo gỡ khó khăn và tạo điều kiện cho nhóm làm việc tốt nhất có thể, huấn luyện đội nhóm thành thực Scrum và hưởng lợi ích từ Scrum.

ScrumMaster không trực tiếp tham gia vào công việc làm ra sản phẩm, nhưng là chất kết dính để các bên phối hợp với nhau tạo ra sản phẩm tốt. ScrumMaster không phải là quản lý của Nhóm mà là một lãnh đạo theo phong cách phục vụ (Servant Leader). Với vai trò là lãnh đạo phục vụ, ScrumMaster cung cấp các dịch vụ cho Product Owner, Nhóm Phát triển và Tổ chức.

*Trích Hướng dẫn Scrum*

**Với Nhóm Phát triển, ScrumMaster phục vụ theo nhiều cách như sau:**

Đào tạo căn bản Scrum trong trường hợp các thành viên chưa biết về khung làm việc này.

Đảm bảo Nhóm Phát triển thực hiện tốt các sự kiện trong Scrum, bao gồm việc tuân thủ khung thời gian, đúng mục đích và nội dung của từng sự kiện.

Hỗ trợ Nhóm Phát triển tìm kiếm và sử dụng hiệu quả các công cụ hỗ trợ phát triển, chẳng hạn công cụ để quản lý Sprint Backlog, duy trì biểu đồ Sprint Burndown, các hệ thống quản lý mã nguồn, kiểm thử tự động, tích hợp liên tục cũng như các công cụ để giao tiếp trong Nhóm Phát triển như email hay hệ thống trao đổi và chia sẻ trực tuyến, v.v..

Bảo vệ Nhóm Phát triển trước những can thiệp bên ngoài trong quá trình triển khai một Sprint. Ví dụ, ScrumMaster cần giải thích rõ cho Product Owner những tác hại mà việc này gây ra cho Nhóm Phát triển và sản phẩm khi Product Owner muốn thay đổi các hạng mục Product Backlog đã lựa chọn của một Sprint đang diễn ra, đồng thời hướng dẫn Product Owner đưa các thay đổi này vào Sprint tiếp theo, v.v..

Đảm bảo thông tin được minh bạch và thông suốt trong Nhóm Phát triển, chẳng hạn như thông qua việc nhắc nhở nhóm luôn cập nhật Sprint Backlog và biểu đồ Sprint Burndown, đảm bảo Nhóm Phát triển thực hiện đúng sự kiện Scrum Hằng ngày, v.v..

Đảm bảo Nhóm Phát triển có đầy đủ các tài nguyên cần thiết phục vụ sản xuất. Ví dụ như không gian làm việc, các công cụ hỗ trợ và những tài nguyên khác khi có nhu cầu, v.v..

Trong trường hợp Nhóm Phát triển gặp khó khăn với một vấn đề kỹ thuật nhất định, nghĩa là Nhóm Phát triển gặp trở ngại, ScrumMaster có thể giải quyết bằng cách tìm kiếm sự trợ giúp từ các chuyên gia bên ngoài nhóm.

*Trích Hướng dẫn Scrum*

## **ScrumMaster phục vụ tổ chức bằng nhiều cách, cụ thể như:**

Giúp nâng cao năng lực Agile\Scrum ở tổ chức, chẳng hạn thông qua việc đào tạo Scrum cho các cấp quản lí và các phòng ban khác nhau, thành lập các nhóm học tập Scrum trong tổ chức.

Làm việc với các ScrumMaster khác để lập kế hoạch triển khai Scrum trong phạm vi tổ chức, gia tăng hiệu quả của việc áp dụng Scrum trong tổ chức của mình.

Giúp đỡ nhân viên và các bên hữu quan hiểu và sử dụng được Scrum cũng như quy trình phát triển sản phẩm thực nghiệm (empirical product development).

Tạo ra sự thay đổi làm tăng năng suất của Nhóm Scrum.

ScrumMaster phục vụ các đối tượng khác nhau với các công việc khác nhau, các công việc này thay đổi theo các giai đoạn của quá trình phát triển.

Đầu tiên, ScrumMaster làm việc với Tổ chức để chuẩn bị và có thể bắt đầu áp dụng Scrum. Đồng thời, ScrumMaster tập trung vào việc xây dựng Nhóm Phát triển cùng với huấn luyện, giúp đỡ Product Owner để đưa nhóm đi vào hoạt động. Ở giai đoạn này, các công việc liên quan đến kĩ thuật phát triển chỉ dừng lại ở mức tối thiểu, đảm bảo vẫn hoàn thành được công việc.

Sau đó, ScrumMaster có thể tạm thời gác lại các công việc liên quan đến tổ chức để tập trung vào việc giúp Nhóm làm việc ổn định hơn, từng bước nâng cao các kĩ năng của từng cá nhân và của cả Nhóm. Dần dần ScrumMaster có thể giới thiệu và hỗ trợ nhóm áp dụng các kĩ thuật phát triển để nâng cao năng suất, chất lượng sản phẩm.

Khi Nhóm Scrum đã thực sự ổn định và vận hành tốt, ScrumMaster có thể toàn tâm toàn ý để tập trung vào việc tìm kiếm, giới thiệu và hỗ trợ nhóm áp dụng các kĩ thuật phát triển tốt hơn. Cùng với đó là

hoạt động để mở rộng Scrum ra tổ chức, giao tiếp và trao đổi giữa các nhóm, hướng đến việc phát triển Scrum bền vững.

### *Trích Hướng dẫn Scrum*

**ScrumMaster phục vụ Product Owner theo nhiều cách như sau:**

Tìm kiếm các kĩ thuật để quản lí hiệu quả Product Backlog;

Giao tiếp tích cực với Nhóm Phát triển về tầm nhìn, mục đích, và các hạng mục của Product Backlog;

Dạy cho Nhóm Phát triển biết cách tạo ra các hạng mục Product Backlog thật rõ ràng và đơn giản;

Hiểu rõ việc lập kế hoạch dài hạn sản phẩm trong một môi trường thực nghiệm;

Hiểu rõ và thực hành sự linh hoạt (agility); và,

Thúc đẩy các sự kiện Scrum theo yêu cầu hoặc khi cần thiết.

### *Trích Hướng dẫn Scrum*



⇒ **Dừng và Nghĩ**

**Bạn có thể viết lại mô tả công việc của ScrumMaster không?**

**Công việc của ScrumMaster**

### **1. Tổ chức các cuộc họp**

Công việc của ScrumMaster gắn liền với nhiều cuộc họp chính thức và không chính thức như họp Lập kế hoạch, họp Sơ kết Sprint, họp Cải tiến Sprint, họp viết Yêu cầu, họp Scrum Hằng ngày, và nhiều buổi họp đột xuất khác.



## **2. Thanh tra, thu thập và bạch hoá thông tin**

Để “quan sát” ScrumMaster có thể bắt đầu với hàng loạt câu hỏi:

Mình có nhìn thấy bảng công việc không?

Khách hàng có biết việc gì đang diễn ra trên bảng không?

Có gì được cập nhật từ hôm qua đến nay không?

Có nhìn thấy burndown\burnup không?

Có điểm nào bất thường trên Biểu đồ Burndown không?

Để điều tra, “đào bới” thêm, ta có thể dùng các câu hỏi Socratic:

Tôi nhận thấy <tình huống>, chúng ta sẽ làm gì? Tôi quan sát thấy <tình huống>, nó có quan trọng không? Tôi thấy <cảm giác>, bạn có thấy điều đó? Chúng ta sẽ cố tìm lý do của <tình trạng>? Bạn nghĩ chúng ta cần làm gì? Ai có ý tưởng gì về <tình trạng>? Điều này có hiệu quả không? Bạn đã quyết định điều gì? Bạn nên làm gì [với tình huống\ vấn đề này]?

Để điều tra nguyên nhân của vấn đề, Scrum- Master còn phải thành thạo và liên tục sử dụng 5WHYs (Hỏi 5 lần tại sao) để tìm ra cách thúc đẩy nhóm tiến lên.

Các thông tin quan sát được, điều tra được cần lưu trữ và tổ chức khoa học để tiện bề truy xuất (cá nhân và đội nhóm)

## **3. Loại bỏ trở ngại**

Sử dụng 5WHYS để tìm những phương án cho những tình huống khó khăn. Ngoài ra, ScrumMaster có thể duy trì một danh sách trở ngại (Impediment Backlog) và cùng với các bên để tháo gỡ dần dần.

## **4. Tìm kiếm cải tiến**

Product Owner của tôi làm việc thế nào? Nhóm Phát triển đang làm việc thế nào? Các kĩ thuật đang được dùng thế nào? Tổ chức đang làm việc ra sao? Ai cần được huấn luyện về cái gì? Quy trình có dư thừa cái gì không? Có thể mua sắm hay làm mới công cụ gì để tăng năng suất và gia tăng không khí vui vẻ không?

## **5. Huấn luyện (Coaching) Scrum cho nhóm**

Thông qua seminar, workshop, coaching bằng hội thoại hoặc làm mẫu, v.v. để giúp thành viên thành thục Scrum và cách thức giải quyết vấn đề.

### **Lưu ý:**

*ScrumMaster là:*

- Lãnh đạo Phục vụ
- Cố vấn
- Nhân tố thay đổi
- Người giải quyết vấn đề
- “Chó chặn cừu”
- Người làm việc toàn thời gian

*...không là:*

- Quản lí Dự án
- Nắm quyền đối với nhóm
- Kiến trúc sư trưởng
- Ai kiêm nhiệm cũng được

## **Quản lí Dự án (PM) và ScrumMaster**



## **Danh mục kiểm tra của ScrumMaster**

### **Công việc hằng ngày**

☐ Nhóm của bạn có ở trạng thái tốt không?

- Mục tiêu rõ ràng (những kỳ vọng và quy định phải rõ ràng, và mục tiêu có thể đạt được, phù hợp với kỹ năng và khả năng của mỗi người).
- Tập trung và có trọng điểm, tập trung cao độ vào một lĩnh vực nhất định cần được chú ý.
- Không có cảm giác tự ti, đề cao hành động và nhận thức.
- Phản hồi trực tiếp và ngay lập tức (nhanh chóng nhìn thấy các thành công và thất bại của một chuỗi hoạt động, nhờ thế có thể điều chỉnh hành vi nếu cần thiết).
- Cân bằng giữa cấp độ khả năng và thử thách (hoạt động đưa ra không quá khó cũng không quá dễ).
- Mỗi người đều có khả năng tự kiểm soát trong mỗi tình huống hay hoạt động.
- Mỗi hoạt động đều hiển nhiên đem lại kết quả, vì thế không cần quá nhiều cố gắng trong hành động.

☐ Các thành viên có thích nhau không, có thư giãn cùng nhau không, và có vui mừng trước thành công của những thành viên khác không?

☐ Các thành viên nhóm có cùng nhau giữ cho mỗi người đều phải đạt được những tiêu chuẩn cao, và thúc đẩy mỗi người đều phát triển?

- Có vấn đề hoặc cơ hội nào mà nhóm đang không thảo luận cùng nhau vì những vấn đề hoặc cơ hội đó có thể gây ra tình trạng quá khó chịu trong nhóm không?
- Nhóm có tập trung liên tục vào các mục tiêu Sprint không? Có thể bạn nên thực hiện một bước kiểm tra giữa Sprint để có thể tái rà soát các tiêu chuẩn chấp thuận của các hạng mục trong Product Backlog đã cam kết hoàn thành trong Sprint hiện tại.
- Bảng phân công nhiệm vụ của Sprint có phản ánh đúng những công việc mà nhóm đang thực sự làm không? Cần nhận thức rõ “vấn đề tiềm ẩn” của những công việc không được nêu ra và những nhiệm vụ có thể tốn hơn một ngày mới có thể hoàn thành. Những công việc không liên quan đến những cam kết trong Sprint sẽ là những trở ngại cho việc hoàn thành các cam kết đó.
- Bảng phân công nhiệm vụ của nhóm bạn có cập nhật không?
- Các thành viên nhóm có biết về các công cụ tự quản lí của nhóm không, các công cụ đó có tiện dụng không?
- Những công cụ quản lí có được những người ngoài nhóm tôn trọng đúng mức không? Sự giám sát quá mức đối với các hoạt động thường nhật của những người bên ngoài nhóm có thể hủy hoại sự minh bạch và cơ chế tự quản lí trong nhóm.
- Các thành viên nhóm có tự nguyện nhận nhiệm vụ không?
- Sự cần thiết của việc hoàn trả nợ kĩ thuật, việc dần dần sẽ giúp cho việc lập trình trở nên dễ chịu hơn, đã được ghi nhận rõ ràng trong khái niệm hoàn thành chưa?
- Các thành viên nhóm có đang cùng nhau chịu trách nhiệm về mọi khía cạnh của sản phẩm chung (kiểm thử, tài liệu cho người dùng, v.v.) mà không quan tâm đến chức danh của mỗi người không?

## **Công việc theo Sprint**

- Product Backlog có được sắp xếp theo hiểu biết mới nhất của nhóm không?
- Các yêu cầu và mong muốn từ tất cả các bên liên quan có được ghi nhận trong Product Backlog không? Ghi nhớ: backlog là quan trọng.
- Khối lượng của Product Backlog có thể quản lí được không? Để duy trì số lượng hạng mục trong giới hạn có thể quản lí được, hãy giữ những hạng mục chi tiết ở trên cùng, còn các hạng mục trừu tượng nói chung ở dưới. Sẽ phản tác dụng nếu chúng ta phân tích quá sâu vào những hạng mục ở phía dưới của Product Backlog. Những yêu cầu đặt ra sẽ thay đổi trong những cuộc trao đổi liên tục giữa sản phẩm đang phát triển và các bên liên quan hoặc khách hàng.
- Có yêu cầu nào (đặc biệt là những yêu cầu ở phía trên cùng của Product Backlog) có thể được thể hiện tốt hơn bằng những user story độc lập, có thể thương lượng được, có thể đánh giá được, có thể ước lượng được, nhỏ, và có thể kiểm thử được?
- Bạn đã giải thích cho Product Owner của bạn về nợ kĩ thuật và cách để tránh nó chưa? Một trong những giải pháp có thể là thêm việc viết kiểm thử tự động và tái cấu trúc vào định nghĩa ‘hoàn thành’ cho mỗi hạng mục.
- Backlog có phải là một biểu đồ thông tin, mà các bên liên quan có thể lập tức nhận thấy được không?
- Nếu bạn đang sử dụng một công cụ tự động trong quản lí backlog, mọi người có biết cách để sử dụng nó dễ dàng không? Các công cụ quản lí tự động cũng dẫn tới nguy cơ trở thành sự tắc nghẽn thông tin nếu thiếu đi sự truyền tải thông tin chủ động từ ScrumMaster.
- Bạn có thể giúp truyền tải thông tin bằng các bản in cho mỗi người không?

- Bạn có thể giúp truyền tải thông tin bằng cách tạo ra các biểu đồ to và rõ ràng không?
- Bạn đã giúp Product Owner sắp xếp các hạng mục backlog vào các thời điểm phát hành thích hợp hoặc trong những nhóm ưu tiên chưa?
- Có phải mỗi người trong nhóm đều nhận thức được liệu kế hoạch phát hành còn phù hợp với thực tế không? Bạn có thể thử cho mọi người xem Biểu đồ tương quan sản phẩm/phát hành với thời gian thực hiện sau khi các hạng mục đã được xác nhận “hoàn thành” trong mỗi cuộc họp Sơ kết Sprint. Biểu đồ thể hiện tỉ lệ các hạng mục Product Backlog đã được hoàn thành và những hạng mục mới được thêm vào để cho phép phát hiện sớm những biến động trong quy mô hoặc kế hoạch thực hiện.
- Product Owner của bạn đã điều chỉnh kế hoạch phát hành sau buổi họp Sơ kết Sprint gần nhất chưa? Chỉ có số ít Product Owner đã bàn giao sản phẩm được kiểm thử đầy đủ đúng hạn sắp xếp lại kế hoạch phát hành mỗi Sprint. Điều này có thể sẽ khiến cho một vài sản phẩm cần được phát hành sau vì có những việc quan trọng hơn được phát hiện ra.
- Bạn đã từng thử nhiều cách thức và địa điểm cho các buổi họp Cải tiến Sprint chưa?

## **PRODUCT OWNER**

---

Product Owner là một trong ba vai trò Scrum. Vai trò này chịu trách nhiệm định hướng sản phẩm trong suốt quá trình sản xuất. Nhiệm vụ của Product Owner là tối ưu hóa giá trị của sản phẩm thông qua việc tận dụng tốt nhất khả năng sản xuất của Nhóm Phát triển.

Để đảm đương tốt vai trò này, Product Owner cần là người có hiểu biết về tầm nhìn của sản phẩm. Có thể Product Owner chưa biết cụ thể là sẽ làm những gì, nhưng có hiểu biết sâu sắc tại sao lại xây dựng sản phẩm này.

Product Owner là một người duy nhất, không phải là một nhóm người. Điều này không có nghĩa là một mình Product Owner phải làm mọi việc, mà có thể trao đổi, nhận sự hỗ trợ từ những người khác. Tuy nhiên, Product Owner vẫn là người đại diện duy nhất và chịu trách nhiệm về sản phẩm đang xây dựng. Cụ thể, Product Owner là người duy nhất chịu trách nhiệm quản lý Product Backlog - nơi lưu trữ các hạng mục cần phát triển của sản phẩm.

Cách thức quản lý Product Backlog được bàn kĩ trong Chương 5 và một phần trong Chương 7 (về sử dụng User Story và ước tính linh hoạt).

Ngoài việc quản lý Product Backlog, Product Owner còn phải tích cực tham gia vào các cuộc họp của nhóm để cung cấp các thông tin cần thiết, chủ động quản lý một Kế hoạch Phát hành cho sản phẩm.

### **Sáu nhiệm vụ chính của một Product Owner**

1. Công việc đầu tiên của Product Owner là tìm hiểu và phân tích kỹ về sản phẩm dự định phát triển, lên danh sách các tính năng mong muốn bằng việc liệt kê chúng trong một Product Backlog. Đây là danh sách các hạng mục mà Nhóm Phát triển dựa vào để làm việc và chuyển thành các tính năng của sản phẩm thật. Danh sách này không phải là cố định mà được điều chỉnh trong suốt quá trình phát triển của sản phẩm sao cho phù hợp nhất.

2. Các hạng mục trong Product Backlog, sẽ được Product Owner đánh giá giá trị và sắp xếp. Hạng mục nào có giá trị cao hơn sẽ được đưa lên trên để đưa vào sản xuất sớm nhất. Khái niệm 'giá trị' ở đây phụ thuộc vào rất nhiều yếu tố, ví dụ như lợi nhuận thu về, mong muốn của khách hàng, các mục tiêu chiến lược, sự phụ thuộc vào các hạng mục khác, rủi ro và nhiều yếu tố khác nữa. Nếu không có được một giá trị cụ thể bằng con số, nhưng Product Owner có thể so sánh dựa theo giá trị tương đối để thực hiện việc sắp xếp. Điều này cũng tương tự như việc, chúng ta không biết chính xác kích thước của quả bưởi, nhưng biết chắc chắn là nó to hơn quả cam.

3. Nhiệm vụ thứ ba của Product Owner là tối ưu hóa lợi nhuận trên vốn đầu tư (ROI). Trong khi, khả năng sản xuất của Nhóm Phát triển thường có một giới hạn nhất định, Product Owner là người luôn phải tìm cách để sử dụng tốt nhất khả năng này. Vì vậy, ngoài việc sắp xếp các hạng mục trong Product Backlog thì Product Owner cũng cần phải biết nói “không” để loại bỏ những hạng mục không cần thiết.

4. Product Backlog là một tạo tác (artifact) quan trọng. Product Owner phải giữ cho nó luôn được minh bạch và rõ ràng đối với tất cả mọi người. Nói cách khác Product Owner phải đảm bảo tính minh bạch của Product Backlog.

5. Một nhiệm vụ quan trọng nữa của Product Owner là phải luôn luôn sẵn sàng giải thích cho Nhóm Phát triển hiểu rõ các hạng mục của Product Backlog. Điều này đảm bảo Nhóm Phát triển hiểu đúng và đầy đủ về từng hạng mục mà họ triển khai.

6. Nhiệm vụ của Product Owner còn là theo dõi tiến độ phát triển sản phẩm, đảm bảo minh bạch và có thể giải trình với các bên liên quan khi có yêu cầu. Product Owner có thể sử dụng các công cụ như Biểu đồ Burndown, hay các công cụ khác để thực hiện nhiệm vụ này.

Ngoài các nhiệm vụ trên, Product Owner là người có quyền và chịu trách nhiệm khi quyết định hủy Sprint (dừng Sprint bất thường).

## **Danh mục kiểm tra của Product Owner**

### **Kiểm tra hằng ngày**

- ☐ Đã làm rõ yêu cầu sản phẩm cho Nhóm Phát triển (nếu có)
- ☐ Đã làm mịn các hạng mục của Product Backlog, đặc biệt các hạng mục ở trên

### **Kiểm tra theo Sprint**



- Đã làm việc với các bên liên quan để cập nhật những kỳ vọng mới về sản phẩm
- Đã sắp xếp các hạng mục trong Product Backlog đảm bảo tối ưu hóa ROI Đã làm mịn các hạng mục của Product Backlog
- Đã xác định tiêu chí chấp nhận cho các hạng mục sẽ phát triển trong Sprint tới
- Đã tham gia Lập kế hoạch Sprint để trình bày về tổng quan sản phẩm, từng hạng mục sẽ làm trong Sprint tới và giải đáp mọi thắc mắc về yêu cầu trong Lập kế hoạch Sprint
- Đã tham gia và đưa ra phản hồi về sản phẩm trong Sơ kết Sprint

Sự thành công của Product Owner phụ thuộc vào nhiều yếu tố như:

Product Owner cần được trao quyền ra quyết định. Các quyết định này cần phải được tôn trọng. Chúng được thể hiện thông qua nội dung và trật tự của các hạng mục trong một Product Backlog duy nhất. Nhóm Phát triển chỉ làm việc dựa trên Product Backlog này chứ không phải làm theo bất cứ yêu cầu nào khác, cho dù yêu cầu đó xuất phát từ bất cứ ai.

Product Owner phải có hiểu biết về lĩnh vực sản phẩm đang xây dựng. Điều này là cần thiết để giữ đúng hướng đi trong quá trình phát triển, đưa ra các quyết định, lựa chọn, thêm hay loại bỏ các tính năng nhằm giữ cho Nhóm Phát triển luôn làm việc trên những hạng mục cần thiết nhất và sản phẩm luôn mang lại giá trị cao nhất.

Product Owner phải có đủ thời gian cho công việc của mình. Tốt nhất anh ta là người làm việc toàn thời gian cho một sản phẩm duy nhất. Nếu Product Owner phải làm việc trên nhiều sản phẩm thì sẽ dẫn đến tình trạng mất tập trung, làm suy giảm đáng kể hiệu quả công việc.

Product Owner phải có kỹ năng giao tiếp và thương lượng tốt. Quá trình phát triển sản phẩm đòi hỏi Product Owner trao đổi và cộng tác

thường xuyên với Nhóm Phát triển và các bên liên quan, cùng với đó là việc phải thương lượng để đưa ra các quyết định ảnh hưởng đến sản xuất và sản phẩm. Do đó, các kỹ năng này là rất quan trọng để đảm bảo công việc của Product Owner có thể đạt kết quả cao nhất.

⇒ **Dừng và Nghĩ**

**Bạn có thể viết lại mô tả công việc của Product Owner không?**

## **NHÓM PHÁT TRIỂN**

---

Nhóm Phát triển là đội ngũ trực tiếp làm ra sản phẩm, nhóm này bao gồm các chuyên gia có nhiệm vụ chuyển giao phần tăng trưởng có thể chuyển giao được ở cuối mỗi Sprint. Nhóm Phát triển trong Scrum cũng là một nhóm tự tổ chức và liên chức năng. Nhóm được trao quyền để tự định hướng và đưa ra các quyết định liên quan đến công việc sản xuất. Tự tổ chức cũng có nghĩa là nhóm có toàn quyền lựa chọn công cụ, kỹ thuật và cách thức để hoàn thành công việc. Trong quá trình sản xuất, họ tự tiến hành ước lượng, phân bổ, theo dõi, điều tiết công việc theo hình thức tập thể.

Khi bắt đầu mỗi Sprint, Nhóm Phát triển tự lựa chọn các hạng mục từ danh sách đã được sắp xếp của Product Backlog sao cho phù hợp với khả năng sản xuất của mình. Đây là một khác biệt so với một số mô hình phát triển truyền thống - các mô hình hoạt động theo hình thức ra lệnh và kiểm soát. Trong mô hình sản xuất truyền thống các công việc thường được bộ phận quản lý phân chia và gán trách nhiệm cho từng người. Với cách thức làm việc như của Nhóm Phát triển trong Scrum sẽ giúp làm tăng ý thức sở hữu và tính cam kết của các thành viên.



Đối với mỗi hạng mục công việc, Nhóm Phát triển là những người ước lượng kích thước tốt nhất bởi vì họ là lực lượng trực tiếp sản xuất, là những người am hiểu về công việc và các yếu tố liên quan

đến nó. Điều này không chỉ cần thiết đối với nhóm mà còn hữu ích đối với Product Owner khi quản lý Product Backlog.

Nhóm Phát triển thường xuyên theo dõi tiến độ công việc để biết được khả năng của mình, tốc độ sản xuất, tình hình đạt được Mục tiêu Sprint và các mốc quan trọng của sản phẩm. Nhờ theo dõi liên tục và sự minh bạch về thông tin nên Nhóm Scrum nói chung, Nhóm Phát triển nói riêng có những điều chỉnh hợp lý trong tổ chức sản xuất.



Một đặc điểm quan trọng của Nhóm Phát triển đó là tính liên chức năng. Liên chức năng đối với Nhóm Phát triển có nghĩa là nhóm được trang bị đầy đủ các kỹ năng cần thiết để hoàn thành công việc và chuyển giao phần tăng trưởng ở cuối mỗi Sprint mà không cần sự hỗ trợ từ bên ngoài. Đặc điểm này là quan trọng vì nó giúp Nhóm Phát triển có thể hoạt động độc lập, trở thành một đơn vị sản xuất hoàn chỉnh, tổ chức vận hành và chuyển giao sản phẩm có chất lượng. Đặc điểm này giúp Nhóm Phát triển đưa ra các quyết định nhanh chóng, kịp thời mà không phụ thuộc và chờ đợi từ các cá nhân, đơn vị khác bên ngoài.

Việc tổ chức nhóm liên chức năng là rất khác biệt so với cách tổ chức theo phòng/ban chuyên môn truyền thống. Ở những hệ thống có các phòng ban chuyên môn, công việc sản xuất đòi hỏi sự cộng tác, phối hợp giữa những bộ phận khá tách biệt. Điều này làm giảm đáng kể sự linh hoạt, gây lãng phí và làm chậm tốc độ sản xuất. Đối với một nhóm liên chức năng, sự cộng tác và đồng bộ diễn ra nhanh chóng và tối ưu giúp cho năng suất và hiệu quả sản xuất được nâng lên.

Nhóm Phát triển không có sự phân chia chức danh hay vai trò, tất cả các thành viên đều có vai trò giống nhau. Mỗi thành viên thường có thể mạnh ở một vài kỹ năng nhất định và những kỹ năng đó góp lại trở thành kỹ năng chung của nhóm chứ không còn là của riêng cá nhân nào. Như vậy, chúng ta sẽ không thấy những chức danh như “nhà phân tích nghiệp vụ (BA)”, “nhà thiết kế (design-er)”, “lập trình

viên (coder)”, “kiểm thử viên (tester)” trong Nhóm Phát triển. Thay vào đó, tất cả các thành viên đều được gọi chung là “nhà phát triển (developer)” và kỹ năng tổng thể của nhóm là phân tích nghiệp vụ, thiết kế, lập trình, kiểm thử. Việc xóa bỏ sự phân chia vai trò giúp làm tăng tính sở hữu tập thể, trách nhiệm tập thể và bình đẳng trong công việc. Đồng thời cũng giúp thúc đẩy việc học tập trong Nhóm Phát triển.



Các thành viên không chỉ tham gia giải quyết những hạng mục công việc mà mình có thể mạnh nhất, mà còn giúp sức trong những công việc khác. Điều này đòi hỏi sự trao đổi, học tập và rèn luyện những kỹ năng khác diễn ra giữa các thành viên. Theo thời gian mỗi thành viên sẽ học được những kỹ năng mới ngoài những kỹ năng vốn có của mình. Ví dụ, một thành viên với kỹ năng chính là thiết kế thì có thể có thêm kỹ năng mới là kiểm thử, một thành viên với kỹ năng chính là phân tích nghiệp vụ thì có thể có thêm kỹ năng mới là lập trình.

### **Nhóm Phát triển gồm bao nhiêu thành viên là đủ?**

Nhóm Phát triển vận hành tốt nhất với số lượng thành viên vừa phải, quy định của Scrum là từ 3 đến 9 người. Nhóm quá ít thành viên sẽ dẫn đến khả năng sản xuất nhỏ và có thể thiếu các kỹ năng cần thiết (không liên chức năng), khó để tạo được các phần tăng trưởng hoàn chỉnh đáng kể ở cuối mỗi Sprint. Nhóm có quá nhiều thành viên sẽ làm tăng tính phức tạp trong quản lý, gây khó khăn trong tương tác, giao tiếp, đảm bảo minh bạch và làm giảm tính linh hoạt của nhóm. Nếu dự án lớn cần nhiều người, chúng ta cần chia nhỏ nhóm và sử dụng kỹ thuật Scrum ở quy mô lớn (tham khảo Chương 9).

### **Có cần sự ổn định không?**

Nhóm Phát triển đòi hỏi tính ổn định cao để đạt được năng suất và chất lượng cao nhất trong sản xuất. Điều này sẽ thuận lợi cho các thành viên trong cộng tác, giao tiếp, tận dụng tối ưu sự đóng góp

của mỗi người. Sự ổn định cũng giúp nhóm điều chỉnh các thói quen hoạt động của mình một cách phù hợp và hiệu quả nhất. Giữ cho nhóm ổn định cũng giúp khả năng tiên lượng khi lập kế hoạch tốt hơn do các yếu tố như hiệu suất và tốc độ sản xuất của nhóm ổn định.

Tuy vậy, trong thực tế việc thay đổi thành viên đôi khi là khó tránh khỏi do nhiều nguyên nhân khác nhau. Khi này, nhóm biết rằng việc đó sẽ ảnh hưởng tới năng suất trong ngắn hạn.

### **Nhóm Phát triển đóng góp những gì cho sản phẩm?**

Ngoài việc trực tiếp làm ra sản phẩm, sự am hiểu của Nhóm Phát triển về sản phẩm là rất quan trọng để đi đến thành công. Đặc biệt là đối với các sản phẩm mới. Ở đó, sản phẩm là cả một quá trình tìm tòi, thử nghiệm, điều chỉnh mà ngay cả Product Owner và các bên liên quan đều không biết rõ ràng từ trước.

Nhóm Phát triển còn giúp sức cho Product Owner trong việc duy trì Product Backlog. Có những công việc mà Nhóm Phát triển là những người làm tốt nhất, chẳng hạn như việc ước tính kích thước của các hạng mục, các yêu cầu liên quan đến công nghệ, sự phụ thuộc giữa các hạng mục, v.v.. Đây là những yếu tố có ảnh hưởng lớn đến nội dung của Product Backlog.

img479

### **LƯU Ý!**

Nhóm Phát triển hoạt động hướng vào mục tiêu chung, thay vì hướng vào công việc của từng cá nhân. Và tuân thủ nguyên tắc giới hạn lượng việc đang làm (Limit WIP - Limit Work-In- Progress).

30

### **CÂU HỎI ỨNG DỤNG**

1. ScrumMaster có phải là người quản lí dự án hay không? Tại sao nói ScrumMaster là người lãnh đạo kiểu mới?
2. Cần có những tố chất gì để trở thành một ScrumMaster tốt?
3. Để làm tốt vai trò Product Owner bạn cần những kĩ năng nào?
4. Công việc chủ đạo của Product Owner với Product Backlog là gì?
5. ScrumMaster làm gì để giúp Nhóm Phát triển đạt được năng suất cao?
6. Tính liên chức năng và tự tổ chức giúp ích gì cho Nhóm Phát triển?
7. Nhóm Scrum sử dụng những công cụ chủ đạo nào để tăng cường sự minh bạch?
8. Nhóm của bạn có đủ tính chất để trở thành một nhóm Scrum không? Cần làm gì để có thể bù đắp những tính chất còn thiếu (nếu có)?
9. Nếu áp dụng Scrum cho một nhóm của bạn, ai sẽ đóng vai trò ScrumMaster?
10. Quy tắc làm việc nhóm của bạn có những mục nào là quan trọng nhất?

# 4 CÁC SỰ KIỆN SCRUM

Trong chương này...

- Sprint
- Lập kế hoạch Sprint
- Scrum Hằng ngày
- Sơ kết Sprint
- Cải tiến Sprint

*Bất cứ ai ngừng học hỏi đều trở nên già cỗi, cho dù ở tuổi hai mươi hay đã tám mươi tuổi. Bất cứ ai không ngừng học hỏi vẫn mãi trẻ trung. Điều tuyệt vời nhất trong cuộc sống chính là giữ cho trí tuệ của bạn luôn tươi trẻ.*

## Henry Ford

Các sự kiện trong Scrum hình thành một thứ tự lặp đi lặp lại các công việc có chủ đích nhằm xây dựng một thói quen làm việc hiệu quả. Không gọi là các cuộc họp (dù chúng là các cuộc họp), các Sự kiện Scrum là cơ hội để cộng tác nhóm. Tất cả các Sự kiện Scrum đều được đóng khung thời gian (time-boxed), nghĩa là mỗi sự kiện có giới hạn thời gian tối đa. Điều này đảm bảo thời lượng vừa đủ để tránh lãng phí thời gian không cần thiết cho sự kiện.

Bao gồm cả bản thân Sprint vốn chứa tất cả các sự kiện khác, mỗi sự kiện trong Scrum là một cơ hội chính thức để thực hiện cơ chế thanh tra và thích nghi. Mỗi sự kiện đều có quy cách rõ ràng. Nếu không thực hiện được hoặc thực hiện không đúng các sự kiện này có thể dẫn đến giảm sự minh bạch và đánh mất cơ hội để thanh tra và thích nghi.

Các sự kiện trong Scrum bao gồm:

1. Sprint
2. Lập kế hoạch Sprint
3. Scrum Hằng ngày
4. Sơ kết Sprint
5. Cải tiến Sprint



### **Bảng tóm tắt các Sự kiện Scrum**



Sprint là trái tim của Scrum. Sự kiện này có khung thời gian (time box) không vượt quá 1 tháng. Thông thường các Sprint có thể kéo dài từ 1 đến 4 tuần. Các nhóm phải quyết định độ dài ổn định cho nhóm của mình dựa trên điều kiện thực tế.

Các phần tăng trưởng của sản phẩm có thể phát hành được được sản xuất gói gọn trong khung thời gian này. Trong suốt quá trình phát triển một sản phẩm, Sprint được khuyến nghị giữ khung thời gian cố định. Một Sprint mới bắt đầu ngay khi Sprint trước kết thúc.



Sprint chứa các sự kiện còn lại của Scrum. Nói cách khác, các sự kiện Lập kế hoạch Sprint, Scrum Hằng ngày, Sơ kết Sprint, Cải tiến Sprint diễn ra trong khung thời gian của Sprint.

### **Chúng ta cần lưu ý, trong suốt Sprint:**

- Không cho phép có bất kì sự thay đổi nào ảnh hưởng đến Mục tiêu Sprint;



- Thành phần Nhóm Phát triển được giữ nguyên;
- Mục tiêu chất lượng không được cắt giảm; và,
- Phạm vi có thể được làm rõ và tái thương lượng giữa Product Owner và Nhóm Phát triển.

Mỗi Sprint có thể được coi như một tiểu dự án với độ dài tối đa là 1 tháng. Mỗi Sprint có một mục tiêu rõ ràng xác định việc phải xây dựng trong Sprint, một bản thiết kế và bản kế hoạch linh hoạt sẽ hướng dẫn quá trình xây dựng đó.

### **Hủy một Sprint:**

Sprint có thể bị hủy trước khi kết thúc khung thời gian. Chỉ có Product Owner mới đủ thẩm quyền kết thúc Sprint, mặc dù Product Owner có thể chịu ảnh hưởng bởi những bên hữu quan khác, bởi Nhóm Phát triển hoặc bởi ScrumMaster.

Một Sprint có thể bị hủy nếu như Mục tiêu Sprint không còn phù hợp nữa. Điều này xảy ra khi công ty chuyển hướng kinh doanh hoặc khi tình thế công nghệ có sự thay đổi. Nhìn chung, Sprint có thể bị hủy nếu nó không mang lại điều gì có ích. Thế nhưng, do thời gian mỗi Sprint tương đối ngắn, nên việc hủy một Sprint không mấy khi xảy ra.

Khi Sprint bị hủy, các phần sản phẩm đã hoàn chỉnh được xem xét lại. Nếu phần nào đó của công việc đó là có thể chuyển giao được, thì Product Owner có thể chấp nhận chúng. Các hạng mục Product Backlog chưa hoàn tất sẽ được ước lượng lại và trả về Product Backlog để phát triển tiếp. Các phần việc đã thực hiện trên đó sẽ nhanh chóng hết tác dụng và phải thường xuyên được ước lượng lại.

Việc hủy Sprint sẽ gây lãng phí tài nguyên, do mọi người phải mất thời gian, công sức để lên kế hoạch cho một Sprint mới. Việc hủy Sprint thường gây tổn hại nhất định cho Nhóm Phát triển, và rất ít khi xảy ra.

## LẬP KẾ HOẠCH SPRINT

---

Một Sprint bắt đầu bằng buổi Lập kế hoạch Sprint để xác định Mục tiêu Sprint và lên kế hoạch các công việc cần thực hiện. Sự kiện này được chia làm hai phần: Phần thứ nhất để lựa chọn các công việc cần làm trong Sprint và Phần thứ hai để quyết định cách thức hoàn thành các công việc đã lựa chọn trước đó.

Toàn bộ buổi Lập kế hoạch Sprint sẽ lần lượt trả lời các câu hỏi:

- Mục tiêu của Sprint này là gì?
- Sprint này phải chuyển giao cái gì?
- Làm sao để đạt được điều đó?

Toàn bộ Nhóm Scrum bắt buộc phải tham gia phần thứ nhất của sự kiện. Ở phần thứ hai Product Owner có thể vắng mặt nhưng phải luôn sẵn sàng để hỗ trợ Nhóm Phát triển làm rõ các hạng mục khi cần thiết, chẳng hạn thông qua điện thoại hay các hệ thống trao đổi trực tuyến.

Buổi Lập kế hoạch Sprint kéo dài trong 8 giờ đối với Sprint 1 tháng, đối với các Sprint ngắn hơn thì khung thời gian cho sự kiện có thể ngắn hơn. Ví dụ, đối với Sprint 2 tuần khung thời gian cho buổi Lập kế hoạch là khoảng 4 giờ. Thời gian dành cho hai phần của sự kiện này là bằng nhau, mỗi phần chiếm một nửa khung thời gian.

### Phần 1: Làm gì trong Sprint này?

Kết thúc Phần 1 của buổi Lập kế hoạch Sprint; Nhóm Scrum sẽ đưa ra câu trả lời cho câu hỏi: Chúng ta sẽ làm gì trong Sprint này? Cụ thể kết quả của phần này là Mục tiêu Sprint và danh sách các hạng mục Product Backlog được lựa chọn để phát triển trong Sprint.

Phần 1 được bắt đầu bằng việc Product Owner trình bày mục tiêu mong muốn đạt được trong Sprint này. Sau đó, Product Owner làm rõ thêm các hạng mục ở phần trên của Product Backlog (những hạng mục có độ ưu tiên cao nhất) để cả nhóm hiểu kỹ về hơn các hạng mục này. Product Owner cần làm rõ số lượng hạng mục Product Backlog nhiều hơn số lượng mà Nhóm Phát triển có thể hoàn thành trong cho một Sprint (con số này dựa vào các Sprint trước đó hoặc dựa trên kinh nghiệm). Ví dụ, nếu ước lượng thấy nhóm có khả năng làm khoảng 5 hạng mục thì ít nhất cả nhóm phải hiểu rõ khoảng 8 hạng mục hoặc hơn thế. Việc đó giúp nhóm có đầy đủ thông tin để đưa ra các quyết định lựa chọn một cách chính xác cho Sprint này.

Thông thường, những hạng mục này đã được phân tích kỹ và làm rõ từ một vài Sprint trước thông qua việc làm mịn Product Backlog, do đó công việc trong phần này không tốn quá nhiều thời gian. Sau đó, cả nhóm sẽ hợp tác để tìm hiểu công việc phải làm trong Sprint.

Căn cứ vào Mục tiêu Sprint và năng lực hiện có, Nhóm Phát triển lựa chọn những hạng mục mà họ tin là có thể hoàn thành trong Sprint này, bắt đầu từ những hạng mục đứng đầu trong Product Backlog - nói cách khác, họ bắt đầu với những hạng mục có độ ưu tiên cao nhất do Product Owner sắp xếp. Đây là cách làm chủ chốt trong Scrum: Nhóm Phát triển quyết định có bao nhiêu hạng mục sẽ được hoàn thành, thay vì được giao bởi Product Owner. Điều này sẽ giúp Nhóm Phát triển đưa ra các dự báo tin cậy hơn do họ làm việc đó căn cứ vào những phân tích và lập kế hoạch của chính bản thân họ.

## **Năng lực của nhóm**

Bạn cần phải tính được nhóm có bao nhiêu người làm việc bao nhiêu giờ trong Sprint này. Vì có thể có ai đó sẽ nghỉ hoặc phải giảm thời lượng làm việc để phục vụ cho nhiệm vụ khác. Ngoài ra, bạn cũng cần phải tính thời gian “không năng suất” của mỗi ngày. Bởi thực chất một người chỉ có thể tập trung làm việc trong một thời gian nhất định trong ngày. Nếu thời gian làm việc mỗi ngày là 8 tiếng, thì thời gian “năng suất” – tức thời gian mà mỗi thành viên có

thể dành để thực sự làm việc trên các hạng mục Product Backlog – có thể chỉ là 5 hoặc 6 giờ.

Để tránh được những ước lượng quá lạc quan (cũng có nghĩa là khả năng lớn sẽ phải làm thêm ngoài giờ hoặc không thể hoàn thành được mục tiêu đã định), bạn sẽ phải hết sức thực tế trong việc tính toán năng lực của cả nhóm trong Sprint.

Để quyết định những hạng mục nào của Product Backlog sẽ triển khai trong Sprint này, Nhóm Phát triển sử dụng tốc độ sản xuất trung bình của mình trong quá khứ đồng thời tính khả năng của nhóm trong Sprint hiện tại.

Ví dụ, vận tốc trung bình của Nhóm Phát triển 7 người là 80 point trong một Sprint (nhóm này ước tính các hạng mục Product Backlog theo point - điểm). Trong Sprint hiện tại, có một thành viên xin nghỉ phép 3 ngày. Vậy, khả năng của nhóm trong Sprint này chỉ đạt được 74 point. Do đó, nhóm chỉ lựa chọn số lượng hạng mục đủ để làm trong khả năng 74 point của mình.

Nếu Nhóm Phát triển muốn lựa chọn một vài hạng mục có độ ưu tiên thấp ở phía dưới của Product Backlog họ cần thảo luận với Product Owner. Điều này thường xảy ra khi có sự phụ thuộc giữa các hạng mục hoặc nhóm cảm thấy hạng mục có độ ưu tiên thấp đó rất phù hợp để phát triển cùng với các hạng mục khác đã lựa chọn.

Kết thúc Phần 1, Nhóm Phát triển và Product Owner thống nhất lại Mục tiêu Sprint và danh sách các hạng mục sẽ chuyển giao trong Sprint này. Mục tiêu Sprint là một đoạn mô tả ngắn về kết quả kỳ vọng đạt được sau khi Sprint kết thúc. Mục tiêu Sprint đóng vai trò định hướng Nhóm Phát triển trong suốt quá trình diễn ra Sprint và đồng thời giúp nhóm đưa ra được những quyết định hợp lý nhằm đạt được mục tiêu này.

Một ví dụ về Mục tiêu Sprint là: “Xây dựng chức năng mua hàng bao gồm: Xem danh sách sản phẩm, thêm sản phẩm vào giỏ hàng, xem giỏ hàng, loại bỏ sản phẩm khỏi giỏ hàng, hiển thị trang thanh toán.”



### **Ví dụ: Lịch trình họp Lập kế hoạch Sprint cho một Sprint 1 tuần**

- 8:00 – 8:30. Phiên họp bắt đầu. Product Owner trình bày mục tiêu của Sprint và tóm tắt các hạng mục Product Backlog dự kiến.
- 8:30 – 9:00. Nhóm Phát triển ước tính thời gian và tìm hiểu rõ các việc cần phải làm. Nhóm Phát triển và Product Owner cùng trao đổi về những điều nghi vấn về hạng mục Product Backlog (nếu có).
- 9:00 – 9:30. Nhóm chọn hạng mục để đưa vào Sprint. Thực hiện tính toán tốc độ để đảm bảo tính khả thi của Mục tiêu Sprint.
- 9:30 – 11:00. Chọn thời gian và địa điểm để họp Scrum Hằng ngày (nếu có thay đổi so với Sprint trước). Chia nhỏ hơn các hạng mục thành các đầu việc phải làm. Trong lúc thảo luận về chia nhỏ cách làm, nhóm có thể sẽ phải vẽ ra những phác thảo về thiết kế hệ thống, giao diện, và những quy ước hợp tác nhất định trong viết mã lệnh.
- 11:00-11:15. Cả nhóm chốt lại kế hoạch của Sprint. Cập nhật các công việc lên Sprint Backlog và chuẩn bị vào Sprint làm việc. Phiên họp kết thúc.

### **Tốc độ (Velocity)**

Tốc độ được tính bằng số lượng đơn vị được hoàn thành trong mỗi Sprint. Nếu bạn dùng đơn vị là điểm (point), thì tốc độ chính là số điểm mà nhóm hoàn thành được trong một Sprint. Qua thời gian, tốc độ của nhóm có thể sẽ tương đối ổn định. Đó là tiền đề quan trọng để nhóm có thể phỏng đoán khối lượng công việc của nhóm trong mỗi Sprint.

### **⇒ Dừng và Nghĩ**

**Vì sao việc đặt mục tiêu Sprint lại quan trọng? Nói rộng ra, tại sao việc đặt mục tiêu lại quan trọng?**

## **Đề đặt mục tiêu và lập kế hoạch hiệu quả cho Sprint, Nhóm Scrum có thể sử dụng SMART Rubric như là công cụ hướng dẫn**



### **Phần 2: Làm sao để hoàn thành công việc đã chọn?**

Phần 2 của buổi Lập kế hoạch Sprint với mục đích trả lời cho câu hỏi: Làm sao để hoàn thành công việc đã chọn? Kết quả của phần này đó là Sprint Backlog - tức là bảng công việc được Nhóm Phát triển sử dụng trong suốt Sprint, bao gồm các hạng mục Product Backlog đã lựa chọn và danh sách công việc tương ứng với từng hạng mục Product Backlog đó.

Nhóm Phát triển bắt đầu thiết kế hệ thống và lên danh sách các công việc cần làm. Đối với mỗi hạng mục trong danh sách đã lựa chọn, nhóm sẽ tách thành các công việc cụ thể. Các công việc này thường đủ nhỏ để hoàn thành trong vòng một ngày hoặc ít hơn. Một số loại công việc thường thấy như: thiết kế, viết kiểm thử, viết mã, viết tài liệu, nghiên cứu kỹ thuật hoặc công nghệ mới, v.v.. Tất cả các công việc này đều phải được Nhóm Phát triển ước lượng nỗ lực để hoàn thành từng công việc. Thường các nỗ lực này được lượng hóa theo đơn vị giờ hoặc point và chúng được Nhóm Phát triển điều chỉnh thường xuyên trong quá trình triển khai. Một trong các kỹ thuật thường dùng để ước lượng là Planning Poker (xem Chương 7).

Tổng nỗ lực cho tất cả các hạng mục trên Sprint Backlog được nhóm sử dụng để theo dõi tiến độ Sprint. Giá trị này được cập nhật ngay vào Sprint Backlog và đồng thời cũng được sử dụng để tạo Biểu đồ Sprint Burndown nhằm theo dõi tiến độ Sprint. Sau mỗi ngày làm việc, các thành viên sẽ cập nhật đồng thời Sprint Backlog và Biểu đồ Burndown với các giá trị mới.

Nếu Nhóm Phát triển thấy rằng lượng công việc vừa lựa chọn là quá nhiều hay quá ít so với khả năng của nhóm họ có thể trao đổi và thương lượng với Product Owner để loại bỏ bớt hoặc chọn thêm các hạng mục khác. Trong Phần 2, Nhóm Phát triển có thể cần thêm

sự hỗ trợ về mặt kĩ thuật từ bên ngoài nhóm để việc phân tích, ước lượng công việc tốt hơn.

## **Lập kế hoạch phát hành**

Có thể bạn sẽ cần một kế hoạch dài hơi hơn để cho ra được một bản phát hành có ý nghĩa lớn nào đó. Lúc này bạn cần phải duy trì một Kế hoạch Phát hành (Release Plan). Kế hoạch này sẽ phải dựa nhiều vào việc ước tính kích thước các hạng mục Product Backlog. Hãy xem thêm kĩ thuật ước tính linh hoạt được đề cập đến trong chương 7.

Khi công việc ước tính cho từng hạng mục đã được hoàn tất, bạn có thể căn cứ vào tốc độ của nhóm mà dự đoán là đến khi nào thì có thể có được một bản phát hành có ý nghĩa.

Để cho kế hoạch không quá cứng nhắc, có thể bạn sẽ cần phân loại các tính năng Phải có, Nên có, và Có thể có khi lựa chọn các tính năng sẽ hiện diện trong bản phát hành sắp tới. Tập hợp những phần Phải có có thể tạo thành một bộ tính năng thuộc dạng MVP (Minimum Viable Product – sản phẩm hữu dụng tối thiểu) có thể chuyển giao giá trị cơ bản tới người dùng cuối.

Nên nhớ rằng tốc độ ước tính là con số dự báo, và trong quá trình hướng đến mục tiêu phát hành, nhóm của bạn có thể gặp phải những vấn đề không tính trước. Vì thế, kế hoạch phát hành cũng phải được hiệu chỉnh thường xuyên sau mỗi Sprint. Bạn cũng có thể dùng Biểu đồ Release Burndown (xem Chương 5) để theo dõi tiến độ đạt được mục tiêu phát hành qua các Sprint.



## **Danh mục kiểm tra Lập kế hoạch Sprint**

- ☐ Buổi lập kế hoạch có diễn ra đúng giờ?
- ☐ Có thành viên nào thiếu không? Lý do vì sao?

- Product Backlog có sẵn sàng trước buổi lập kế hoạch?
- Product Owner có sơ kết lại tình hình sản phẩm hiện tại đầu buổi lập kế hoạch?
- Product Owner có đưa ra mong muốn mục tiêu Sprint đầu buổi lập kế hoạch?
- Khả năng của Nhóm Phát triển đã được tính?
- Có những vấn đề, sự kiện nào đặc biệt ảnh hưởng tới khả năng của nhóm không (ví dụ nghỉ hè, thành viên nào có việc riêng)?
- Nhóm Phát triển và Product Owner có rà soát những hạng mục có thể sẽ phát triển trong Sprint?
- Những hạng mục có thể sẽ phát triển trong Sprint đã có tiêu chí chấp nhận?
- Nhóm Phát triển đã phân ra các hạng mục sẽ phát triển thành các công việc đưa vào Sprint Backlog?
- Các hạng mục trong Sprint Backlog đã được ước lượng? Product Owner có hài lòng với cam kết của Nhóm Phát triển?
- Nhóm Phát triển có tin vào khả năng hoàn thành cam kết? Có điều gì cần lưu ý trong Sprint?
- Buổi Lập kế hoạch Sprint có đảm bảo khung thời gian?
- Đã cập nhật các tạo tác như Sprint Backlog, Biểu đồ Burndown?

## **SCRUM HẰNG NGÀY**

---



Scrum Hằng ngày là một nghi thức quan trọng diễn ra đều đặn hằng ngày trong suốt quá trình sản xuất. Đây là một buổi trao đổi ngắn



không kéo dài quá 15 phút với mục đích giúp Nhóm Phát triển đồng bộ công việc và lập kế hoạch cho ngày làm việc tiếp theo. Đây là một sự kiện dành cho Nhóm Phát triển, tất cả các thành viên của Nhóm Phát triển đều phải tham dự. Scrum-Master thường tham dự nhưng chỉ để đảm bảo Nhóm Phát triển thực hiện đúng công việc này chứ không trực tiếp tham gia điều hành hay phát biểu đóng góp vào nội dung của sự kiện. Product Owner và một số người khác có thể tham dự nhưng chỉ được lắng nghe và không đóng bất cứ vai trò nào trong sự kiện này.

Buổi Scrum Hằng ngày cần được diễn ra tại một địa điểm và khung thời gian cố định để giảm thiểu sự phức tạp. Thời điểm tổ chức được Nhóm Phát triển lựa chọn sao cho thuận lợi nhất đối với tất cả các thành viên. Thường thì các nhóm lựa chọn thời điểm Scrum Hằng ngày là đầu buổi sáng để khởi động cho một ngày làm việc mới. Việc lựa chọn thời điểm tùy thuộc vào nhóm, điều quan trọng là đảm bảo sự kiện này luôn luôn diễn ra đúng thời điểm đã lựa chọn nhằm tạo ra thói quen và không biến nó thành một sự kiện phức tạp.

Nội dung buổi Scrum Hằng ngày chỉ gói trong khoảng thời gian 15 phút để các thành viên Nhóm Phát triển trả lời 3 câu hỏi:

- Tôi đã làm được những gì từ buổi Scrum Hằng ngày trước cho tới bây giờ?
- Tôi sẽ làm những gì từ bây giờ tới hôm sau?
- Tôi đang gặp những khó khăn gì?

Có một vấn đề đối với các Nhóm Phát triển khi thực hiện Scrum Hằng ngày đó là thảo luận sâu trong suốt buổi Scrum Hằng ngày, đặc biệt khi phát hiện ra vấn đề nào đó. Hãy đảm bảo mỗi thành viên chỉ trả lời ba câu hỏi trên, việc trao đổi giải quyết các vấn đề có thể thực hiện ngay sau khi Scrum Hằng ngày kết thúc. Việc này giúp

đảm bảo thời lượng của buổi trao đổi không kéo dài quá 15 phút đồng thời giữ tập trung vào việc đồng bộ và phối hợp công việc giữa

các thành viên thay vì giải quyết vấn đề.

Scrum Hằng ngày được khuyến nghị người tham gia đứng thay vì ngồi để tăng sự tập trung. Đó là lý do sự kiện này còn được gọi là Standup Meeting.

## **Nhiệm vụ của ScrumMaster trong Scrum Hằng ngày**



Nhiệm vụ của ScrumMaster trong buổi Scrum Hằng ngày là đảm bảo điều kiện thuận lợi cho Nhóm Phát triển tiến hành sự kiện, ví dụ như địa điểm, đảm bảo nhóm giữ đúng khung thời gian tối đa 15 phút, ghi nhận các trở ngại được nhóm phát lộ, đảm bảo các thành viên tập trung vào đồng bộ và phối hợp công việc hiện tại thay vì sa đà vào các vấn đề khác.

Ngoài ra, ScrumMaster quan sát trong suốt cuộc họp để cảm nhận không khí và những vấn đề tiềm ẩn. Nếu các vấn đề được phát biểu, ScrumMaster ghi lại và có thể quản lý trong một Danh sách Trở ngại (Impediment Backlog) và giải quyết dần.

Để tháo gỡ khó khăn, đôi khi ScrumMaster phải vận dụng kỹ thuật hỏi 5 lần tại sao (5WHYs).

### **5WHYS**

Đây là kỹ thuật có nguồn gốc từ phương thức sản xuất Toyota đã được dùng phổ biến trong hầu hết các lĩnh vực. Để điều tra nguyên nhân và các lựa chọn để giải quyết vấn đề, người điều tra sẽ hỏi 5 lần liên tiếp tại sao.

Dưới đây là ví dụ về một cuộc điều tra nguyên nhân của việc trả lại hàng:

Vấn đề: “Khách hàng phàn nàn về chất lượng lô hàng vừa giao”

Tại sao 1: Tại sao chất lượng lô hàng kém?

Trả lời: Do nguyên vật liệu đầu vào kém chất lượng.

Tại sao 2: Tại sao chất lượng nguyên vật liệu kém?

Trả lời: Do trong quá trình sản xuất bị thiếu nguyên vật liệu nên phải mua bổ sung từ nguồn khác.

Tại sao 3: Tại sao lại thiếu nguyên vật liệu?

Trả lời: Do ước tính không chính xác nên mua thiếu.

Tại sao 4: Tại sao lại ước tính nguyên vật liệu chưa chính xác?

Trả lời: Do không biết yêu cầu nguyên vật liệu của bản thiết kế mới.

Tại sao 5: Tại sao lại không biết yêu cầu nguyên vật liệu của thiết kế mới?

Trả lời: Tại vì chưa làm thử trước đó.

Thay vì quay ra đổ lỗi cho khách hàng ngay từ đầu, hoặc đổ lỗi cho đồng nghiệp, hoặc đổ lỗi cho bên giao hàng, chúng ta có thể nhìn ra vấn đề gốc rễ nằm ở quy trình sản xuất để khắc phục và tránh lặp lại.

Một ngộ nhận thường thấy là dùng 5WHYs luôn cho ra nguyên nhân gốc rễ (root cause), kì thực 5WHYs chỉ giúp ta tìm ra được những khả năng. Và từ đó ta có thể ra quyết định chính xác hơn.

## **Những lỗi nên tránh khi Scrum Hằng ngày**

#1. Scrum Hằng ngày trở thành là một buổi báo cáo công việc cho ScrumMaster

#2. Scrum Hằng biến thành nơi dành cho ScrumMaster cập nhật tiến độ công việc

#3. Scrum Hằng ngày biến thành một phiên thảo luận để giải quyết vấn đề

#4. Scrum Hằng ngày diễn ra quá xa nơi làm việc

#5. Nhóm Phát triển không thấy giá trị của buổi Scrum Hằng ngày

#6. Các thành viên thường báo cáo là không gặp vấn đề gì

#7. Scrum Hằng ngày không diễn ra như là một thói quen, hôm làm hôm bỏ

## **SƠ KẾT SPRINT**

---



Buổi Sơ kết Sprint sẽ được tiến hành khi thời gian triển khai Sprint đã hết để kiểm tra phần tăng trưởng đạt được trong Sprint vừa qua. Đây là một hoạt động thanh tra và thích nghi đối với sản phẩm đang được xây dựng. Kết thúc buổi Sơ kết Sprint lộ trình sản phẩm và Product Backlog có thể được điều chỉnh để phù hợp hơn với tình hình phát triển mới.

Tham dự buổi Sơ kết Sprint có Nhóm Phát triển, Product Owner, ScrumMaster. Ngoài ra, Product Owner có thể mời thêm những người khác nếu phù hợp, chẳng hạn như khách hàng, người dùng, các bên liên quan, các lãnh đạo, và bất cứ ai quan tâm. Tất cả những người tham dự đều hoàn toàn tự do trong việc đưa ra các câu hỏi và đóng góp ý kiến của mình.

Khung thời gian của buổi Sơ kết Sprint là một giờ tương ứng với một tuần làm việc của Sprint. Có nghĩa là, đối với Sprint 4 tuần thì thời gian tối đa là 4 giờ, với Sprint 2 tuần là 2 giờ. Bạn cũng có thể điều chỉnh thời lượng họp theo nhu cầu thực tiễn của nhóm.

Một nội dung quan trọng của buổi Sơ kết Sprint là Nhóm Phát triển và Product Owner trao đổi với nhau để tìm hiểu về tình hình, ghi nhận các khuyến nghị của nhau. Đây là cơ hội để Product Owner tìm hiểu và nắm tình hình của sản phẩm và của Nhóm Phát triển.

Còn đối với Nhóm Phát triển đây là cơ hội để tìm hiểu và nắm tình hình của Product Owner và của thị trường.

Nội dung của Sơ kết Sprint như sau:

1. Product Owner trình bày Mục tiêu Sprint
2. Nhóm Phát triển trình bày kết quả đã hoàn thành
3. Trực tiếp sử dụng sản phẩm



*Tiến trình một cuộc họp Sơ kết Sprint*

**Một số lưu ý:**

1. Không trình bày những tính năng chưa “hoàn thành”
2. Các phản hồi được đưa ra – Product Backlog có thể được đánh giá lại độ ưu tiên
3. Product Owner nên sử dụng kĩ thuật kiểm thử chấp nhận để đánh giá các tính năng.
4. Đây không là buổi demo sản phẩm. Trong thực tế, việc demo sản phẩm chỉ là một nội dung của buổi Sơ kết Sprint. Nếu chỉ tập trung vào demo sản phẩm thì sẽ bỏ qua một nội dung quan trọng khác liên quan đến việc thảo luận và cộng tác giữa các thành viên tham gia.

Từ đó gây hiểu nhầm và thực hiện sai mục đích thực sự của buổi Sơ kết Sprint đó là thanh tra và thích nghi sản phẩm đang được xây dựng.

**Danh mục kiểm tra Sơ kết Sprint**

- ☐ Buổi Sơ kết Sprint có thiếu thành viên nào trong Nhóm Scrum?
- Buổi Sơ kết Sprint có diễn ra đúng thời gian?

- Nhóm Scrum có trình bày Mục tiêu Sprint?
- Nhóm Phát triển chỉ trình bày những hạng mục đã hoàn thành? Product Owner và các bên liên quan có kiểm tra sản phẩm?
- Product Owner có đưa ra quyết định chấp nhận hoặc không chấp nhận phần tăng trưởng?
- Product Owner và các bên liên quan có đưa ra phản hồi về sản phẩm? Buổi Sơ kết Sprint có giữ đúng khung thời gian?

## CẢI TIẾN SPRINT

---



Cải tiến Sprint là sự kiện diễn ra ở cuối Sprint, ngay sau buổi Sơ kết Sprint. Mục đích của sự kiện này là để thanh tra và thích nghi quy trình làm việc. Nói cách khác là để tìm ra cách làm tốt hơn trong Sprint tới.

Nhóm Phát triển và ScrumMaster bắt buộc phải tham gia buổi Cải tiến Sprint. Product Owner có thể tham gia hoặc không. ScrumMaster cũng có thể mời những người khác cùng tham gia sự kiện này nhằm đóng góp ý kiến cho nhóm.

Thời gian tối đa dành cho buổi Cải tiến Sprint là 3 giờ đối với Sprint 1 tháng. Đối với các Sprint ngắn hơn thì thời gian thường ngắn hơn, khoảng 45 phút tương ứng với một tuần làm việc của Sprint. Chẳng hạn, với Sprint 2 tuần thì khung thời gian là khoảng 1 giờ 30 phút.

### ⇒ Dừng và Nghĩ

**Giả sử sau mỗi Sprint, nhóm của bạn cải thiện năng suất 3%. Hãy tính xem, cả năm nhóm bạn có thể tăng năng suất được bao nhiêu?**

Dừng và nhìn lại, tìm kiếm các cải tiến và xây dựng tổ chức học tập.

### *Cây cầu Cải tiến, theo Rachel Davies*

Cần có một người giữ vai trò hỗ trợ cho buổi Cải tiến Sprint, người này có thể là ScrumMaster hoặc một người khác bên ngoài nhóm. Hoặc có một cách làm khá phổ biến đó là trao đổi chéo giữa các ScrumMaster, có nghĩa là ScrumMaster của nhóm này hỗ trợ Cải tiến Sprint cho nhóm khác.

Các hoạt động chính của buổi Cải tiến Sprint là:

- Liệt kê những điểm đã làm tốt
- Liệt kê những điểm đã làm chưa tốt
- Đưa ra một vài hành động cải tiến
- Kế hoạch cải tiến cho Sprint sau

Cách đơn giản nhất là cả nhóm cùng họp và bắt đầu với câu hỏi “Chúng ta đã làm tốt những gì?”, rồi trả lời câu hỏi tiếp theo “Có điều gì cần loại bỏ?” và “Có điều gì có thể làm tốt hơn? Bằng cách nào?”. Rồi sau đó cùng thống nhất một kế hoạch hành động cho Sprint sau. Có rất nhiều kỹ thuật cụ thể để tiến hành buổi Cải tiến

Sprint. Ví dụ như Glad-Sad-Mad, SpeedBoat, v.v.. Trong đó, kỹ thuật đơn giản và được sử dụng phổ biến là Glad-Sad-Mad.

### **Danh mục kiểm tra Cải tiến Sprint**

- ☐ Nhóm Phát triển có mời thêm thành viên khác tham gia?
- ☐ Có thành viên nào của Nhóm Phát triển thiếu không? Lý do vì sao? Địa điểm, các văn phòng phẩm đã được chuẩn bị?
- ☐ Buổi Cải tiến có diễn ra đúng giờ?

- Mọi thành viên đã rõ mục đích, quy tắc và cách thức thực hiện buổi Cải tiến?
- Có rà soát các hành động cải tiến ở những Sprint trước?
- Đã chuẩn bị các dữ liệu cần thiết cho việc cải tiến?
- Có thành viên nào không tham gia tích cực vào buổi làm việc? Tại sao?
- Có hành động cụ thể được tìm ra?
- Buổi Cải tiến Sprint có giữ đúng khung thời gian?

## **Các kĩ thuật Cải tiến Sprint**



*Glad Sad Mad*



*SpeedBoat*

### **Glad-Sad-Mad**

Glad-Sad-Mad là một kĩ thuật để thực hiện buổi cải tiến dựa trên việc phân loại các ý kiến của thành viên thành 3 nhóm: Glad (vui mừng), Sad (buồn), và Mad (tức giận). Những hạng mục nào mà thành viên đưa ra cảm thấy hài lòng thì được phân loại vào mục Glad. Những hạng mục nào mà họ chưa cảm thấy hài lòng và có thể cải tiến được thì đưa vào mục Sad. Những hạng mục nào mà gây cản trở nghiêm trọng và mình mong muốn loại bỏ nó ngay thì đưa vào mục Mad. Đây là một kĩ thuật được sử dụng phổ biến trong các buổi Cải tiến Sprint.



**Cách tiến hành cụ thể như sau:**



## **1. Chuẩn bị**

Để bắt đầu thì cần chuẩn bị một tấm bảng (tờ giấy - cỡ A0) đủ lớn, chia thành 3 cột: Glad, Sad, Mad và đủ rộng để dán ý kiến của tất cả các thành viên.

## **2. Thu thập dữ liệu**

Người hỗ trợ yêu cầu các thành viên ghi tất cả những quan sát của mình vào từng mảnh giấy dán. Mỗi ý kiến trên một mảnh riêng biệt, sau đó dán lên các cột tương ứng trên bảng. Mỗi ý kiến nên được viết ngắn gọn, súc tích đủ để mọi người hiểu đúng về vấn đề. Hoạt động này cần sử dụng khung thời gian, chẳng hạn 10 hoặc 15 phút.

## **3. Tổng hợp thông tin**

Người hỗ trợ gom các ý kiến trùng nhau lại thành một (loại bỏ), sau đó các thành viên trong nhóm bỏ phiếu để lựa chọn ra những hạng mục mà mình muốn thảo luận. Cách làm phổ biến thường là sử dụng “bỏ phiếu chấm”, tức là mỗi một thành viên được lựa chọn tối đa 3 hạng mục bằng cách ghi một dấu chấm vào mỗi một tờ giấy dán mà mình lựa chọn. Những hạng mục nào có nhiều dấu chấm thì sẽ được thảo luận trước. Người hỗ trợ sẽ sắp xếp lại trật tự các tờ giấy dán dựa trên kết quả bỏ phiếu của cả nhóm.

## **4. Thảo luận và đưa ra hành động cải tiến**

Sau đó, cả nhóm sẽ lần lượt thảo luận từng hạng mục cho đến khi không còn hạng mục nào nữa, hoặc là đã hết thời gian đóng khung. Việc thảo luận nên tập trung vào việc đưa ra những hành động cải tiến có thể thực hiện được trong thời gian tới. Kể cả trong những hạng mục ở cột “Glad” thì cũng vẫn có thể cải tiến để nó tốt hơn.

## **SpeedBoat**

SpeedBoat là kĩ thuật để tiến hành buổi cải tiến dựa trên hình ảnh ẩn dụ của một con tàu cao tốc. Mục đích của kĩ thuật này là xác định độ hài lòng của khách hàng đối với dịch vụ mà mình cung cấp.

Kỹ thuật này cũng được sử dụng phổ biến trong các buổi Cải tiến Sprint.



## **Các bước để triển khai họp cải tiến dùng Speedboat:**

### **1. Chuẩn bị**

Người hỗ trợ vẽ lên bảng (hoặc tờ giấy đủ rộng, cỡ A0) hình ảnh một chiếc tàu cao tốc đang lướt sóng, bổ sung thêm các hình ảnh hỗ trợ và cản trở chuyển động của con tàu như mỏ neo, sóng lớn, v.v..

### **2. Thu thập dữ liệu**

Yêu cầu tất cả mọi người cùng viết ra những ý kiến của mình về cả những yếu tố trở ngại và tích cực trong công việc (chẳng hạn trong Sprint vừa qua). Mỗi một ý kiến được viết lên một tờ giấy dán và lần lượt được dán vào các vùng thích hợp trên hình vẽ.

### **3. Tổng hợp thông tin**

Người hỗ trợ yêu cầu các thành viên nhóm các ý kiến có liên quan lại với nhau. Sau đó, các thành viên sẽ bỏ phiếu để lựa chọn những hạng mục mà mình muốn thực hiện cải tiến. Có thể sử dụng hình thức “bỏ phiếu chấm”, tức là mỗi thành viên được lựa chọn 3 hạng mục và mình muốn thảo luận bằng cách ghi một dấu chấm lên đấy. Những hạng mục nào có nhiều dấu chấm thì sẽ được lựa chọn để cải tiến.

### **4. Đưa ra các hành động cải tiến**

Khi đã lựa chọn được các hạng mục cần cải tiến cả nhóm thảo luận để tìm ra nguyên nhân gốc rễ của vấn đề. Sau đó đề xuất các thay đổi cần thiết thực hiện trong thời gian tới.

### **Ghi chú về những sự kiện khác có liên quan**

Trong chương này chúng ta đã khảo sát những sự kiện chính yếu của Scrum. Trong thực tiễn, có thể chúng ta cần tổ chức thêm những phiên làm việc cộng tác khác đặc thù Agile như Lập kế hoạch Phát hành (Release Planning) để xác định kế hoạch dài hơi cho sự ra đời và phát triển sản phẩm; Workshop viết User Story để chuẩn bị Product Backlog và có thể thêm ước lượng các hạng mục; Họp Làm mịn Product Backlog để cập nhật các hạng mục và độ ưu tiên trong Product Backlog; Workshop Hình dung Kiến trúc tổng thể (Architecture Envisioning Workshop) để thống nhất chung cho toàn đội phát triển về kiến trúc tổng thể được sử dụng trong sản phẩm.

Ngoài ra cũng có thể có hàng loạt những sự kiện khác tùy nhu cầu như Họp khởi công dự án, Huấn luyện trước dự án, Họp nghiệm thu và tổng kết dự án, Họp Kích não tạo ý tưởng (Brainstorming Session) v.v..

Nhóm của bạn nên tìm hiểu thêm và có một kế hoạch tổng thể cho những sự kiện như thế này để thúc đẩy thành công của dự án.

## **CÂU HỎI ỨNG DỤNG**

1. Khung thời gian cho buổi Lập kế hoạch Sprint với một nhóm triển khai Sprint 3 tuần là bao nhiêu?
2. Kết quả cần đạt được sau khi Lập kế hoạch Sprint kết thúc là gì? Nếu bạn là ScrumMaster, khi khung thời gian của sự kiện này đã qua mà nhóm chưa đạt được kết quả theo yêu cầu bạn sẽ làm gì? Tại sao?
3. Nhóm Phát triển và Product Owner cộng tác như thế nào trong Lập kế hoạch Sprint?
4. Bạn thử lý giải việc đóng khung thời gian cho Scrum Hằng ngày chỉ là 15 phút?
5. Nếu trong buổi Scrum Hằng ngày, các thành viên Nhóm Phát triển dành thời gian để giải quyết các vấn đề bạn sẽ làm gì?

6. Là thành viên Nhóm Phát triển bạn có những cách nào để theo dõi tiến độ của nhóm mình?

7. ScrumMaster cần có hành động gì khi thấy đường tiến độ của Nhóm Phát triển trên biểu đồ Sprint Burndown đi trên đường cơ sở?

8. Nhóm Phát triển làm những gì trong buổi Sơ kết Sprint?

9. Thành viên Nhóm Phát triển làm gì trong buổi Cải tiến Sprint?

10. Theo bạn, dấu hiệu của một Sprint thành công sẽ như thế nào?

# 5 CÁCH TẠO TÁC VÀ CÔNG CỤ

Trong chương này...

- Quản lý sản phẩm bằng Product Backlog
- Duy trì một Product Backlog tốt theo tiêu chuẩn DEEP
- Sử dụng Sprint Backlog để quản lý công việc
- Một số kiểu Sprint Backlog thông dụng
- Chuyển giao Phần tăng trưởng cuối mỗi Sprint
- Xây dựng Định nghĩa Hoàn thành
- Các biểu đồ Burndown

*Hành trình vạn dặm bắt đầu với một bước chân nhỏ.*

## Lão Tử

### Tạo tác Scrum

Các tạo tác Scrum gồm một công cụ quản lý yêu cầu sản phẩm gọi là Product Backlog, một công cụ kế hoạch thích ứng trực quan cho nhóm cộng tác là Sprint Backlog, và bản thân gói tăng trưởng chuyển giao được (Increment) ở cuối Sprint. Tạo tác (artifact) ở đây vừa là công cụ vừa là đồ được Nhóm Scrum tạo ra và duy trì liên tục trong quá trình làm việc. Các tạo

tác cung cấp sự minh bạch, hỗ trợ quá trình thanh tra và thích nghi. Việc sử dụng và duy trì tốt các tạo tác này giúp cho tất cả các bên liên quan có được thông tin đầy đủ, hiểu đúng và có cùng một cách hiểu về tình trạng phát triển sản phẩm. Ngoài các tạo tác tiêu chuẩn của

Scrum, các nhóm có thể dùng những công cụ khác để theo dõi tiến độ và thúc đẩy cộng tác.

45

- Danh sách ưu tiên mô tả các tính năng và kết quả của sản phẩm
- Có thể chỉnh sửa được

46

- Danh sách các công việc cần hoàn thành trong Sprint
- Được cập nhật hằng ngày

47

- Tổng tất cả các hạng mục Product Backlog đã được hoàn thành trong một Sprint và giá trị của những phần tăng trưởng của những Sprint trước
- Được chuyển giao ở cuối mỗi Sprint

## PRODUCT BACKLOG

---

Product Backlog là danh sách các tính năng mong muốn của sản phẩm. Danh sách này được sắp xếp dựa trên độ ưu tiên của từng hạng mục. Các hạng mục có độ ưu tiên cao hơn nằm ở phía trên của danh sách và sẽ được Nhóm Phát triển lựa chọn để đưa vào sản xuất sớm, các hạng mục có độ ưu tiên thấp hơn sẽ nằm ở phía cuối của danh sách và được phát triển muộn hơn.

Các Nhóm Scrum có thể dùng nhiều hình thức khác nhau để làm ra Product Backlog. Một lựa chọn thường thấy là bảng trắng hoặc một mảng tường và thẻ chỉ mục (Index Card). Nhiều nhóm (đặc biệt là các nhóm phát triển các sản phẩm lớn hoặc sản phẩm dài ngày) thường dùng phần mềm để làm Product Backlog như Spreadsheet

(Excel hoặc Google Spreadsheet), hay các công cụ quản lý chuyên dụng như Trello, Redmine, Asana, Pivotal Tracker v.v..



### **Ví dụ 1: Product Backlog dùng Spreadsheet**



### **Ví dụ 2: Product Backlog sử dụng công cụ quản lý dự án**



Product Owner là người chịu trách nhiệm quản lý và bảo trì Product Backlog, bao gồm xác định nội dung, đánh giá độ ưu tiên và sắp xếp các hạng mục, làm mịn các hạng mục, làm rõ và giữ cho nội dung của Product Backlog luôn minh bạch với tất cả các bên.

### **Hạng mục Product Backlog**



Product Backlog chứa các hạng mục Product Backlog. Các hạng mục này thường mô tả tính năng của sản phẩm, ngoài ra còn có thể có các hạng mục khác như lỗi, công việc liên quan đến kỹ thuật, công việc nghiên cứu.

Scrum không quy định hình thức cụ thể để thể hiện hạng mục Product Backlog, nhưng trong thực tế thì kỹ thuật được sử dụng phổ biến là User Story (xem chi tiết trong Chương 7).

Scrum không quy định hình thức cụ thể để thể hiện hạng mục Product Backlog, nhưng trong thực tế thì kỹ thuật được sử dụng phổ biến là User Story.

### **Đánh giá độ ưu tiên**

Các hạng mục trong Product Backlog được sắp xếp dựa trên độ ưu tiên, do đó nhiệm vụ của Product Owner là đánh giá độ ưu tiên của từng hạng mục. Product Owner cần phải xem xét một số yếu tố khác nhau để xác định độ ưu tiên này, chẳng hạn như:

- Giá trị đối với phần lớn khách hàng hay người dùng
- Giá trị đối với những khách hàng hay người dùng quan trọng
- Chi phí
- Thời gian để triển khai
- Mối quan hệ với các hạng mục khác
- Rủi ro và cơ hội

Độ ưu tiên của các hạng mục có thể được cập nhật liên tục trong quá trình phát triển, khi Product Owner đã có thêm những thông tin mới, hiểu biết nhiều hơn về sản phẩm và có thể có những điều chỉnh về lộ trình của sản phẩm.

### **Ước tính các hạng mục Product Backlog**

Mỗi hạng mục Product Backlog được ước tính kích thước, tức là lượng nỗ lực cần để triển khai nó. Việc này được thực hiện bởi Nhóm Phát triển. Kích thước của một hạng mục có thể ảnh hưởng đến độ ưu tiên của nó trên Product Backlog.

Việc ước tính có thể được thực hiện theo nhiều cách khác nhau. Các nhóm Scrum thường sử dụng trò chơi Planning Poker để làm việc này (xem Chương 7).

Kết thúc việc ước tính các hạng mục Product Backlog, bạn có thể gán các giá trị xác định độ lớn, ví dụ: số điểm, lên trên các hạng mục Product Backlog. Bạn cũng có thể gán kích thước giống như cỡ áo (ví dụ: XS, S, M, L, XL, XXL, XXXL) để đánh dấu độ lớn



tương ứng của các hạng mục. Việc này sẽ giúp bạn quản lí Product Backlog thuận lợi hơn. Đặc biệt, nó sẽ giúp bạn làm mịn Product Backlog sau này.

## **Làm mịn Product Backlog**



Các hạng mục Product Backlog có kích thước khác nhau, mức độ chi tiết khác nhau và chúng thường cần phải được làm mịn trước khi sẵn sàng để đưa vào sản xuất. Những hạng mục ở phần trên thường là “mịn” hơn so với những hạng mục ở dưới.

Việc làm mịn có thể đơn giản là chi tiết hoá một hạng mục nào đó còn chung chung. Ví dụ một hạng mục “Quản lí user” có thể được phân tách thành “Thêm user”, “Sửa user”, “Liệt kê toàn bộ user” và “Tìm kiếm một user”. Làm mịn cũng có thể bao gồm cả việc bẻ nhỏ những hạng mục có kích thước lớn (ví dụ: những hạng mục cỡ XL trở lên) thành những hạng mục cỡ nhỏ hơn. Càng nhỏ, càng chi tiết, càng rõ ràng thì càng mịn.

Một Nhóm Scrum có thể phải dành ra khoảng 10% thời gian của một Sprint để ngồi cùng với nhau để họp và làm mịn lại Product Backlog, đảm bảo cho các phiên Lập kế hoạch Sprint tiếp theo được thuận lợi, cũng như luôn cập nhật trạng thái hiểu biết tốt nhất về yêu cầu sản phẩm. Phiên họp làm mịn Product Backlog này diễn ra trong Sprint, có thể là theo nhu cầu, hoặc cố định, tùy sắp đặt của nhóm.

## **Tiêu chuẩn DEEP dành cho Product Backlog**

DEEP là một tiêu chuẩn được sử dụng khá phổ biến hướng đến việc xây dựng một Product Backlog tốt.

DEEP quy định 4 tính chất mà một Product Backlog nên có, bao gồm:

**D - Detailed appropriately (Đủ chi tiết hợp lý):** Những hạng mục có độ ưu tiên cao hơn thì có nhiều chi tiết cụ thể hơn, rõ ràng hơn so với những hạng mục có độ ưu thấp.

**E - Estimated (Được ước tính):** Tất cả các hạng mục Product Backlog phải được ước tính. Độ chính xác của việc ước tính các hạng mục ở trên thường là cao hơn so với phần ở dưới.

**E - Emergent (Tiến hóa):** Product Backlog không phải là một danh sách cố định các hạng mục cần phát triển mà nó luôn được tiến hóa trong suốt quá trình phát triển dựa theo những hiểu biết học được.

**P - Prioritized (Được sắp xếp ưu tiên):** Các hạng mục trong Product Backlog được sắp xếp theo độ ưu tiên. Những hạng mục có độ ưu tiên cao hơn sẽ được đưa vào sản xuất sớm, chúng nằm phía trên của Product Backlog.

⇒ **Dừng và Nghĩ**

**Nhóm của bạn có quản lý yêu cầu một cách có hệ thống không?  
Cần làm gì để làm tốt hơn nữa việc này?**

## **SPRINT BACKLOG**

---

Sprint Backlog là bảng công việc được Nhóm Phát triển sử dụng để quản lý quá trình sản xuất trong một Sprint. Nội dung cơ bản của một Sprint Backlog bao gồm danh sách các hạng mục Product Backlog đã lựa chọn cho Sprint hiện tại và danh sách các công việc cần thực hiện trong Sprint để hoàn thành các hạng mục Product Backlog đó.



*Một nhóm Scrum đang đứng họp Scrum Hằng ngày trước Sprint Backlog được dán lên tường.*

*Thông thường, sau buổi họp này, các thành viên sẽ cập nhật Sprint Backlog và biểu đồ Sprint Burndown.*

## **Sprint Backlog do Nhóm Phát triển duy trì**



Sprint Backlog do Nhóm Phát triển sở hữu và vận hành bởi vì quá trình làm việc trong Sprint cũng do Nhóm Phát triển tự quản lí, đây là một đặc điểm quan trọng trong Scrum. Nhóm Phát triển tạo Sprint Backlog ở trong buổi Lập kế hoạch Sprint và vận hành nó cho đến khi Sprint kết thúc.

Giống như Product Backlog, các nhóm có thể sử dụng các phương tiện khác nhau để làm ra Sprint Backlog tiện lợi nhất cho mình. Nhưng điều quan trọng hơn cả là luôn đảm bảo Sprint Backlog hiện diện và cập nhật.



## **Sprint Backlog dạng Spreadsheet**

Có nhiều cách để thể hiện Sprint Backlog, tùy theo sự lựa chọn và tính phù hợp đối với nhóm.

Ví dụ, cách trình bày dưới dạng một bảng truyền thống như sau:



## **Sprint Backlog dạng Kanban**

Một dạng cấu trúc khác được ưa chuộng và áp dụng rộng rãi hiện nay là bảng Kanban bao gồm các cột.

Với dạng này, Sprint Backlog được trình bày như sau:



## **PHẦN TĂNG TRƯỞNG CHUYỂN GIAO ĐƯỢC**

---

Phần tăng trưởng (tên gọi ngắn của Phần tăng trưởng Sản phẩm Có khả năng Chuyển giao được) là phần sản phẩm Nhóm Phát triển tạo ra cuối mỗi Sprint.

Đây là một khái niệm quan trọng trong Scrum, tạo ra sự khác biệt lớn về mặt sản phẩm so với các phương pháp truyền thống.

### **Khác biệt với tư duy phát triển truyền thống**

Ở các phương pháp phát triển truyền thống, quá trình phát triển sản phẩm được tiến hành lần lượt theo từng bước, chẳng hạn đi từ thu thập yêu cầu, phân tích và thiết kế, xây dựng, kiểm thử rồi đi đến phát hành. Do đó, ở phần lớn thời gian đầu, kết quả thu được là rất nhiều tài liệu, bao gồm cả tài liệu mô tả yêu cầu lẫn tài liệu thiết kế. Trong khi đó thiếu vắng một sản phẩm thật có thể sử dụng được bởi người dùng.



*Trong phương pháp phát triển truyền thống, sản phẩm chỉ được hình thành ở giai đoạn cuối của chu trình phát triển*

Scrum có một hướng tiếp cận khác, không chỉ đơn giản tách quá trình phát triển thành các Sprint nhỏ liên tiếp nhau, mà cuối mỗi Sprint đòi hỏi Nhóm Phát triển phải chuyển giao một phần tính năng hoàn chỉnh của sản phẩm.



Các phần tăng trưởng được tạo ra liên tục ở mỗi Sprint

### **Các đặc điểm của Phần tăng trưởng**

- Sử dụng được: Tính năng đạt được sau mỗi Sprint là thật và sử dụng được ngay chứ không chỉ là một bản thiết kế hay một bản mẫu.

- Có khả năng chuyển giao được: Phần tăng trưởng ở cuối mỗi Sprint cần phải có khả năng chuyển giao được, Điều này không có nghĩa là nó phải lập tức được chuyển giao ngay, mà có nghĩa là nó đạt được trạng thái sẵn sàng chuyển giao.
- Hoàn thành: Phần tăng trưởng phải tuân thủ theo Định nghĩa Hoàn thành đã được thống nhất trước đó.

## **Tại sao chúng ta cần Phần tăng trưởng?**

Việc tạo ra được các phần tăng trưởng sau mỗi Sprint mang lại rất nhiều lợi ích từ những góc độ khác nhau, có thể kể đến như:

- Sớm có được phản hồi: Các phản hồi này sẽ được ghi nhận để đưa ra những điều chỉnh cho sản phẩm nếu cần thiết.
- Sớm thu được giá trị: Chúng ta sớm thấy được giá trị của sản phẩm và giá trị này cũng đến tay người dùng sớm hơn.
- Tăng minh bạch: Sau những khoảng thời gian ngắn, chúng ta đã có được những phần sản phẩm thực sự “hoàn thành” mà tất cả mọi người đều có thể nhìn thấy được, tương tác được.
- Giảm rủi ro: Phần tăng trưởng giúp giảm thiểu rủi ro, xét về cả mặt kinh doanh lẫn kĩ thuật.

## **ĐỊNH NGHĨA HOÀN THÀNH**



Định nghĩa Hoàn thành là một danh sách quy định những công việc cần được thực hiện xong trước khi một hạng mục được công nhận là đạt trạng thái có khả năng chuyển giao được.

Định nghĩa Hoàn thành là một công cụ quan trọng để đảm bảo tính minh bạch của các tạo tác trong Scrum, tiêu chuẩn để thành viên nhóm tự kiểm tra và đạt được chất lượng tự thân.

Ngoài ra, Định nghĩa Hoàn thành giúp mọi người hiểu chung về yêu cầu công việc, ngăn ngừa những mâu thuẫn do hiểu nhầm hoặc không hiểu rõ. Gia tăng tính minh bạch về yêu cầu giúp nhóm tự tổ chức hiệu quả hơn.

## **Nội dung của Định nghĩa Hoàn thành**

Nội dung của Định nghĩa Hoàn thành dành cho từng nhóm và từng sản phẩm là khác nhau, điều này phụ thuộc vào một số yếu tố như:

- Đặc điểm của sản phẩm đang xây dựng
- Công nghệ và kĩ thuật đang được sử dụng
- Các quy định của tổ chức
- Trình độ của Nhóm Phát triển

Chẳng hạn, nếu một sản phẩm được cung cấp ở nhiều thị trường khác nhau thì có thể có những quy định liên quan đến ngôn ngữ hiển thị, tiền tệ, định dạng thời gian. Nếu là một ứng dụng web thì có thể có những quy định liên quan đến việc hỗ trợ tốt trên các loại trình duyệt khác nhau. Nếu tổ chức có những quy ước liên quan đến việc phát triển thì có thể có những quy định liên quan đến việc tuân thủ các quy ước này. Nếu trình độ của Nhóm Phát triển cao thì có thể có những quy định rất chặt chẽ về mã nguồn, ví dụ như là các phương thức không vượt quá 8 dòng mã, v.v..



## **Nhóm Scrum xây dựng Định nghĩa Hoàn thành**

Việc xây dựng Định nghĩa Hoàn thành cần có sự tham gia của toàn bộ Nhóm Scrum, tức là bao gồm các thành viên Nhóm Phát triển, Product Owner và ScrumMaster. Thông thường, Product Owner là người có vai trò xác nhận cuối cùng đối với Định nghĩa Hoàn thành.

Định nghĩa Hoàn thành cũng không phải là một tài liệu cố định mà sẽ được cải tiến theo quá trình trưởng thành của nhóm để phản ánh

sự trưởng thành này cũng như nâng cao chất lượng của sản phẩm.



## **Quá trình tiến hóa của “Hoàn thành”**



### **Xây dựng Định nghĩa Hoàn thành cho từng cấp độ khác nhau**

Các ví dụ về Định nghĩa Hoàn thành được đề cập ở trên là áp dụng cho các User story để được công nhận là đạt trạng thái sẵn sàng chuyển giao. Tuy nhiên, trên thực tế thì Định nghĩa Hoàn thành có thể được áp dụng cho những cấp độ khác nữa, chẳng hạn như đối với các User story để quy định trạng thái sẵn sàng để đưa vào phát triển, đối với các công việc trong Sprint và đối với các Phiên bản phát hành.

### **Định nghĩa Hoàn thành dành cho User Story**

Đối với các User story, chúng ta có thể xây dựng Định nghĩa Hoàn thành để quy định trạng thái của một User story được công nhận là sẵn sàng để đưa vào phát triển trong một Sprint. Loại Định nghĩa Hoàn thành này dành cho User story còn được gọi là Định nghĩa Sẵn sàng.

- User story đã được ước tính
- Có danh sách các tiêu chí chấp nhận
- User story là duy nhất
- Có đính kèm tài liệu phù hợp (chẳng hạn như bản mẫu, dữ liệu,...)
- Giá trị ước tính ban đầu không nhiều hơn 5 điểm

### **Định nghĩa Hoàn thành dành cho công việc trong Sprint**

Đối với các công việc trong Sprint, Nhóm Phát triển có thể đưa ra Định nghĩa Hoàn thành để quy định trạng thái cuối của từng công việc. Một phần quan trọng của Định nghĩa Hoàn thành này thường đề cập đến các yêu cầu về mặt kỹ thuật cũng như quy trình để cộng tác trong nhóm.

- Mã nguồn đã được kiểm tra chéo
- Mã nguồn có đầy đủ chú thích
- Mã nguồn đã được đưa lên hệ thống lưu trữ
- Đã cập nhật thời gian của công việc trên Sprint Back- log về 0

### **Định nghĩa Hoàn thành dành cho Phiên bản Phát hành**

Đối với các Phiên bản Phát hành, Nhóm Scrum và các bên liên quan có thể xây dựng Định nghĩa Hoàn thành nhằm đưa ra các quy định mà một bản phát hành cần phải thỏa mãn. Nội dung của Định nghĩa Hoàn thành này thường tập trung vào các điều kiện để cài đặt sản phẩm, sử dụng sản phẩm, tính ổn định của sản phẩm và việc hỗ trợ người dùng.

- Tất cả mã nguồn đều đã được triển khai trên máy chủ thực tế
- Sản phẩm được dùng thử bởi 10 người ngoài Nhóm Phát triển trong vòng 5 ngày
- Không có lỗi được phát hiện
- Nhóm chăm sóc khách hàng đã được huấn luyện về các tính năng mới

### **Định nghĩa Hoàn thành khác với Tiêu chí chấp nhận**

Đối với User story, có hai khái niệm thường gây ra nhầm lẫn cho những người mới bắt đầu đó là Định nghĩa Hoàn thành và Tiêu chí Chấp nhận. Việc phân biệt rõ hai khái niệm này là cần thiết để đảm



bảo việc sử dụng đúng và hiệu quả. Chúng ta có thể phân biệt dựa trên phạm vi áp dụng của chúng.

Lưu ý, Định nghĩa Hoàn thành được áp dụng cho tất cả các User story trong khi đó Tiêu chí Chấp nhận chỉ được áp dụng cho một hoặc một vài User story nhất định.

- Hoạt động tốt với các loại thẻ Visa, Master
- Thời gian phản hồi không vượt quá 20 giây
- Hỗ trợ thanh toán bằng USD, EURO, VNĐ

*Một ví dụ về Tiêu chí Chấp nhận cho tính năng thanh toán trực tuyến*

## **BIỂU ĐỒ SPRINT BURNDOWN**

---

Biểu đồ Sprint Burndown là một công cụ được Nhóm Phát triển sử dụng để trực quan hóa tiến độ hướng tới Mục tiêu Sprint. Nhóm Phát triển tập trung sự quan tâm của mình vào lượng nỗ lực còn lại cần thực hiện để đạt được Mục tiêu Sprint chứ không phải là mình đã làm được những gì.

Nhóm Phát triển tạo ra Sprint Burndown sau buổi Lập kế hoạch Sprint và cập nhật nó hàng ngày để thấy được tiến độ thực của mình. Nếu có những biểu hiện không tốt về mặt tiến độ công việc thì nhóm có thể đưa ra các quyết định điều chỉnh nhanh chóng một cách chủ động để hướng đến việc hoàn thành Mục tiêu Sprint đúng như cam kết của mình.



Nhóm Phát triển dựa vào Sprint Backlog để tính tổng lượng nỗ lực còn lại cần thực hiện để đạt được mục tiêu Sprint. Mỗi công việc trên Sprint Backlog đều đã được ước tính trong buổi Lập kế hoạch

Sprint, tổng các giá trị ước tính của những công việc chưa hoàn thành chính là lượng nỗ lực còn lại.

Hàng ngày, khi đã có một số công việc được hoàn thành, nhóm cập nhật lại lượng nỗ lực cho các công việc chưa hoàn thành.

## Cách vẽ Sprint Burndown

Biểu đồ Sprint Burndown có 2 trục. Trục hoành thể hiện thời gian, được chia thành các ngày tương ứng với thời gian của một Sprint. Ví dụ, đối với Sprint 1 tuần có 5 ngày làm việc thì chúng ta sẽ chia thành các ngày: 1, 2, 3, 4, 5...

Trục tung thể hiện lượng nỗ lực còn lại cần thiết để hoàn thành tất cả các công việc. Giá trị này được lấy từ Sprint Backlog.

img694

## Cập nhật Sprint Burndown

Hàng ngày, Nhóm Phát triển tính lượng nỗ lực còn lại và cập nhật vào Sprint Burndown.

Tổng lượng nỗ lực còn lại theo từng ngày lần lượt là 71, 58, 45

68

## Đường cơ sở của Sprint Burndown

69

Trong biểu đồ Sprint Burndown, chúng ta có một đường cơ sở được nối từ điểm khởi đầu cho đến điểm mong muốn đạt được khi kết thúc Sprint. Tức là, trong trường hợp lý tưởng nhất, tốc độ sản xuất của nhóm luôn ổn định và lượng nỗ lực công việc còn lại sau từng ngày sẽ giảm dần đều trong suốt Sprint, đạt được giá trị mong muốn vào đúng ngày cuối cùng.

70

Nếu đường đồ thị thực tế nằm ở phía trên của đường cơ sở có nghĩa là tốc độ sản xuất của nhóm đang không đạt như kỳ vọng. Nhóm cần xem xét và phân tích để tìm ra nguyên nhân, có những điều chỉnh nếu thực sự cần thiết để tăng năng suất của nhóm. Bởi vì nếu tiếp tục như vậy thì đến cuối Sprint nhóm sẽ không thể đạt được mục tiêu.



Trong trường hợp thuận lợi, tốc độ sản xuất nhóm cao hơn ước tính, đường đồ thị thực tế sẽ nằm phía dưới của đường đồ thị cơ sở. Điều này có nghĩa là chúng ta sẽ đạt được mục tiêu, hoàn thành tất cả các công việc sớm hơn ước tính.

Thường thì việc này không gây ra tác hại gì với Sprint hiện tại. Nhưng nếu sự chênh lệch là khá lớn và xảy ra nhiều thì nhóm cần chú ý hơn đến việc lập kế hoạch. Bởi vì những yếu tố có thể gây ra hiện tượng này đó là việc ước tính nỗ lực dành cho các công việc hoặc ước tính khả năng của nhóm đang quá sai lệch so với thực tế.

## **Công cụ vẽ Sprint Burndown**

Trong thực tế, bạn rất dễ dàng để sử dụng các công cụ điện tử để vẽ biểu đồ Sprint Burndown, từ bảng tính cho đến tính năng của các công cụ quản lý Backlog.

Tuy nhiên, nhiều nhóm thích tự tay vẽ ra và treo ngay cạnh Sprint Backlog trong khu vực nhóm làm việc. Cách làm này giúp gia tăng tính trực quan, và tăng khí thế làm việc cho nhóm.



## **BIỂU ĐỒ RELEASE BURNDOWN**

---

Biểu đồ Release Burndown được Product Owner sử dụng để theo dõi tiến độ của sản phẩm. Đường biểu đồ thể hiện tổng lượng nỗ lực còn lại cần thực hiện để đạt được mục tiêu Phát hành Sản

phẩm. Nếu có những biểu hiện không tốt về mặt tiến độ phát triển sản phẩm thì Product Owner và những thành viên khác có thể đưa ra những điều chỉnh sớm và chủ động để đạt được một kết quả tốt và hợp lý nhất về mặt sản phẩm.



## Cách vẽ Biểu đồ Release Burndown



Biểu đồ Release Burndown gồm 2 trục.

Trục hoành thể hiện thời gian thông qua các Sprint liên tiếp nhau. Chẳng hạn, chúng ta sử dụng ngày bắt đầu của Sprint để đánh dấu lần lượt các Sprint.

Trục tung thể hiện lượng nỗ lực còn lại cho đến khi đạt được mục tiêu.

## Duy trì Biểu đồ Release Burndown

Sau mỗi Sprint, Product Owner tính lại tổng lượng nỗ lực còn lại để đạt được mục tiêu Phát hành, giá trị này được thể hiện trên biểu đồ Release Burndown.

Tổng lượng nỗ lực còn lại lần lượt sau mỗi Sprint là: 90, 77, 62



## Đường cơ sở của Biểu đồ Release Burndown

Trên Biểu đồ Release Burndown này chúng ta có thể vẽ một đường cơ sở thể hiện mối quan hệ giữa 3 yếu tố là tổng lượng nỗ lực ước tính ban đầu, tốc độ của Nhóm Phát triển và ngày phát hành dự kiến.

Chẳng hạn, trong trường hợp này thì tổng lượng nỗ lực ước tính ban đầu là 90 điểm, tốc độ của Nhóm Phát triển là 15 và ngày phát

hành là sau khi kết thúc Sprint thứ 6.



### **Chậm tiến độ so với dự kiến**

Nếu Nhóm Phát triển không hoàn thành tốt các mục tiêu Sprint, dẫn đến một lượng nỗ lực không được hoàn thành theo như dự kiến. Có nghĩa là, nếu tiếp tục với tốc độ như vậy, nhóm sẽ khó hoàn thành được phiên bản phát hành theo như kế hoạch ban đầu.

Trong trường hợp này, việc đưa ra các điều chỉnh là cần thiết. Chúng ta cần phải xem xét 3 yếu tố là tốc độ sản xuất của Nhóm Phát triển, các tính năng mong muốn của sản phẩm và ngày phát hành.



Nếu muốn giữ nguyên các tính năng mong muốn của sản phẩm và không thay đổi ngày phát hành thì cần có các điều chỉnh đối với

Nhóm Phát triển để cải thiện tốc độ phát triển.

Ví dụ, trong trường hợp này, cần có những điều chỉnh để nâng tốc độ sản xuất lên 21 điểm.

Nếu muốn giữ nguyên tất cả các tính năng của sản phẩm nhưng lại không tăng được khả năng của Nhóm Phát triển thì cần phải xem xét lùi ngày phát hành. Ví dụ, trong trường hợp này sẽ cần thêm 2 Sprint nữa để hoàn thành hết tất cả các tính năng của sản phẩm.

Nếu muốn phát hành sản phẩm đúng ngày đã định nhưng cũng không tăng được khả năng của Nhóm Phát triển thì cần phải xem xét loại bỏ bớt một số hạng mục nhằm giảm bớt lượng công việc mà

Nhóm Phát triển cần thực hiện. Chẳng hạn, trong trường hợp này cần loại bỏ 22 điểm khỏi Product Backlog.



## Công cụ vẽ Biểu đồ Release Burndown



Trong thực tế, tính năng theo dõi tiến độ sản phẩm thông qua Biểu đồ Burndown đều được hỗ trợ ở hầu hết các công cụ quản lý backlog, chẳng hạn như JIRA, Assembla, Backlog, Pivotal Tracker hay sử dụng Excel và nhiều công cụ khác.

Tuy nhiên, Product Owner cũng có thể tự thiết kế một Biểu đồ Release Burndown vật lý để nâng cao tính trực quan và dễ dàng tùy chỉnh theo ý muốn.



## CÂU HỎI ỨNG DỤNG

1. Khi Nhóm Phát triển nhận thấy cần bổ sung thêm một hạng mục vào Product Backlog họ phải làm gì? Tại sao?
2. Sprint chạy được 1 ngày thì Product Owner nhận thấy cần phải hoàn thành sớm một hạng mục có giá trị nào đó thì ta gán luôn vào Sprint Backlog, với cương vị ScrumMaster bạn sẽ làm gì?
3. Tại sao Nhóm phát triển ước tính nỗ lực cần thiết để hoàn thành các hạng mục trong Sprint Backlog?
4. Những giá trị của biểu đồ Sprint Burndown là gì?
5. Đây là mối liên hệ giữa Product Backlog và Sprint Backlog?
6. Kết thúc một Sprint thì tạo tác nào nói lên sự thành công của Sprint? Tại sao?
7. Các hạng mục trong Product Backlog được sắp xếp ưu tiên như thế nào? Ai có quyền cao nhất trong việc đó?
8. Bạn giải thích với một đồng nghiệp như thế nào khi anh ta nói: Product Owner là người quản lý Product Backlog nên sẽ là người

ước tính các hạng mục trong đó?

9. Bạn tư vấn công cụ nào làm Product Back- log và Sprint Backlog cho một Nhóm Scrum trong một Startup cùng làm việc ở một văn phòng? Tại sao?

10. Đây là những nguyên nhân có thể khiến một Nhóm Phát triển gặp khó khăn trong việc xác định hạng mục nào đó của Product Backlog đã hoàn thành hay chưa?

# 6SCRUM TRONG TỔ CHỨC

Trong chương này...

- Các vấn đề khi đưa Scrum vào tổ chức
- Các tình huống áp dụng Scrum
- Quản trị dự án linh hoạt với Scrum
- Các khía cạnh văn hoá cần quan tâm
- Lộ trình áp dụng Scrum
- Nhận diện và vượt qua những trở ngại
- Quản lý sự thay đổi

*Scrum dễ hiểu, nhưng khó tinh thông.*

**Ken Schwaber**

## CÁC VẤN ĐỀ TỔ CHỨC CẦN LƯU TÂM

---

Khi bạn quyết định mang Scrum về tổ chức của mình, bạn sẽ phải giải quyết một loạt các vấn đề để tổ chức của bạn đón nhận Scrum thành công và gặt hái những lợi ích do Scrum mang lại. Những vấn đề nổi cộm có thể dễ dàng nhìn ra như: thay đổi các vai trò công việc của cá nhân và phòng ban hiện có, thay đổi cách thức đánh giá kết quả, và vẽ lại lộ trình nghề nghiệp cho hàng loạt vị trí cũ và mới.

Scrum sẽ xung đột vai trò/tổ chức: bạn có thể sẽ phải định nghĩa lại công việc của quản trị dự án do cách thức quản trị dự án đã thay đổi, sắp xếp lại chức năng và nhiệm vụ của những phòng ban chức năng để tổ chức những nhóm Scrum liên chức năng, những phòng



ban truyền thống như phòng đảm bảo chất lượng, các phòng kiểm thử, phòng thiết kế..., có thể sẽ biến mất khỏi cơ cấu tổ chức, hoặc sẽ phải định vị lại chức năng và nhiệm vụ.

Song song với việc thay đổi các vị trí, phòng ban, tổ chức của bạn cũng sẽ phải thay đổi lại cách thức đánh giá hiệu suất, định nghĩa lại sự thành công để vừa đảm bảo cá nhân vẫn nỗ lực, nhưng lại khuyến khích tinh thần hợp tác; vừa đánh giá đúng giá trị của cá nhân, lại vừa đánh giá thành tích dựa trên sự chung sức trong tập thể. Đây vừa là vấn đề kĩ thuật, nhưng cũng là vấn đề mang tính giá trị quan cần sự thay đổi sâu sắc trong tư duy của lãnh đạo.

Khi đưa Scrum vào, hàng loạt vai trò mới phát sinh như ScrumMaster, Product Owner, Agile Coach, Scrum Coach, và những công việc của từng thành viên trong đội phát triển sẽ thay về chất. Điều này đòi hỏi bộ phận quản lí nhân sự và lãnh đạo phải quy hoạch lại lộ trình chuyên nghiệp, hệ thống mô tả nghiệp vụ và những chính sách đi kèm để phát triển đội ngũ vững mạnh đáp ứng nhiệm vụ theo cách làm mới mà Scrum mang lại. Không chú trọng tới yếu tố này thì những cách áp dụng Scrum chỉ mang tính bề mặt, thiếu bền vững và khó lòng gặt được thành công lâu dài.

Cuối cùng, việc du nhập một cách làm mới, một công cụ mới hay một công nghệ mới về bản chất là một sự thay đổi, lãnh đạo công ty sẽ phải lập kế hoạch quản lí sự thay đổi để đảm bảo sự du nhập Scrum thành công, vượt qua những trở ngại và phản đối để có những thay đổi căn bản, cắm rễ lâu bền vào văn hóa của công ty.

## **CÁC TÌNH HUỐNG VÀ MỨC ĐỘ SỬ DỤNG SCRUM**

---



Ở mức độ thấp nhất, Scrum được dùng với mục đích thí điểm. Khi doanh nghiệp chưa sẵn sàng để dùng Scrum cho các dự án quan trọng, thì mô hình này tỏ ra phù hợp. Theo đó Nhóm Scrum được lập ra và thực hiện một nhiệm vụ (thật hoặc giả lập) ngắn hạn chừng vài Sprint để kiểm tra xem Scrum có phù hợp với doanh

nghiệp mình không, đội ngũ có dễ dàng làm chủ Scrum không, và có những vấn đề phát sinh nào. Nhiều doanh nghiệp chọn bước thí điểm là bước khởi đầu trước khi áp dụng đồng loạt Scrum cho tổ chức.

Mức độ phổ biến nhất là các doanh nghiệp sử dụng Scrum như là một khuôn khổ mới cho quản trị dự án. Theo đó, tùy theo tính trọng yếu mà một số dự án sẽ được tổ chức dưới dạng Scrum. Đây là hình thức đầu tư ngắn hạn, thu lợi nhanh, phù hợp với các công việc thời vụ hoặc do doanh nghiệp chưa có chiến lược sử dụng Scrum lâu dài.

Mức độ cao hơn của việc vận dụng Scrum là dùng nó làm quy trình tiêu chuẩn để sản xuất phần mềm. Trong trường hợp này toàn bộ đội ngũ sản xuất được tổ chức dưới dạng các nhóm Scrum, nếu sản phẩm lớn hơn sẽ áp dụng những cách vận dụng Scrum ô lớn (tham khảo Chương 9) để tổ chức sản xuất. Đây là cách làm đòi hỏi chiến dài, đầu tư bài bản cho con người, lộ trình nghề nghiệp, hạ tầng công nghệ và cơ sở vật chất đi kèm.

Trong trường hợp bên trên, Scrum vẫn chỉ đóng vai trò thu hẹp trong đội ngũ phát triển sản phẩm. Trường hợp sử dụng Scrum với phạm vi lớn hơn là khi toàn bộ tổ chức tạo lập được văn hoá Scrum để cùng hoạt động. Tổ chức sẽ lấy các giá trị Scrum làm giá trị của văn hoá công ty, các lãnh đạo sẽ vận dụng tư duy Agile và các quy trình

Scrum cho các lĩnh vực vận hành, phát triển sản phẩm, và dịch vụ. Thông thường, các nhỏ hơn (như các startup vài người đến vài chục người) cho đến những công ty sẽ dễ dàng vận dụng Scrum theo cách này. Các doanh nghiệp lớn hơn sẽ đòi hỏi chiến lược chuyển đổi bài bản cùng với các kế hoạch quản trị thay đổi đường dài.

## **Lộ trình đơn giản để đưa Scrum vào tổ chức**

## ⇒ Dừng và Nghĩ

Dưới đây là những trở ngại thường thấy khi dùng Scrum.

### Nhóm của bạn sẽ vượt qua bằng cách nào?

- Những ảo tưởng về mệnh lệnh và kiểm soát (command-and-control), vẫn nghĩ rằng cách làm việc truyền thống là thứ không thể bỏ qua và vẫn hiệu quả.
- Vẫn giữ các lễ thói cũ, ví dụ: khoác lên Quản trị Dự án cái áo ScrumMaster, trong khi vẫn làm y nguyên như cũ; họp hằng ngày, nhưng thành viên báo cáo cho ScrumMaster, v.v..
- Tính “hiển nhiên” của ScrumBut (Scrum nhưng mà...), với lí do là “Scrum kia là của người ta, đến đây thì phải thay đổi cho phù hợp”, mà quên mất rằng khi đã làm ScrumBut ngay từ đầu thì tức là đội nhóm chưa từng học được Scrum để làm việc.
- Trông chờ phép màu nào đó: Công ty có quá nhiều vấn đề, Scrum sẽ mang lại liều thuốc chữa trị bách bệnh. Sự thật là không có liều thuốc nào như thế, kể cả Scrum.
- Thiếu sự minh bạch: làm Scrum nhưng thông tin về dự án, về khách hàng, về yêu cầu không minh bạch, dẫn đến không thể đảm bảo được ba trụ cột của Scrum, cản trở nhóm Scrum tự tổ chức đạt được kết quả tốt.
- “Quán tính Waterfall”, khi ta nói quán tính xã hội, tức là một thứ đã tồn tại lâu (văn hoá công ty), nó sẽ cản trở những cái mới.
- Các bộ phận hoàn toàn đứng ngoài (HR, QA, PMO, ...)

## QUẢN TRỊ DỰ ÁN LINH HOẠT VỚI SCRUM

---

Như trên đã nói, quản trị dự án là một trong các tình huống phổ biến nhất khi một doanh nghiệp bắt đầu vận dụng Scrum. Lợi ích của cách làm này là hạn chế sự ảnh hưởng quá lớn tới cơ cấu tổ chức

và văn hoá công ty trong khi vẫn đạt được những lợi ích mà Scrum mang lại như khả năng chuyển giao nhanh chóng, thích ứng nhanh, cải tiến liên tục quy trình làm việc và đảm bảo chất lượng cao.

Việc vận dụng Scrum vào quản trị dự án có thể được tiến hành khá thuận lợi với việc định nghĩa lại các vai trò trong nhóm dự án và chuyển đổi quy trình dự án sang các Sprint. Điều cần lưu tâm nhất có lẽ là nằm ở việc định nghĩa lại vai trò của nhà quản trị dự án.

Nhiều người nhầm lẫn vai trò của ScrumMaster với Project Manager, và gọi ScrumMaster là một “project manager” kiểu mới. Thực tế khác hẳn. Công việc của Project Manager truyền thống đã được phân bổ lại cho các vai trò của một Nhóm Scrum để nhóm tự quản lí lấy công việc của mình. Do vậy đối với các công việc được định nghĩa trong Scrum cần đảm bảo, tránh ScrumBut. Người Project Manager hiện có của nhóm có thể đóng vai trò mới (Product Owner hoặc ScrumMaster) nhưng không được vượt qua các quy tắc mà Scrum đã thiết lập.

Bên cạnh việc định nghĩa lại quy trình và các vai trò, quản trị dựa án linh hoạt (hoặc ScrumMaster) cần thiết lập được một hệ thống báo cáo (dùng các công cụ Backlog hoặc Project Dashboard) phù hợp để luôn đảm bảo thông tin minh bạch đối với các cấp quản lí và các bên liên quan. Trong khả năng có thể, nhóm dự án cần tận dụng tối đa sự tham gia của khách hàng, lãnh đạo và các bên liên quan để thúc đẩy dự án tiến lên.

Đối với các dự án lần đầu được dùng Scrum, khâu đào tạo kĩ năng là rất quan trọng. Thực chất hàng loạt kĩ năng quan trọng để vận hành cách làm mới có thể còn quá lạ lẫm với các thành viên trong nhóm. Đối với thành viên Nhóm Phát triển, hàng loạt kĩ năng có thể là mới tinh: giao tiếp với đồng nghiệp, hợp tác cực và hiệu quả, nêu vấn đề rõ ràng và kĩ thuật giải quyết vấn đề có hệ thống, tự rút kinh nghiệm và đóng góp trong các phiên cải tiến liên tục, kĩ năng tự thanh tra và tự lập kế hoạch cá nhân. Đối với Product Owner, có thể việc phải bảo trì Product- Backlog là công việc rất lạ lẫm, vì “ngày xưa, chỉ làm yêu cầu một lần là xong”. Đối với ScrumMaster, việc tháo gỡ khó khăn cho nhóm, hoặc phải “đứng ngoài lè thúc đẩy chứ

không can thiệp thô bạo đối với công việc của Nhóm Phát triển” là một nhiệm vụ khó khăn. Do vậy, công tác đào tạo ban đầu, học tập liên tục trong công việc và rút kinh nghiệm, cùng với sự học hỏi từ các nguồn bên ngoài là cực kì quan trọng. Nhiều nhóm thất bại vì không thể học hỏi được cách làm mới, trong khi quá nôn nóng mong chờ kết quả tức thì khi áp dụng Scrum. Đó là điều cần hết sức cảnh giác để tránh vấp phải.

Điều cuối cùng cần lưu ý, đặc biệt đối với các dự án dài hơi nằm trong một tổ chức lớn đã có thâm niên sử dụng phương pháp quản trị dự án truyền thống là cần tới những bước chuyển đổi phù hợp. Một trong số đó là việc phân loại các Sprint thành ra những Sprint có mục tiêu khác nhau như Sprint để làm bản mẫu, Sprint để làm yêu cầu, Sprint để phân tích và thiết kế, Sprint để triển khai và Sprint để ổn định hoá. Đây có thể là cách làm giúp tổ chức loại bỏ được những khó khăn quá lớn khi phải chuyển từ hình thức truyền thống sang với một cách thức mới. Khi đội nhóm quen với Scrum rồi thì việc loại bỏ việc phân loại Sprint kia sẽ đến một cách tự nhiên.

### **Nhận diện và vượt qua các trở lực từ yếu tố văn hoá**

Một trong những lo ngại rất phổ biến ở những nơi chuẩn bị du nhập Scrum là lo ngại những đặc thù văn hoá bản địa sẽ không phù hợp với văn hoá làm việc của Scrum vốn đánh giá cao giá trị tự chủ, làm việc nhóm, hướng đến khách hàng và sẵn sàng thay đổi để thích nghi.

Chúng ta có thể vận dụng mô hình do Tiến sĩ Hofstede đề xuất để hiểu thêm về đặc thù văn hoá quốc gia và có cách để vượt qua những trở ngại thuộc phạm trù văn hoá. Tất nhiên mô hình này thật khó mà đầy đủ, nhưng trong chừng mực nào đó, nó hữu ích để việc du nhập Scrum vào một tổ chức mới thuận lợi hơn.

Bạn có thể truy xuất thông tin về mức độ của các yếu tố văn hoá theo khung Hofstede trên trang web: <https://geert-hofstede.com/countries.html>

## “Người Nhật không phù hợp với Scrum”

img837

img839

*Chúng tôi nghe câu này từ nhiều đồng nghiệp người Nhật Bản. Thật kì lạ, những tư duy nguyên thủy của Agile, Scrum chịu sự ảnh hưởng rất lớn của tư duy tinh gọn (Lean Thinking), nhưng Agile lại không nhanh lan toả ở đó. Bạn hãy nhìn vào những biểu đồ về các chỉ số trong mô hình Hofstede so sánh bốn nước: Nhật Bản, Việt Nam, Mỹ và Thụy Điển – nơi Agile rất dễ được chấp nhận, và tự mình rút ra nhận xét.*

### ⇒ Dừng và Nghĩ

**Trong một hội thảo Agile năm 2009, chuyên gia Jean Tabaka có liệt kê 12 nguyên nhân hàng đầu dẫn đến thất bại trong việc áp dụng Agile như dưới đây. Bạn có kế hoạch nào để tránh những điều này?**

### **Top 12 nguyên nhân thất bại trong việc áp dụng Scrum**

1. Sử dụng không hiệu quả hoạt động cải tiến
2. Không có khả năng lôi kéo tất cả mọi người cùng tham gia vào lập kế hoạch
3. Không chú ý tới cơ sở hạ tầng cần thiết
4. ScrumMaster tồi
5. Product Owner không giữ được sự nhất quán
6. Thất bại trong việc thúc đẩy hoạt động kiểm thử
7. Khôi phục lại khuôn mẫu trước đây
8. Chỉ quan tâm tới “cam kết sổ sách” từ phía quản lí điều hành

9. Nhóm thiếu thẩm quyền và khả năng ra quyết định
10. Không có người chịu trách nhiệm truyền đạt khi tiến hành làm việc phân tán
11. Văn hóa của tổ chức không hỗ trợ việc học tập
12. Từ chối chấp nhận một cách gay gắt

*Jean Tabaka*

## **QUẢN LÝ SỰ THAY ĐỔI**

---

Thay đổi cung cách làm việc là một thay đổi căn cơ, đòi hỏi sự lưu tâm của lãnh đạo. Phạm vi áp dụng càng lớn thì việc quản lý thay đổi càng quan trọng. Chúng ta có thể vận dụng mô hình quản trị sự thay đổi của John Kotter để quản lý sự thay đổi khi tiếp nhận Scrum vào tổ chức. Bạn cần có một kế hoạch thay đổi chu toàn, triển khai và giám sát, đánh giá liên tục và thích nghi để hướng tới mục tiêu định trước. Một kế hoạch thay đổi có thể được tổ chức dưới dạng dự án gồm một Nhóm Scrum, với “Phần tăng trưởng” chính là những thay đổi bạn muốn có ở trong tổ chức.



### **Tảng băng trôi Scrum**



Những kĩ thuật, phương pháp, công cụ được sử dụng trong Scrum chỉ là phần nổi của văn hoá Scrum.

Bên dưới đó là các giá trị văn hoá của một nhóm Scrum hiệu quả, những năng lực của từng cá nhân trong lĩnh vực của mình trong nhóm Scrum và rộng hơn là trong tổ chức.

Chỉ chăm lo cho kĩ thuật và phương pháp mà không tính tới phần tư duy, những thay đổi trong cách nghĩ và hành vi và năng lực, thì khó

lòng đạt được văn hoá Scrum ở mức độ cao.

## **TỔ CHỨC VIỆC HỌC TẬP SCRUM**

---

Không thể thành công với việc vận dụng Scrum mà thiếu việc lưu tâm đúng mức tới việc học tập ở các cấp. Kể cả ở phạm vi hẹp nhất, với mục đích là học hỏi, là việc Thí điểm Scrum cũng đòi hỏi những nỗ lực học tập nghiêm túc. Việc học hỏi Scrum và những tri thức liên quan có thể được tổ chức dưới dạng các dự án học tập ngắn hoặc dài hạn, tùy quy mô và phạm vi. Trong đó ghi rõ mục tiêu (học gì), và cách thức đạt được mục tiêu đó (học thế nào). Bạn có nhiều lựa chọn để học Scrum, từ đọc sách, xem các bài giảng trên YouTube, tới việc tham gia các khoá học tập trung hoặc các khoá học online về Scrum. Ngoài ra, bạn có thể cân nhắc mời các nhà đào tạo, huấn luyện viên Scrum chuyên nghiệp bên ngoài về để kèm cặp đội ngũ để gia tốc quá trình học hỏi và vận dụng Scrum tại công ty. Nếu công ty đã có sẵn những chuyên gia nội bộ, việc tổ chức học tập dưới dạng các seminar/workshop kết hợp với kèm cặp trong công việc (on-job- training) là một lựa chọn rất tốt để thúc đẩy học hỏi. Ở quy mô lớn hơn, bạn có thể có những chương trình học tập diện rộng, với nhiều hoạt động đa dạng hơn.

Cần lưu ý việc thành thực một kĩ năng nào đó luôn cần thời gian, tránh đốt cháy giai đoạn. Bạn cần phải trải qua các bước học hỏi, bắt chước, trước khi có thể tự mình xoay sở và giải quyết các vấn đề (theo từng bước: làm thô, làm tinh hơn, rồi tiến đến thành thạo). Do vậy, việc tránh ScrumBut không đơn thuần là vấn đề làm Scrum đúng sách, mà còn là vấn đề lĩnh hội và rèn luyện kĩ năng căn bản trước khi điều chỉnh. Không chỉ các nhà huấn luyện Scrum nói điều này, các huấn luyện viên thể thao hay các nhà giáo chuyên nghiệp cũng thấm nhuần các quy tắc học tập này.

Tùy từng mức độ và điều kiện mà một tổ chức có thể có các chiến lược và chiến thuật khác nhau đối với việc học tập Scrum nói riêng và học tập nói chung. Nhưng cần lưu ý, học tập và quản trị tri thức (cả tiềm ẩn trong các cá nhân dưới dạng kĩ năng, know-how, hay tường minh dưới dạng văn bản, quy trình...) ngày càng trở nên quan



trọng, và trở thành động lực mới của tổ chức. Do vậy việc quan tâm tới xây dựng một Tổ chức Học tập không còn là chủ đề lạ lẫm với nhiều công ty hiện đại. Các công ty có thể gây dựng tổ chức học tập theo từng cấp độ từ hẹp tới lớn. Tiếp cận từ dưới lên, công ty có thể khuyến khích và thưởng những nỗ lực học tập chủ động từ phía cá nhân thông qua các chương trình học tập tập trung hoặc học trực tuyến với chi phí hợp lý. Mức độ tiếp theo, công ty chủ động đưa việc học tập vào trong quy trình làm việc. Điều này dễ dàng thực hiện khi vận dụng Scrum do các hoạt động retrospective hay kaizen về bản chất là việc học tập nhóm được nhúng trong công việc. Tổ chức các không gian chia sẻ kiến thức, thúc đẩy nghiên cứu và nâng cao trình độ với các cộng đồng thực hành (community of practices) hay các nhóm cùng sở thích (interest groups). Bước tiếp theo là lập các đội ngũ chuyên biệt trong việc tổng hợp các tri thức từ bên trong hay bên ngoài tổ chức để thúc đẩy đổi mới một cách có hệ thống và chủ đích. Có thể cân nhắc tới việc thiết lập một hệ thống quản trị tri thức ở mức công ty và duy trì lâu dài. Cuối cùng, lãnh đạo có thể có một chiến lược nhất quán cho việc lấy tri thức làm trọng tâm trong học thuyết quản trị của công ty, và hướng sự quản trị ở các cấp sang hình thức quản trị dựa vào tri thức (knowledge-based management) nhằm biến tri thức và sự học hỏi trở thành động lực phát triển quan trọng của tổ chức.

### **⇒ Dừng và Nghĩ**

**Làm thế nào để cả nhóm vẫn học hỏi trong khi phải hoàn thành công việc được uỷ nhiệm?**

### **CÂU HỎI ỨNG DỤNG**

1. Tại sao tổ chức của bạn cần phải thay đổi? Nếu không thay đổi thì sao?
2. Tổ chức của bạn được gì khi áp dụng Scrum?
3. Liệt kê những thuận lợi khi tổ chức của bạn áp dụng Scrum về mặt văn hóa, cơ cấu tổ chức, con người, sự ủng hộ của lãnh đạo, v.v.?

4. Liệt kê những khó khăn khi bạn đưa Scrum vào tổ chức của mình về mặt văn hóa, cơ cấu tổ chức, con người, sự ủng hộ của lãnh đạo, v.v.?
5. Những ai nên tham gia nhóm tiên phong tạo sự thay đổi trong công ty bạn nếu áp dụng Scrum?
6. Bạn sẽ bắt đầu từ đâu để đưa Scrum vào tổ chức của mình?
7. Các quản lí sẽ có thay đổi công việc như thế nào nếu như tổ chức của bạn áp dụng Scrum?
8. Cần áp dụng những phương thức đào tạo, huấn luyện gì cho thành viên của nhóm/công ty của bạn khi áp dụng Scrum?
9. Định nghĩa Hoàn thành nên như thế nào để đảm bảo tiêu chuẩn chất lượng của tổ chức của bạn?
10. Có nên thay đổi hệ thống công cụ trong tổ chức để đảm bảo thông tin minh bạch nếu tổ chức của bạn áp dụng Scrum? Việc thay đổi nên như thế nào?

# 7KỸ THUẬT AGILE

Trong chương này...

- Quản lý yêu cầu người dùng linh hoạt với User Story
- Ước tính và lập kế hoạch linh hoạt
- Phát triển hướng kiểm thử (TDD)
- Phát triển hướng hành vi (BDD)
- Phát triển hướng kiểm thử chấp nhận (ATDD)
- Tái cấu trúc mã nguồn
- Lập trình cặp
- Hệ thống tích hợp liên tục
- Thiết kế bản mẫu
- Tư duy thiết kế

*Bạn bỏ lỡ 100% cơ hội ghi bàn từ những cú đánh chưa đánh.*

## **Wayne Gretzky** - Cầu thủ khúc côn cầu trên băng

Scrum được thiết kế như một khung tổng thể cho công việc phát triển sản phẩm. Để Scrum đi vào hiện thực, nó luôn cần phải được cụ thể hoá cho từng nhóm, từng bối cảnh riêng. Các chương 2, 3, 4, 5 đã tìm hiểu kỹ các thành phần của Scrum, bàn cách hiện thực hoá từng phần để một nhóm có thể đưa Scrum vào công việc phát triển sản phẩm của mình. Tuy nhiên sẽ thật thiếu sót nếu chúng ta không đề cập những kỹ thuật thường được các nhóm Scrum “cắm thêm” vào khung Scrum để ra được “công nghệ sản xuất” thực sự

của mình. Quá trình phát triển lâu dài hơn hai thập kỉ của các phương pháp Agile đã để lại cho các nhóm ngày nay hàng loạt kĩ thuật hữu dụng trong tất cả các khâu, từ thu thập và quản lí yêu cầu người dùng, lập trình, phân tích, thiết kế cho đến trải nghiệm người dùng và tự động hoá. Chương này chỉ giới thiệu điểm một số kĩ thuật. Một số phần có thể vận dụng được ngay, số khác sẽ đòi hỏi bạn phải nghiên cứu thêm để vận dụng. Khi bạn bắt đầu với Scrum, điều quan tâm đầu tiên là tổ chức lại công việc. Khi bạn bắt đầu quan tâm đến các kĩ thuật nâng cao hơn, là lúc bạn đã lên một tầm cao mới trong năng lực vận dụng Agile.



## QUẢN LÝ YÊU CẦU NGƯỜI DÙNG VỚI USER STORY

---

Các hạng mục Product Backlog mô tả yêu cầu sản phẩm, Scrum không quy định cụ thể kĩ thuật cũng như hình thức để thể hiện các yêu cầu này, tuy nhiên trong thực tế thì kĩ thuật được sử dụng phổ biến hiện nay là User Story.

User Story là một bản tóm tắt nhu cầu người dùng. Thông thường, User Story do khách hàng, hoặc đại diện của khách hàng, người thực sự hiểu nghiệp vụ và nắm bắt được chính xác yêu cầu của mình đối với nhóm phát triển. User Story không đơn thuần là công cụ requirement, mà còn là công cụ để giao tiếp, chốt kết dính và cái “phanh hãm” trong phát triển. Scrum quy định Product Owner sở hữu các story (thông qua product backlog), nhưng đó không phải công việc chỉ riêng Product Owner làm.

Nhiều nhóm dùng Index Card để viết User Story để dễ thảo luận, nhiều nhóm khác lại thích dùng các phần mềm hỗ trợ (thường là các công cụ tạo lập ProductBacklog, xem chương 5) để tạo và quản lí tập trung. Một User Story có thể được viết theo mẫu phổ biến như sau:

**Là <vai trò gì đó>, tôi muốn <làm gì đó> để <đạt được cái gì đó>**

Là khách hàng, tôi muốn xem danh sách sản phẩm mới để lựa chọn mua

Là khách hàng, tôi muốn mua hàng online

Là quản lí, tôi muốn xem được danh sách khiếu nại để có thể phản hồi

User Story được ưa chuộng nhờ tính ngắn gọn và sử dụng ngôn ngữ gần gũi với người dùng, không phải là bản mô tả đầy đủ chi tiết của tính năng người dùng, mà chủ yếu đóng vai trò liệt kê các tính năng đó. Việc tìm hiểu cụ thể về tính năng này sẽ được thực hiện thông qua trao đổi và thảo luận giữa Nhóm Phát triển, Product Owner và các bên liên quan khác.

Do đó, điều quan trọng không phải là viết ra các User Story với đầy đủ chi tiết mà là tổ chức thảo luận để làm rõ chúng. Có thể coi User Story như một nửa của việc mô tả tính năng người dùng, đây là nửa được viết ra, còn một nửa khác là thảo luận về tính năng đó.

img860

## **Làm rõ một User Story**

Thông thường, có 2 cách để chi tiết hóa một User Story.

Cách thứ nhất là phân tách một User Story thành các User Story khác nhỏ hơn.

81

## **Ai là người làm ra User Story?**

Mặc dù Product Owner là người quản lí Product Backlog (và tất cả những User Story trong đó), nhưng không có nghĩa là Product Owner là người chịu trách nhiệm viết toàn bộ User Story. Thay vào đó, trong trường hợp lý tưởng nhất, người dùng thực sự của sản phẩm sẽ viết User Story. Trong những trường hợp khác, Product Owner có thể đại diện cho người dùng, nhưng phải luôn viết User

Story với vai trò của người dùng, không phải với vai trò của Product Owner. Trong tất cả các trường hợp, các thành viên của Nhóm Phát triển đều tham gia vào việc viết User Story.



## Viết User Story khi nào?

Hoạt động này diễn ra trong suốt quá trình phát triển dự án, có nghĩa là bất cứ lúc nào các thành viên cũng có thể thêm vào các User Story mới. Tuy nhiên, ở giai đoạn ban đầu của dự án, trước khi bắt đầu Sprint đầu tiên thì Nhóm Scrum cần phải đảm bảo có một Product Backlog trước khi bắt tay vào sản xuất.

Việc này thường được tiến hành thông qua tổ chức một buổi viết User Story. Ở đó, tất cả các thành viên đều tham gia tạo ra các User Story cơ bản, đủ để sản xuất trong một thời gian. Có thể có các User Story lớn, chúng sẽ được làm mịn hơn song song với quá trình phát triển.

## INVEST – các tiêu chuẩn cho một user story tốt

INVEST là một tập các tiêu chí hướng đến việc viết các User Story tốt. Các tiêu chí của INVEST bao gồm:

**Independent – Độc lập:** Hạn chế sự phụ thuộc lẫn nhau giữa các User Story, nhờ đó mà có thể triển khai chúng theo bất cứ thứ tự nào.

**Negotiable – Đàm phán được:** User Story không phải là một bản mô tả chi tiết và chặt chẽ một tính năng của sản phẩm. Việc này cần đạt được thông qua sự cộng tác.

**Valuable – Đáng giá:** User Story cần có giá trị đối với người dùng, không phải đối với những người khác, như nhà phát triển.

**Estimable – Ước tính được:** Không cần phải ước tính chính xác, nhưng cần có một giá trị ước tính để phục vụ cho việc lập kế hoạch.

**Sized appropriately – Kích thước phù hợp:** Những User Story sắp được đưa vào sản xuất cần có kích thước nhỏ, những User Story chưa được đưa vào sản xuất trước mắt có thể có kích thước lớn hơn.

**Testable – Kiểm thử được:** Một User Story đạt được tiêu chí kiểm thử được có nghĩa là nó có đủ các chi tiết cần thiết để hiểu đúng và làm đúng.

## ƯỚC TÍNH LINH HOẠT

---

Ước tính linh hoạt là một kỹ thuật được sử dụng khá phổ biến trong các Nhóm Scrum để phục vụ cho việc lập kế hoạch tốt hơn. Việc ước tính này không hướng đến mục đích đưa ra một dự báo chính xác về công sức để hoàn thành một công việc, chẳng hạn là 2 ngày hay 18 giờ.

Thay vào đó, các Nhóm Scrum sử dụng hình thức ước tính tương đối, bằng cách so sánh độ lớn giữa các hạng mục với nhau, chẳng hạn, nếu hạng mục A cần 1 đơn vị công sức để hoàn thành và đối với hạng mục B chúng ta cần đầu tư khoảng gấp đôi lượng nỗ lực dành cho hạng mục A thì hạng mục B sẽ có giá trị ước tính là 2 đơn vị. Đơn vị ở đây có thể là thời gian tuyệt đối hoặc điểm tương đối.

Các nhóm Agile thành thực thích sử dụng đơn vị Điểm Story (story point) để xác định độ lớn cho các User Story.

### Chơi Planning Poker để ước tính nỗ lực

Ước tính linh hoạt thường được thực hiện bởi nhóm, làm việc dựa trên sự đồng thuận và cố gắng để mỗi thành viên đưa ra suy nghĩ độc lập giúp nhóm có cái nhìn đa chiều hơn. Sử dụng planning poker là một cách phổ biến để đạt được những tiêu chí trên.

## PLANNING POKER

Planning Poker là một kỹ thuật rất hiệu quả được sử dụng phổ biến để thực hiện ước tính trong Agile. Planning Poker kết hợp cả ba cách thức ước tính là dựa trên ý kiến chuyên gia, so sánh tương đối và chia nhỏ các hạng mục. Việc ước tính sử dụng Planning Poker khá nhanh chóng và mang lại kết quả đáng tin cậy. Nhóm Scrum có thể sử dụng Planning Poker để ước tính các hạng mục Product Backlog hoặc các hạng mục công việc trong Sprint Backlog.

Để thực hiện kỹ thuật này, cần phải chuẩn bị các bộ bài Planning Poker đủ cho tất cả các thành viên tham dự. Bộ bài Planning Poker có nhiều kiểu khác nhau, tuy nhiên thông thường thì một bộ poker này chứa các quân bài thuộc dãy số: 0, 1, 2, 3, 5, 8, 13, 20, 40, 100. Đây chính là dãy số Fibonacci đã được điều chỉnh một chút để phù hợp với việc ước tính. Dãy số này thể hiện các giá trị ước tính, không phụ thuộc vào đơn vị ước tính. Do đó, kỹ thuật này có thể sử dụng để ước tính với các đơn vị khác nhau, chẳng hạn là điểm tương đối hoặc giờ lý tưởng. Các nhóm có thể tự thiết kế bài cho mình hoặc là mua các loại có sẵn trên thị trường.

### **Ước tính với Planning Poker**

Bạn thực hiện ước tính cho hạng mục Product Backlog hoặc hạng mục Sprint Backlog theo các bước sau:

- Bước 1: Nhóm xác định các hạng mục sẽ được ước lượng.
- Bước 2: Chọn một hạng mục.
- Bước 3: Mỗi thành viên sẽ tự xác định điểm (point) tương ứng với nỗ lực mà nhóm cần bỏ ra để hoàn thành hạng mục đó bằng cách chọn một cây poker (quân bài) có số tương ứng. Úp cây poker đã chọn xuống trước mặt.
- Bước 4: Tất cả thành viên cùng lật cây poker mình đã chọn lên.
- Bước 5: Nếu cả nhóm cùng chọn một cây poker (cùng số điểm) thì việc ước lượng cho hạng mục đó đã xong. Ghi lại số điểm của hạng mục đó.



- Bước 6: Nếu có sự khác biệt thì các thành viên lý giải lựa chọn của mình. Thông thường thì chỉ người đưa ra ước lượng thấp nhất và cao nhất cần giải thích lựa chọn của mình. Chú ý nên giới hạn thời gian trình bày là 1 phút cho mỗi người. Sau đó mọi người thực hiện lại Bước 2 cho tới khi hết các hạng mục cần ước tính.



### *Một Nhóm Phát triển đang ước tính với Planning Poker*

**Lưu ý:** Mỗi hạng mục chỉ nên giới hạn ước lượng trong 3 lần. Tới lần thứ 3 nếu chưa đạt được sự đồng thuận nhóm nên lấy theo số đồng hoặc chọn cơ chế khác để không tốn quá nhiều thời gian. Giá trị ở đây là sự trao đổi chứ không phải các con số. Khi cần nhóm có thể mời thêm chuyên gia tham dự cùng để hỏi ý kiến khi không đạt được sự đồng thuận. Hãy nhớ, các con số đó chỉ là ƯỚC LƯỢNG mà thôi.

Cách ước lượng này giúp cho mọi thành viên trong nhóm đưa ra ý kiến một cách độc lập và từ đó họ sẽ giúp nhau tìm ra một cách hiểu đúng về hạng mục sẽ phải làm.

## **PHÁT TRIỂN HƯỚNG KIỂM THỬ (TDD)**

---

Đối với việc phát triển phần mềm, nếu ta đạt được chất lượng tự thân, chi phí cho sửa lỗi và bảo trì sẽ giảm đáng kể. Nếu ta không quan tâm tới việc đảm bảo chất lượng sản phẩm từ sớm, những lỗi kỹ thuật sẽ tích tụ và trở thành nợ kỹ thuật (technical debt), và ai đó sẽ phải trả món nợ này về sau.

Các lỗi được phát hiện càng sớm thì càng dễ để sửa chữa hơn, các lỗi phát hiện càng muộn thì chi phí sửa chữa càng tăng lên. TDD là một trong số các lựa chọn phổ biến để đảm bảo lỗi luôn được sớm phát hiện và xử lý.

Cảm giác ban đầu với hầu hết các nhóm Scrum khi nhìn vào TDD là “tốn thì giờ”, tuy nhiên, khi nhìn từ góc độ dài hạn, thì càng đầu tư

cho cái tốn thì giờ ban đầu, càng tiết kiệm về sau. Nói cách khác “muốn nhanh, cứ phải từ từ”.

img888

*Theo S. Ambler*

**TDD (Test-Driven Development)** là một kĩ thuật được giới thiệu trong XP, trong đó việc viết các bài kiểm thử được thực hiện trước khi bắt tay vào viết mã nguồn. Các vòng phát triển (bao gồm việc viết các bài kiểm thử và mã nguồn) được lặp đi lặp lại trong khoảng thời gian tương đối ngắn.

Kĩ thuật TDD được sử dụng rộng rãi bởi vì nó giúp nhóm nâng cao chất lượng tự thân của sản phẩm, giảm thiểu các lỗi có thể có ngay từ giai đoạn đầu, gia tăng độ tin cậy, tính linh hoạt và khả năng mở rộng. Ngoài ra, có một số nghiên cứu đã chỉ ra rằng các nhóm sử dụng TDD thường nâng cao được năng suất của mình so với trước đó.

### **Các bước trong TDD**

Mô hình hoạt động của TDD thường được mô tả bằng thuật ngữ “Red – Green – Refactor” nhằm thể hiện các bước thực hiện:

84

#### **Định dạng của một bài kiểm thử**

Một bài kiểm thử thường có định dạng như sau:

##### **Given**

Mô tả một trạng thái hay điều kiện của hệ thống

Ví dụ: Một danh sách các sản phẩm

##### **When**

Mô tả một hành động hoặc sự kiện xảy ra

Ví dụ: Bấm vào nút SẮP XẾP THEO GIÁ

**Then**

Mô tả kết quả đạt được

Ví dụ: Danh sách sản phẩm được sắp xếp theo giá

## **TÁI CẤU TRÚC MÃ NGUỒN (CODE REFACTORING)**

---

Tái cấu trúc mã nguồn là một kỹ thuật điều chỉnh và cải tiến mã nguồn hiện có để làm cho nó tốt hơn nhưng không thay đổi hành vi của nó đối với bên ngoài. Tái cấu trúc mã nguồn là một phần cơ hữu của TDD. Nhưng bản thân Tái cấu trúc mã nguồn cũng có thể là một phần tách biệt mà một nhóm không thực hành TDD có thể sử dụng.

Khái niệm “tốt” ở đây được hiểu giới hạn là dễ bảo trì và dễ mở rộng.

### **Dễ bảo trì**

Mã nguồn dễ bảo trì có đặc điểm là dễ đọc, dễ hiểu và dễ nắm bắt được ý đồ của người viết ra nó. Có thể làm điều này bằng nhiều cách, chẳng hạn như tách những đoạn mã lớn thành những khối mã nhỏ hơn với một mục đích cụ thể, rõ ràng, đặt tên hợp lý, sắp xếp lại các phương thức về đúng chỗ của nó, loại bỏ những ghi chú gây ra hiểu lầm...

### **Dễ mở rộng**

Mã nguồn dễ mở rộng nghĩa là có khả năng thêm các tính năng mới một cách thuận lợi. Chúng ta thường đạt được việc này bằng cách áp dụng các mẫu thiết kế phù hợp.

### **Một số kỹ thuật tái cấu trúc**

- **Trừu tượng hóa (Abstraction)**

- Bao gói các trường
- Dùng kiểu khái quát (generic)
- Thay thế type-checking bằng State/ Strategy
- Thay thế các điều kiện bằng đa hình

- **Phân tách mã**

- Tách một phương thức lớn thành những đơn vị nhỏ hơn có thể tái sử dụng được, với một nhiệm vụ đơn, rõ ràng, dễ giao tiếp.
- Tách lớp, di chuyển một phần mã nguồn sang lớp mới

- **Cải tiến tên gọi và vị trí đặt các đoạn mã**

- Chuyển phương thức hoặc trường sang vị trí phù hợp hơn ở trong một file hoặc file khác
- Đổi tên phương thức hoặc trường để thể hiện tốt hơn mục đích của phương thức hoặc trường đó
- Đẩy lên lớp cấp trên hoặc lớp cha (trong - Lập trình Hướng Đối tượng)
- Đẩy xuống lớp cấp dưới hoặc lớp con (trong Lập trình Hướng Đối tượng)

## **Thiết kế tiến hóa**

Cách tiếp cận thiết kế tiến hóa khác với cách làm truyền thống. Trước đây, trước khi bắt tay vào viết mã nguồn, thông thường các nhóm đã xây dựng sẵn một thiết kế hoàn thiện, khá chi tiết về hệ thống cũng như các tính năng của nó. Một thiết kế như vậy rất khó để chấp nhận các thay đổi có thể xảy ra trong tương lai.

Thiết kế tiến hóa (đôi khi ta gọi là thiết kế đơn giản - simple design) tức là xây dựng một bản thiết kế với định hướng sẵn sàng chấp nhận các thay đổi xảy ra. Thiết kế tiến hóa không có mục đích xây dựng một bản thiết kế đầy đủ và chi tiết ngay từ ban đầu mà chỉ dừng lại ở việc thiết kế những thứ cần thiết nhất, đủ để nhóm cùng thảo luận về cấu trúc và dùng trong thời gian trước mắt. Thiết kế sẽ luôn được điều chỉnh và thích nghi để phù hợp với từng giai đoạn phát triển.

Để duy trì một thiết kế tiến hóa, thông thường chúng ta sử dụng các mẫu thiết kế phù hợp và thường xuyên tái cấu trúc mã nguồn để gia tăng tính linh hoạt và đáp ứng được các yêu cầu hiện tại.

## **ATDD và BDD**

---

Một dạng mở rộng của TDD là Phát triển Hướng Kiểm thử Chấp nhận (Acceptance Test-Driven Development). Đây là một kỹ thuật dựa vào sự cộng tác giữa khách hàng, nhà phát triển và kiểm thử viên, trong đó toàn bộ các thành viên tập trung cộng tác để thảo luận về các tiêu chí chấp nhận, sau đó chuyển chúng thành các bài kiểm thử chấp nhận trước khi bắt tay vào viết các dòng mã cho một tính năng.

ATDD khuyến khích sự tham gia của khách hàng, sớm đưa ra các thiết kế của sản phẩm từ góc nhìn của người dùng, nâng cao chất lượng sản phẩm và giúp cho các nhà phát triển hiểu rõ hơn nghiệp vụ của sản phẩm.

Một phương pháp khác gần ý tưởng với TDD và ATDD là BDD (Behavior-Driven Development) - Phát triển Hướng Hành vi. Đây là một mô hình phát triển trong đó kết hợp các kỹ thuật của TDD với các nguyên lý cơ bản của DDD (Domain Driven Design – Thiết kế Hướng Nghiệp vụ) nhằm cung cấp các công cụ và thúc đẩy sự cộng tác của đội ngũ phát triển và những người liên quan. Một trong số các lợi ích của BDD là cung cấp cách thức để quản lý quá trình phát triển sản phẩm từ hai góc độ khác nhau đó là người dùng và kỹ thuật.

Để triển khai được BDD hay ATDD chúng ta thường phải dùng các công cụ đi kèm, có thể kể đến Cucumber, Rspec, Behat, SpecFlow, Jbehave, Lettuce (BDD); hay Thucydides, Spectacular, FitNesse, Concor- dion (ATDD).

## LẬP TRÌNH CẶP

---



*Ảnh: Wikimedia*

Lập trình cặp là hình thức cộng tác diễn ra dưới hình thức hai nhà phát triển cùng chia sẻ một vấn đề, với một máy tính, một bàn phím và với mục tiêu: giải quyết vấn đề đó. Các thành viên sử dụng sự đồng thuận, nhưng thông qua tranh luận. Tốc độ sản xuất có thể chậm hơn nhưng hiệu quả và chất lượng sẽ được nâng cao.

Mỗi cặp có hai thành viên với hai vai trò khác nhau: Người lái (Driver) và Hoa tiêu (Navigator). Người lái thường không quan tâm đến bức tranh toàn cảnh mà tập trung vào giải quyết vấn đề trước mắt. Hoa tiêu thường có xu hướng sử dụng tư duy mẫu trong giải quyết vấn đề.

Lập trình cặp là một kỹ thuật rất tốt được sử dụng nhằm nâng cao chất lượng sản phẩm, giảm thiểu lỗi ngay từ rất sớm, gia tăng cộng tác trong nhóm, thúc đẩy học tập lẫn nhau và cổ xúy sở hữu và trách nhiệm tập thể.

## LÀM BẢN MẪU

---

Làm bản mẫu (Prototyping/Mockup) là kỹ thuật sớm của sản phẩm dưới dạng một “bản mẫu” thể hiện được ý tưởng và các tính năng của sản phẩm. Việc nhanh chóng có được phiê sản phẩm sẽ giúp người dùng dễ hình dung ra sản phẩm sau khi hoàn thành, nó cũng khuyến khích sự tham gia tích cực của người dùng và nhà phát triển

từ những giai đoạn đầu tiên. Dựa trên bản sớm có các phản hồi quan trọng và nhanh chóng điều chỉnh với chi phí thấp.

Các kĩ thuật và công cụ để làm bản mẫu tùy thuộc và sản phẩm lựa chọn của từng nhóm. Các nhóm có thể làm bản mẫu trên giấy, trên các công cụ trình chiếu cho đến các công cụ làm bản mẫu hỗ trợ tương tác rất tốt.

Trong các chu trình phát triển (như Design Thinking đề cập dưới đây) các sản phẩm sáng tạo, việc làm mẫu là một phần vô cùng quan trọng.



## DESIGN THINKING

---

Design Thinking (Tư duy thiết kế) là một cách thức xây dựng sản phẩm dựa trên việc nhanh chóng đưa ra các phiên bản của sản phẩm, liên tục làm mịn và cải tiến sản phẩm dựa trên các phản hồi của người dùng thực sự. Một vòng phát triển của Design Thinking bao gồm các bước: Empathize (Thấu cảm), Define (Xác định), Ideate (Lên ý tưởng), Prototype (Làm bản mẫu) và Test (Kiểm thử).



## TÍCH HỢP LIÊN TỤC

---

Tích hợp liên tục (Continuous Integration, viết tắt CI) là kĩ thuật hợp nhất thường xuyên mã nguồn từ những nhà phát triển riêng lẻ vào một nhánh chính duy nhất.

Khi các thành viên Nhóm Phát triển làm việc trên cùng một sản phẩm, họ thường làm việc trên những phần khác nhau, do đó mỗi người tạo ra một “nhánh” khác của sản phẩm, các nhánh này thường chia sẻ chung một mã nguồn cơ sở, nếu trong quá trình phát triển có các thay đổi xảy ra với phần mã nguồn chia sẻ chung

này thì rất có thể nó sẽ ảnh hưởng đến những “nhánh” khác. Do vậy, nếu kéo dài thời gian phát triển riêng lẻ của từng nhánh thì nguy cơ xảy ra các xung đột khi các nhánh này hợp nhất lại là rất cao, và các xung đột này thường mất rất nhiều thời gian để giải quyết.

Tích hợp liên tục là kĩ thuật giải quyết được vấn đề này bằng cách rút ngắn thời gian tồn tại của các nhánh riêng lẻ, thường xuyên hợp nhất chúng lại, từ đó ngăn ngừa các xung đột cũng như sớm phát hiện ra chúng để giải quyết một cách dễ dàng và ít tốn kém hơn.

Khi chúng ta có một hệ thống CI mạnh, kết hợp với các công cụ tự động hoá cùng với các quy trình chặt chẽ kết hợp giữa bộ phận phát triển sản phẩm và bộ phận vận hành và cung cấp dịch vụ, để nhanh chóng chuyển giao liên tục (Continuous Delivery) giá trị tới người dùng cuối, là lúc chúng ta có thể tiến đến một hệ thống DevOps mạnh mẽ và hiệu quả. Đây có thể coi là một bước tiến tiếp theo trong phát triển phần mềm linh hoạt.

## **HỆ THỐNG TÍCH HỢP LIÊN TỤC**

Để triển khai Tích hợp Liên tục thì chúng ta cần xây dựng một Hệ thống Tích hợp Liên tục, trong đó bao gồm một số thành phần và cấu hình khác nhau. Đầu vào của hệ thống này chính là mã nguồn riêng lẻ từ các nhà phát triển, đầu ra của hệ thống là một phiên bản hợp nhất của sản phẩm và các báo cáo, phản hồi và lỗi phát sinh nếu có.



## **CÂU HỎI ỨNG DỤNG**

1. Tại sao Nhóm Scrum cần áp dụng thêm những kĩ thuật Agile khác?
2. Tại sao dùng User story để mô tả yêu cầu khi nó quá ngắn không thể mô tả chi tiết?



3. Ưu điểm của việc ước lượng linh hoạt so với ước lượng theo giờ?

4. Tại sao khi chơi planning poker để ước lượng nỗ lực các thành viên lại cần suy nghĩ độc lập?

5. Những giá trị mà TDD mang lại cho bạn là gì?

6. Những khó khăn và cách vượt qua khi nhóm muốn áp dụng TDD?

7. Có cần áp dụng tái cấu trúc mã nguồn để tăng khả năng bảo trì và mở rộng của phần mềm nhóm của bạn? Nếu có chiến thuật áp dụng là gì?

8. Việc áp dụng lập trình cặp giải quyết được vấn đề gì ở nhóm của bạn?

9. Làm bản mẫu có thể giúp nhóm bạn giải quyết được những vấn đề gì?

10. Nhóm bạn có đang gặp vấn đề gì với việc tích hợp? Phương án giải quyết là gì?

# 8SCRUM PHÂN TÁN

Trong chương này...

- Phân loại mức độ phân tán
- Các thử thách chính của nhóm phân tán và cách vượt qua
- Hợp Tiên-kế-hoạch-hóa
- Lưu ý cho công tác hậu cần
- Thiết lập hạ tầng

*Biết một vài trong số các câu hỏi còn tốt hơn biết tất cả các câu trả lời.*

**James Thurber**

## LÀM VIỆC PHÂN TÁN

---



Làm việc phân tán ngày càng phổ biến do nhiều ưu điểm về sự linh hoạt, chi phí và sự tối ưu nguồn lực.

Các nhóm có thể phân tán với mức độ khác nhau dựa trên hai tiêu chí chính là địa lý và thời gian:

- Ngồi cùng một chỗ nhưng phân tán về thời gian.
- Không ngồi cùng một chỗ nhưng trùng khung giờ làm việc.
- Không ngồi chung một địa điểm, không trùng thời gian làm việc.

Do đặc thù lệch nhau về thời gian làm việc, hoặc cách xa nhau về địa lí, các nhóm làm việc phân tán phải có phương án điều chỉnh cách làm việc cho phù hợp. Scrum dành cho các nhóm phân tán cũng phải điều chỉnh để thích nghi.

## Phân tán địa lí, chưa phân tán múi giờ



## Phân tán toàn bộ



## THỬ THÁCH ĐỐI VỚI NHÓM PHÂN TÁN

---

### Những khó khăn chính yếu đối với các nhóm phân tán có thể kể đến

**Giao tiếp:** do lệch múi giờ hoặc không ngồi cùng nhau nên việc giao tiếp mặt đối mặt sẽ khó khăn hơn, các nhóm sẽ phải tối ưu thời gian trùng nhau, cũng như tận dụng những công nghệ giao tiếp hiện đại để tháo gỡ các rào cản do phân tán địa lí hoặc phân tán thời gian gây ra.

**Văn hóa:** khi các nhóm làm việc ở những địa điểm cách xa nhau, sự khác biệt văn hóa cũng sẽ đáng kể hơn, các nhóm sẽ phải lưu tâm trong cách thức giao tiếp, thói quen làm việc để có thể duy trì sự cộng tác tốt.

**Minh bạch:** khi phân tán, nhiều thông tin có thể bị che khuất dễ dàng hơn, việc đảm bảo thông tin đầy đủ và minh bạch luôn gặp thử thách lớn. Các nhóm sẽ phải bỏ nhiều nỗ lực vào việc văn bản hóa ở mức độ nhất định, cập nhật các hệ thống báo cáo và đồng bộ hóa trực tuyến cũng như tận dụng các cơ hội họp trực tuyến để đảm bảo thông tin minh bạch.

## VỚI NHÓM PHÂN TÁN

---

**Đồng bộ hóa:** Do cản trở về địa lí hoặc thời gian, luôn có những độ trễ nhất định trong việc chuyển giao hoặc tiếp nối các công việc. Việc này đòi hỏi các nhóm cần đầu tư nhiều hơn vào việc lập kế hoạch chi tiết, khả năng thích ứng cũng như công cụ hỗ trợ.

**Tích hợp:** Do các tác tác được phân bố rộng khắp, nếu không có chiến lược tích hợp hệ thống thì rất khó để đảm bảo chuyển giao các gói tăng trưởng cuối mỗi Sprint đúng hạn với chất lượng cao.

**Chia sẻ kiến thức chung:** Do khó khăn về mặt giao tiếp, việc truyền giao kiến thức từ các thành viên trong nhóm sẽ trở nên khó khăn hơn, đòi hỏi đội nhóm phải bỏ công nhiều hơn vào việc chia sẻ kiến thức thông qua các hình thức tài liệu, hoặc họp trực tuyến thay vì gặp mặt nhóm trực tiếp. Điều này cũng đúng đối với hầu hết các cuộc họp khác được định nghĩa trong Scrum.

**Sự gắn bó đội ngũ:** Do cách trở về địa lí và thời gian, các nhóm phân tán ít có cơ hội chia sẻ các mối quan tâm trong/ngoài công việc. Do vậy lãnh đạo và bản thân các thành viên trong nhóm sẽ phải lưu tâm hơn trong việc gắn kết đội ngũ.

## **GIAO TIẾP HIỆU QUẢ HƠN**

---

Phải ngồi từ xa, các nhóm phân tán nên ưu tiên lựa chọn các phương án giao tiếp giàu thông tin hơn như truyền hình trực tiếp thay vì dùng các kênh giao tiếp bất đồng bộ như email hay thông qua tài liệu.

Việc sử dụng truyền hình trực tiếp có nhiều lợi thế, chúng ta không chỉ tiếp nhận thông tin qua việc báo cáo mà còn qua sắc mặt, không khí làm việc, từ đó cảm nhận những vấn đề tiềm ẩn đằng sau.

Ngoài ra, việc nhìn thấy nhau cũng tăng tính gắn kết và gia cố tính đồng đội.

Các nhóm nên kết hợp giữa trao đổi trực tiếp và gửi văn bản cùng với hệ thống lưu trữ thông tin tập trung, dễ truy xuất.

## XÂY DỰNG NIỀM TIN

---

Việc xây dựng niềm tin không phải là vấn đề riêng của nhóm dự án phân tán, nhưng ở bối cảnh phân tán, nó có thể là một vấn đề trở thành vật cản của Nhóm Phát triển. Đặc biệt là khi Product Owner và Nhóm Phát triển không cùng chỗ hoặc cùng múi giờ.

ScrumMaster và Nhóm Phát triển sẽ phải dùng nhiều nỗ lực để “xích lại gần” với Product Owner hơn, thông qua gây dựng quan hệ cá nhân, thiết lập kỉ luật làm việc, báo cáo đúng hẹn, trao đổi thông tin định kì, minh bạch; và nếu có thể, tổ chức các cuộc viếng thăm trực tiếp.

Niềm tin có thể đến từ giây phút chuyên nghiệp và thiện cảm từ ngày đầu của dự án khi các thành viên trao đổi với nhau rõ ràng về yêu cầu, tầm nhìn và bối cảnh của dự án, cũng như những ràng buộc, tiêu chuẩn và lưu ý khi làm việc trong dự án. Giao tiếp tích cực và chuyên nghiệp trong giai đoạn đầu là cực kì quan trọng.

Thêm nữa, niềm tin cũng đến từ những cam kết làm việc, sự đúng hẹn trong làm việc và chuyển giao sản phẩm, cũng như việc thông tin kịp thời các vấn đề gặp phải.

Việc xây dựng niềm tin giữa các bên trong một nhóm phân tán nên là một ưu tiên quan trọng của ScrumMaster.

⇒ **Dừng và Nghĩ**

**Hãy liệt kê 10 hành vi phá hủy niềm tin trong một nhóm.**

## HỌP TIỀN-KẾ-HOẠCH-HÓA

---

Để ứng phó với tình trạng thiếu thông tin cho lập kế hoạch, các nhóm phân tán thường tổ chức các cuộc họp trước khi họp lập kế hoạch gọi là họp Tiền-kế-hoạch-hóa để:

- Chuẩn bị cho cuộc họp lập kế hoạch
- Chủ yếu chuẩn bị cho phần WHAT của cuộc họp Lập Kế hoạch Sprint

Cuộc họp chỉ gồm một số đại diện cần thiết của nhóm Scrum (Product Owner, ScrumMaster và một vài Developer ở các nơi khác nhau), không nhất thiết phải diễn ra ngay trước buổi Lập kế hoạch Sprint.

## LƯU Ý CÔNG TÁC HẬU CẦN

---

Để đảm bảo các nhóm phân tán hoạt động hiệu quả, ScrumMaster có thể kết hợp với các bộ phận hành chính/quản trị để chăm lo chu đáo các khâu hậu cần, bao gồm lập thời khóa biểu, lịch họp, chuẩn bị phòng ốc và các thiết bị họp, các phần mềm hỗ trợ, cũng như các tài liệu và tạo tác liên quan.

Rất nhiều cuộc họp từ xa bị gián đoạn chỉ vì những hỏng hóc kỹ thuật có thể tránh được. Hậu cần chu đáo đồng nghĩa với việc ít thiệt hại nhất, tiết kiệm thời gian nhất và làm việc hiệu quả hơn.



## CÔNG CỤ CỘNG TÁC PHÂN TÁN

---



Bạn cần thiết lập một hệ thống công cụ tốt để có thể làm việc từ xa mà vẫn đảm bảo thông tin minh bạch, tiến độ được cập nhật và cộng tác hiệu quả. Chúng ta có thể kể đến các công cụ căn bản như:

- Giao tiếp: Các công cụ hỗ trợ hội thoại trực tuyến, truyền hình trực tuyến, hội thoại nhóm, chat trực tuyến miễn phí như Google, Slack, Skype, Hang-out hoặc những giải pháp teleconference chất lượng cao.

- Workflow/PMS: Có hàng loạt phần mềm quản trị dự án và luồng công việc để chúng ta sử dụng, có thể miễn phí hoặc trả phí như Trello, Redmine, Asana, JIRA, IBM Rational, Microsoft TFS... Hãy thiết lập một hệ thống tốt, viết quy tắc sử dụng chu đáo và huấn luyện đầy đủ, bạn sẽ làm việc như một thành viên của nhóm, dù cách xa cả lục địa.
- Hệ thống Build/CI tự động hóa để đồng bộ và tích hợp.
- Hệ thống lưu trữ file share/Cloud storage đảm bảo những tài liệu/tài nguyên cần thiết cho cộng tác luôn sẵn sàng để sử dụng.
- Một hệ thống quản trị tri thức (KMS), đơn giản như một trang wiki, hoặc một giải pháp KMS hoàn chỉnh sẽ giúp bạn chia sẻ được những bài học thành công/thất bại, cũng như cung cấp những tài liệu huấn luyện cơ bản cho các thành viên trong đội dự án.

## **CÂU HỎI ỨNG DỤNG**

1. Phân tích những vấn đề mà nhóm gặp phải do tình trạng phân tán đang gây ra?
2. Các kênh giao tiếp nào có thể áp dụng để giảm thiểu những vấn đề hiện có do tình trạng phân tán.
3. Nếu có sự tin tưởng lẫn nhau thì các thành viên (khách hàng, Product Owner, nhà phát triển, v.v.) sẽ giải quyết được những vấn đề gì?
4. Tìm giải pháp tăng sự tin tưởng cho nhóm của bạn.
5. Tìm những thay đổi về hậu cần nào có thể thay đổi để giải những vấn đề do tình trạng phân tán?
6. Phân tích những ưu và nhược điểm của các công cụ hiện thời cho cộng tác.
7. Tìm các phương án áp dụng các công cụ hiện thời và mới để giảm thiểu các vấn đề do phân tán.

8. Liệt kê các vấn đề của buổi họp kế hoạch trong nhóm của bạn?

9. Đưa giải pháp cho các vấn đề của buổi lập kế hoạch?



# 9SCRUM VỚI QUY MÔ LỚN

Trong chương này...

- Các vấn đề của quy mô lớn
- Scrum ở quy mô lớn với Nexus framework
- Các vai trò mới trong Nexus
- Các cơ chế cộng tác mới trong Nexus
- Chuẩn bị về công cụ và hạ tầng

*Không có gì bất biến, trừ sự thay đổi.*

## Heraclitus

### THỬ THÁCH CHÍNH ĐỐI VỚI QUY MÔ LỚN

---

Chúng ta đã thảo luận ở chương 6 về lộ trình vận dụng Scrum tại công ty. Phạm vi áp dụng có thể được mở rộng từ việc làm thí điểm (Pilot) đến quản trị dự án, nâng lên mức độ toàn bộ khu vực sản xuất, và mức cuối là toàn bộ công ty. Việc vận dụng Scrum ở quy mô lớn thường được đặt ra khi tổ chức của bạn đã thành thực Scrum nhất định ở quy mô nhỏ. Khi đó bạn cần có kế hoạch chu toàn để mở rộng phạm vi áp dụng Scrum.

Đối với các dự án lớn cần huy động nhiều hơn một nhóm Scrum hạt nhân, thì các thử thách chính nằm ở các khâu:

- **Tích hợp:** Làm sao để các nhóm làm việc trên các hạng mục khác nhau có thể tích hợp thành công để chuyển giao đều đặn gói tăng trưởng vào cuối mỗi Sprint? Nếu như vấn đề tích hợp cũng là cản trở của các nhóm phân tán (quy mô nhỏ), thì vấn đề tích hợp đối với

dự án lớn gồm nhiều nhóm Scrum cùng làm việc vừa là vấn đề phân tán, vừa là vấn đề quy mô.

- **Đồng bộ hóa:** Tương tự như trong các nhóm phân tán, Scrum ở quy mô lớn vấp phải vấn đề đồng bộ hoá công việc của các nhóm với nhau, sao cho vẫn có thể đảm bảo các ưu tiên và khả năng tích hợp.
- **Chuẩn hóa:** Không giống như các nhóm nhỏ, mọi vấn đề dễ giải quyết thông qua họp nhanh, trao đổi mặt đối mặt; các nhóm lớn cần phải có quy chuẩn để cùng làm việc, dễ dàng đồng bộ hoá, tích hợp và cộng tác.
- **Minh bạch hóa:** Dự án càng lớn, càng phức tạp thì nguy cơ để sót thông tin, hoặc thông tin sai lệch càng cao.
- **Phân tán:** Một dự án lớn gồm nhiều nhóm Scrum về bản chất là một nhóm lớn phân tán. Vì thế tất cả những khó khăn mà nhóm phân tán gặp phải, thì cũng sẽ có nguy cơ hiện diện trong nhóm Scrum ở quy mô lớn.
- **Xu hướng quay về với command-and-control:** Do những nguy cơ kể trên, cùng với nỗi sợ mất kiểm soát của các cấp lãnh đạo, các dự án lớn có xu hướng quay trở lại lối quản lí ra lệnh/ kiểm soát.

Các vấn đề và chiến lược thích ứng của nhóm phân tán được đề cập tới trong chương 8 hầu như vẫn giữ nguyên giá trị trong bối cảnh làm Scrum ở quy mô lớn. Ngoài ra, những đặc thù của quy mô lớn đặt ra các vấn đề quản lí mới, gồm:

- **Yêu cầu:** các yêu cầu cần được quản lí tập trung, cập nhật liên tục, chia sẻ và minh bạch để các nhóm dễ bề đồng bộ hoá công việc phát triển.
- **Kiến thức và hiểu biết chung:** thông qua các cơ chế họp và chia sẻ thông tin tập trung, các kiến thức về sản phẩm/công nghệ/tiêu chuẩn sẽ phải được chia sẻ để tăng khả năng thành công trong hợp

tác. Quy mô dự án càng lớn và kéo dài, thì việc quản trị tri thức càng trở nên quan trọng.

- **Các tạo tác:** Tất cả phần mềm, tài liệu kiểm thử, kịch bản kiểm thử... cần được quản lý tập trung, minh bạch và luôn cập nhật.

## KHUNG NEXUS CHO SCRUM QUY MÔ LỚN

---

Hiện nay có nhiều cách tiếp cận đối với việc áp dụng Agile ở quy mô lớn để giải quyết các vấn đề kể trên. Chúng ta có thể nhắc đến những phương pháp phổ biến như SAFe, LeSS, hay một biện pháp đã được biết đến từ lâu trong cộng đồng những người thực hành Scrum là Scrum of Scrums.

Gần đây, Ken Schwaber và Scrum.org đã cho ra mắt một khung làm việc cho các sản phẩm/dự án có quy mô lớn từ 3-9 nhóm Scrum. Khung làm việc này được gọi là Nexus. Và bản hướng dẫn định nghĩa chi tiết cách làm được đăng tải trên Scrum.org.

Chúng ta sẽ cùng tìm hiểu phương pháp này, do nó vận dụng triệt để cách làm của Scrum để thích ứng với các vấn đề của quy mô lớn.



*Nexus Framework, Scrum.org*

### Cùng một sản phẩm, nhiều nhóm Scrum

Toàn bộ yêu cầu của sản phẩm được quản lý tập trung trong một Product Backlog duy nhất.

Các nhóm sẽ tự tổ chức để hiện thực hoá các hạng mục có độ ưu tiên cao hơn, chuyển giao các gói tăng trưởng nhỏ, và tích hợp chúng lại để đảm bảo các gói tăng trưởng tổng thể (hay Increment tích hợp) luôn được chuyển giao cuối mỗi Sprint.



## CƠ CHẾ CỘNG TÁC TRONG NEXUS

---

### Thực thể mới: Nexus

Trong dự án lớn, thực thể lớn nhất là Nexus, bao gồm tất cả các Nhóm Scrum (từ 3 đến 9 nhóm) thành viên, cùng làm việc trên một sản phẩm, chia sẻ chung một Product Backlog duy nhất, cùng duy trì một Nexus Sprint Backlog để quản lý công việc cần làm, thực hiện Họp ở mức Nexus và tạo Increment tích hợp ở cuối mỗi Sprint.



### Vai trò mới: Đội tích hợp Nexus

Để quản lý chủ động các công việc liên quan đến tích hợp, Đội tích hợp Nexus được thành lập với các thành viên: Product Owner, ScrumMaster, các thành viên của đội tích hợp.

Thành viên có thể thay đổi, lấy từ các nhóm Scrum, kể cả ScrumMaster của Đội tích hợp Nexus cũng có thể kiêm vai trò ScrumMaster của một Nhóm Scrum thành viên. Tuy nhiên cần phân biệt rõ việc đóng vai trò ScrumMaster của Đội tích hợp Nexus thì nhiệm vụ quan trọng là phải đảm bảo thành viên thấu hiểu Nexus và vai trò của Đội tích hợp Nexus trong toàn bộ hệ thống Nexus.

Công việc của Đội tích hợp Nexus: xử lý tất cả các vấn đề liên quan đến tích hợp như lập kế hoạch tích hợp, công cụ tích hợp, công việc hằng ngày. Đảm bảo Minh bạch – Thanh tra – Thích nghi mức độ Nexus. Trong những điều kiện thích hợp, thành viên Đội Tích hợp Nexus phải cố vấn cho các Nhóm Scrum thành viên cách thức phát triển sao cho sản phẩm của họ tích hợp thuận lợi với phần còn lại của sản phẩm đang dần được làm ra bởi các nhóm khác trong Nexus.

### Các Sự kiện Nexus

Giống như Scrum, Nexus cũng có những cuộc họp tiêu chuẩn như Lập kế hoạch Nexus Sprint, Nexus Daily Scrum, Sơ kết Nexus Sprint, Họp cải tiến Nexus Sprint. Nhưng tất cả các cuộc họp kiểu này diễn ra ở quy mô liên nhóm, với các thành viên là đại diện từ các Nhóm Scrum thành viên, và không trùng với những cuộc họp đó ở cấp độ nhóm Scrum thành viên của Nexus. Nexus có thể hiểu là Scrum của các Scrum.

Các sự kiện Nexus không giam chân vào các công việc của các nhóm thành viên, nó tập trung vào khía cạnh chia sẻ hiểu biết chung, bạch hóa thông tin và đồng bộ hóa công việc của toàn bộ Nexus.



## CÔNG CỤ VÀ QUY TRÌNH

---

Để đảm bảo minh bạch thông tin, toàn bộ Nexus sử dụng một Nexus Product Backlog và một Nexus Sprint Backlog chung. Các nhóm có thể thiết lập riêng các tạo tác này ở cấp độ nhóm. Việc này có thể đạt được bằng các công cụ đơn giản như các bảng trắng và giấy dán, nhưng cũng có thể được thực hiện thông qua việc thiết lập các hệ quản trị dự án tập trung và các công cụ quản lý luồng công việc.

Giống như các nhóm phân tán, các Nexus cần có một hệ thống các công cụ quản trị tri thức, quản trị tài liệu và các tạo tác chung, các công cụ giao tiếp và tổ chức họp cũng như hệ thống build/tích hợp liên tục.

## CÂU HỎI ỨNG DỤNG

1. Theo bạn, đâu là những khó khăn chung khi áp dụng Scrum ở quy mô lớn và Scrum tiêu chuẩn?
2. So sánh những điểm chung và khác nhau giữa Nexus và Scrum?

3. Scrum cho nhóm phân tán có những lợi thế và khó khăn gì so với Nexus?

4. Khi triển khai Nexus thì những người nào sẽ tạo ra Định nghĩa Hoàn thành cho việc tích hợp? Tại sao?

5. Nhiều người nói khi dùng Nexus sẽ không cần thêm các phương pháp, kĩ thuật khác vì các nhóm vẫn thực hành Scrum, bạn nghĩ sao về nhận định này?

6. Để giúp các nhóm trong Nexus làm việc tốt cần hạn chế sự phụ thuộc giữa các hạng mục Product Backlog mà các nhóm chọn cho Sprint của mình, ngoài ra còn cần thêm những biện pháp nào nữa không?

7. Có cần Sơ kết Sprint cho chung cho tất cả các nhóm trong Nexus hay không? Tại sao?

8. Khi thêm một hạng mục vào Product Backlog trước buổi Lập kế hoạch Nexus Sprint có ảnh hưởng đến các nhóm không?

9. Tại sao trong Nexus hoạt động làm mịn Product Backlog được làm thường xuyên hơn trong Scrum tiêu chuẩn?

10. Tại sao công cụ tích hợp tự động là rất cần thiết khi triển khai Nexus?

# Tài liệu tham khảo và đọc thêm

## Sách

- Beck, K. (2000). Extreme Programming explained: Embrace change. Addison-Wesley Professional.
- Cohn, M. (2004). User stories applied: For agile software development. Addison-Wesley Professional.
- Cohn, M. (2010). Succeeding with Agile: Software development using Scrum. Pearson Education.
- Deemer, P., Bene eld, G., Larman, C., & Vodde, B. (2012). Scrum Căn bản - Phiên bản 2.0 (bản dịch tiếng Việt của Học viện Agile).
- Deemer, P., Bene eld, G., Larman, C., & Vodde, B. (2012). The Scrum Primer: An introduction to agile project management with scrum. Good Agile version 2.0.
- Deemer, P., Hazrati, N. K. V., & Bene eld, G. B. R. (2013). The Distributed Scrum Primer.
- Kniberg, H. (2015). Scrum và XP từ các chiến hào. Bản dịch tiếng Việt của HanoiScrum (HanoiScrum.net)
- Kotter, J. P. (1996). Leading change. Harvard Business Press.
- Ries, E. (2011). The Lean Startup. New York: Crown Business.
- Schwaber, K. (2004). Agile project management with Scrum. Microsoft Press.
- Schwaber, K. & Sutherland, J (2013). Hướng dẫn Scrum: Các quy tắc của trò chơi (bản dịch tiếng Việt của Học viện Agile).

- Schwaber, K., & Sutherland, J. (2012). Software in 30 days: How Agile managers beat the odds, delight their customers, and leave competitors in the dust. John Wiley & Sons.
- Schwaber, K., & Sutherland, J. (2015). Nexus Guide: The Definitive Guide to Nexus: The exoskeleton of scaled Scrum development.
- Schwaber, K., & Sutherland, J. (2016). The Scrum guide-the definitive guide to Scrum: The rules of the game.

### **Bài báo**

- Snowden, D. J., & Boone, M. E. (2007). A leader's framework for decision making. Harvard Business Review, 85(11), 68.
- Sutherland, J. (2010). Agile principles and values. Link: <http://msdn.microsoft.com/en-us/library/dd997578.aspx>.
- Sutherland, J., Schoonheim, G., Rustenburg, E., & Rijk, M. (2008, August). Fully distributed Scrum: The secret sauce for hyperproductive offshored development teams. In Agile, 2008. AGILE'08. Conference (pp. 339-344). IEEE.
- Takeuchi, H., & Nonaka, I. (1986). The new new product development game. Harvard Business Review, 64(1), 137-146.

### **Báo cáo**

- Báo cáo CHAOs của Standish Group năm 2015.
- Báo cáo “10th Annual State of Agile” của VersionOne năm 2015.

### **Trang web**

- Tuyên ngôn Agile (<http://agilemanifesto.org>)
- Hiệp hội Scrum thế giới ([www.scrumalliance.org](http://www.scrumalliance.org))
- Hiệp hội Agile thế giới ([www.agilealliance.org](http://www.agilealliance.org))



- Agile Vietnam ([www.agilevietnam.org](http://www.agilevietnam.org))
- Chỉ số văn hoá Hofstede (<https://www.geert-hofstede.com>)
- eduScrum (<http://eduscrum.nl>)
- Tái cấu trúc mã nguồn ( <http://refactoring.com> )
- Blog AgileBreakfast của Học viện Agile ([www.hocvienagile.com/agilebreakfast](http://www.hocvienagile.com/agilebreakfast) )

# Phụ lục 1 THUẬT NGỮ

## **Agile (Agile)**

Agile là một tập hợp các nguyên lý dành cho phát triển phần mềm, trong đó khuyến khích việc lập kế hoạch thích ứng, phát triển tăng dần, chuyển giao sớm, và cải tiến liên tục. Agile cũng chủ trương thích ứng nhanh chóng với các thay đổi. Những nguyên lý này được chia sẻ trong Tuyên ngôn Phát triển Phần mềm Linh hoạt (Manifesto for Agile Software Development) và 12 Nguyên lý phía sau.

## **Acceptance Test-Driven Development (Phát triển Hướng Kiểm thử Chấp nhận)**

Được gọi tắt là ATDD, Phát triển Hướng Kiểm thử Chấp nhận là một phương pháp phát triển tương tự như TDD. Tuy nhiên ATDD sử dụng những kiểm thử chấp nhận tự động. Trong trường hợp lý tưởng thì khách hàng, Product Owner hoặc chuyên gia nghiệp vụ là người viết các kiểm thử chấp nhận và Nhóm Phát triển chỉ cần vượt qua các kiểm thử này để hoàn thành sản phẩm.

## **Behaviour-Driven Development (Phát triển Hướng Hành vi)**

Phát triển Hướng Hành vi (BDD) là phương pháp phát triển phần mềm kế thừa từ TDD và ATDD. BDD thêm vào những phương pháp sau:

Áp dụng kỹ thuật 5WHYs vào mỗi user story để biết được giá trị kinh doanh của mỗi user story.

“Tư duy từ ngoài vào”, tức là chỉ cài đặt những hành vi mang lại giá trị kinh doanh để giảm thiểu lãng phí.

Mô tả hành vi theo một loại ngôn ngữ mà cả chuyên gia nghiệp vụ, kiểm thử viên và lập trình viên đều có thể giao tiếp được với nhau.

## **Burndown (Burndown)**

Xu hướng của công việc còn lại xuyên suốt thời gian của một Sprint, một bản phát hành, hoặc một sản phẩm. Nguồn dữ liệu thô được lấy từ Sprint Backlog và Product Backlog, khối lượng công việc còn lại được thể hiện ở trục tung còn thời gian (các ngày của Sprint, hoặc các Sprint) được thể hiện ở trục hoành.

Thuật ngữ này thường đi kèm với từ chart (biểu đồ), trong Scrum có hai loại biểu đồ burn down được sử dụng phổ biến là Sprint Burndown và Release Burndown.

## **Continuous Integration (Tích hợp Liên tục)**

Thuật ngữ này có tên ngắn gọn là CI. Đây là phương pháp mà nhóm liên tục tích hợp các phần của sản phẩm chứ không đợi tới cuối. CI giúp giảm thiểu thời gian, công sức và rủi ro khi tích hợp ứng dụng diễn ra ở giai đoạn cuối của chu trình phát triển, tạo ra khả năng phát hành liên tục.

## **Continuous Deployment (Triển khai Liên tục)**

Triển khai Liên tục (CD) được coi như phần mở rộng của Tích hợp Liên tục (CI). Kỹ thuật này được sử dụng để giảm thiểu thời gian chờ, khoảng thời gian mà người dùng thật sự được sử dụng phần mềm với những mã nguồn mới nhất được viết bởi nhà phát triển. Giống như CI, kỹ thuật này được triển khai với sự trợ giúp của một loạt các công cụ tự động.

## **Cross-functional (Liên chức năng)**

Liên chức năng là đặc điểm của một nhóm có đầy đủ những chuyên môn khác nhau để có khả năng đạt được một mục tiêu chung.

Một nhóm liên-chức năng có nghĩa là được trang bị đầy đủ tất cả các kỹ năng cần thiết để hoàn thành toàn bộ công việc và tạo ra phần tăng trưởng chuyển giao được cuối mỗi Sprint mà không cần sự trợ giúp từ bên ngoài. Điều này không có nghĩa là mỗi thành viên

đều phải biết hết tất cả các kĩ năng, mà họ có thể chỉ có một số kĩ năng nhất định nhưng bổ sung đầy đủ cho nhau để tổng thể nhóm có được tất cả các kĩ năng cần thiết.

## **Daily Scrum (Scrum Hằng ngày)**

Một sự kiện Scrum. Đây là buổi trao đổi ngắn của từng Nhóm Phát triển diễn ra hằng ngày để các thành viên Nhóm thanh tra công việc của mình, đồng bộ công việc và tiến độ, trình bày các trở ngại gặp phải để ScrumMaster có thể loại bỏ chúng. Có thể diễn ra các cuộc thảo luận ngay sau đó để thích nghi phần công việc tiếp theo nhằm tối ưu hóa Sprint.

## **Development Team (Nhóm Phát triển)**

Một trong ba vai trò trong Nhóm Scrum gồm ScrumMaster, Product Owner và Development Team.

Nhóm Phát triển gồm tất cả những người đủ năng lực để chuyển giao phần tăng trưởng chất lượng (Increment) cuối mỗi Sprint.

Trong Scrum, Nhóm Phát triển là liên chức năng.

## **DevOps (DevOps)**

Khi triển khai kĩ thuật Triển khai Liên tục (CD) sẽ dẫn đến nhu cầu cộng tác chặt chẽ giữa nhóm phát triển (development) và nhóm vận hành (operation). Nhu cầu này đã cho ra đời các công cụ tự động để tích hợp các công đoạn và cộng tác giữa các bên liên quan. Có một công thức nói lên tính chất này:

DevOps = Development + Operation.

## **Done (Hoàn thành)**

Khái niệm hoàn thành được thống nhất bởi tất cả các bên và tuân thủ các tiêu chuẩn, quy ước, và chỉ dẫn của tổ chức. Khi một thứ được coi là “hoàn thành” trong buổi Sơ kết Sprint thì nó phải tuân thủ được định nghĩa đã thống nhất này.

Các nhóm Scrum thường phải viết Định nghĩa Hoàn thành (Definition of Done) trước khi bắt đầu công việc.

### **Estimated Work Remaining (Công việc Còn lại Được ước lượng)**

Khối lượng công việc còn lại mà một thành viên Nhóm Phát triển ước tính dành để làm việc trên một hạng mục nào đó. Ước tính này được cập nhật vào mỗi cuối ngày khi hạng mục đó đã được triển khai. Con số này là tổng lượng công việc còn lại được ước tính chứ không phụ thuộc vào số người sẽ tham gia làm việc.

### **eXtreme Programming (Lập trình Cực hạn)**

Extreme Programming (XP) là một phương pháp phát triển phần mềm hướng đến việc nâng cao chất lượng phần mềm và khả năng đáp ứng với thay đổi yêu cầu người dùng. XP là một trong các phương pháp thuộc họ Agile, phương pháp này chủ trương đưa ra các bản phát hành thường xuyên thông qua các chu trình phát triển ngắn. Việc này là để nâng cao năng suất và tạo ra những thời điểm để tiếp nhận những yêu cầu người dùng mới.

### **Increment (Phần tăng trưởng)**

Tính năng của sản phẩm được Nhóm Phát triển xây dựng trong từng Sprint có khả năng chuyển giao được hoặc sử dụng bởi các bên liên quan của Product Owner.

Increment of Potentially Shippable Product Functionality (Phần tăng trưởng Tính năng Sản phẩm Có khả năng Chuyển giao được)

Một lát cắt hoàn chỉnh của tổng thể sản phẩm hoặc hệ thống mà Product Owner hoặc các bên liên quan có thể sử dụng nếu họ muốn triển khai.

Thuật ngữ này còn được gọi tắt là Increment.

### **Kanban (Kanban)**

Kanban là một phương pháp Agile dựa trên Phương thức Sản xuất Toyota với bốn nguyên lý:

- Trực quan hóa công việc
- Giới hạn công việc đang làm (Limit WIP – Limit Work In Progress)
- Tập trung vào luồng làm việc
- Cải tiến liên tục

Kanban đã không chỉ được ứng dụng trong làm việc nhóm mà còn cho cả quản lý công việc cá nhân với tên gọi Kanban cho cá nhân.

Cần phân biệt Kanban với tư cách là một phương pháp, và Kanban như là một bảng công việc.

Kanban áp dụng cho công việc cá nhân có tên riêng là Kanban Cá nhân (Personal Kanban) với chỉ gồm 2 nguyên lý trên cùng.

### **Lean Software Development (Phát triển Phần mềm Tinh gọn)**

Phát triển Phần mềm Tinh gọn (LSD) là hình thức áp dụng Lean Manufacturing (Sản xuất Tinh gọn) cho lĩnh vực phát triển phần mềm.

Thuận ngữ Lean Software Development có nguồn gốc từ một cuốn sách cùng tên của Mary Poppendieck và Tom Poppendieck. Cuốn sách diễn dịch lại tư duy Tinh gọn với ý nghĩa mới kèm theo các công cụ hữu hiệu để triển khai thực tiễn.

### **Manifesto for Agile Software Development (Tuyên ngôn Phát triển Phần mềm Linh hoạt)**

Tháng 2 năm 2001, 17 chuyên gia là đại diện cho những phương pháp phát triển phần mềm đã gặp nhau trong một cuộc hội thảo tại Utah, Hoa Kỳ. Hội thảo đã đi đến thống nhất về quan điểm chung giữa các phương pháp phát triển phần mềm đang nở rộ lúc đó và cho ra đời một tài liệu được gọi là: Tuyên ngôn Phát triển Phần

mềm Linh hoạt kèm với 12 nguyên lý phía sau. Đây chính là thời điểm mà thuật ngữ Agile được sử dụng hiện nay ra đời, mặc dù các phương pháp riêng lẻ thì đã có trước đó.

Thuật ngữ này thường được gọi tắt trong tiếng Việt là Tuyên ngôn Agile.

## **Product Backlog (Product Backlog)**

Một danh sách ưu tiên chứa các yêu cầu với thời gian ước tính cần thiết để phát triển thành tính năng sản phẩm. Các hạng mục ở phía trên của Product Backlog có độ ưu tiên cao hơn thì được ước tính chính xác hơn. Danh sách này tiến hóa và thay đổi khi các điều kiện kinh doanh hoặc công nghệ thay đổi.

## **Product Backlog Item (Hạng mục Product Backlog)**

Các yêu cầu chức năng, phi chức năng, và các vấn đề được đánh giá độ ưu tiên theo mức độ quan trọng đối với nghiệp vụ kinh doanh và sự phụ thuộc lẫn nhau, chúng cũng được ước lượng. Độ chính xác của việc ước tính phụ thuộc vào độ ưu tiên và mức chi tiết của hạng mục Product Backlog, các hạng mục có độ ưu tiên cao nhất có thể được lựa chọn cho Sprint tiếp theo sẽ khá chi tiết và chính xác.

## **Product Backlog Refinement (Làm mịn Product Backlog)**

Việc làm mịn Product Backlog là hoạt động thêm vào các chi tiết, ước lượng, và trình tự của các hạng mục trong Product Backlog. Đây là quá trình liên tục, theo đó Product Owner và Nhóm Phát triển thảo luận về các chi tiết của từng hạng mục. Trong suốt quá trình làm mịn này, các hạng mục liên tục được xem xét và rà soát cẩn thận.

## **Product Owner (Product Owner)**

Là người chịu trách nhiệm quản lý Product Backlog với mục tiêu tối ưu hóa giá trị của sản phẩm. Product Owner cũng là người đại diện

cho quyền lợi của tất cả những ai có liên quan đến dự án và sản phẩm được tạo ra.

## **Scrum (Scrum)**

Không phải là một từ viết tắt mà là những cơ chế trong trò chơi bóng bầu dục nhằm đưa quả bóng đã ra ngoài quay trở lại sân.

Các tác giả của Scrum đã lấy ý tưởng đó để đặt tên cho một phương pháp Agile phổ biến nhất hiện nay.

## **Scrum Artifacts (Các tạo tác Scrum)**

Các tạo tác trong Scrum là những công cụ hoặc kết quả được tạo ra và sử dụng trong quá trình vận hành Scrum. Các tạo tác trong Scrum bao gồm:

- Product Backlog
- Sprint Backlog
- Phần tăng trưởng
- Scrumban (Scrumban)

Scrumban là phương pháp có sự kết hợp giữa Scrum và Kanban. Phương pháp này bao gồm toàn bộ những ưu điểm của cả Scrum và Kanban. Scrumban khuyến khích các đội phải liên tục cải tiến quy trình thông qua cơ chế của Kanban.

Scrumban được xem là một quy trình đơn giản dùng trong quản lý những dự án phức tạp, phương pháp này được khuyến nghị dùng trong các dự án thiết kế website, bảo trì và phát triển phần mềm.

## **ScrumMaster (ScrumMaster)**

Là người chịu trách nhiệm về quy trình Scrum, đảm bảo nó được triển khai đúng và tối ưu hóa các lợi ích mà nó mang lại.



## **Self-organized (Tự tổ chức)**

Tự tổ chức (self-organized) có nghĩa là nhóm có khả năng và thẩm quyền để định hướng và đưa ra các quyết định trong quá trình sản xuất. Điều đó cũng có nghĩa là nhóm có toàn quyền trong việc lựa chọn công cụ, kỹ thuật và cách thức để hoàn thành công việc. Không ai có quyền yêu cầu nhóm tự tổ chức phải làm theo một cách nhất định để hoàn thành mục tiêu của họ. Song song với đó, tính cam kết và trách nhiệm của các thành viên trong nhóm cũng cao hơn rất nhiều so với khi nhóm được tổ chức theo mô hình ra lệnh-điều khiển.

## **Sprint (Sprint)**

Một phân đoạn, hoặc một chu trình lặp với các công việc giống nhau để sản xuất ra phần tăng trưởng của sản phẩm hoặc hệ thống. Sprint không được phép kéo dài hơn một tháng và thường thì dài hơn một tuần. Độ dài của Sprint được cố định trong suốt quá trình phát triển, và tất cả các nhóm cùng tham gia làm việc trên một sản phẩm hoặc hệ thống thì sử dụng chung chu trình có cùng độ dài.

## **Sprint Backlog (Sprint Backlog)**

Danh sách công việc của Nhóm Phát triển trong một Sprint. Nó thường được phân tách thành một tập các nhiệm vụ chi tiết hơn. Danh sách này tiến hóa trong suốt buổi Lập kế hoạch Sprint và có thể được Nhóm cập nhật trong suốt Sprint thông qua việc loại bỏ một số hạng mục hoặc thêm các công việc mới khi cần thiết. Từng hạng mục công việc trong Sprint Backlog được theo dõi trong suốt Sprint và hiển thị khối lượng công việc ước tính còn lại.

## **Sprint Backlog Task (Công việc trong Sprint Backlog)**

Là một trong số các công việc mà Nhóm Phát triển hoặc thành viên của Nhóm xác định là cần phải thực hiện để biến các hạng mục Product Backlog thành chức năng hệ thống. Thường được gọi ngắn gọn là task (công việc, tác vụ, nhiệm vụ) hoặc đôi khi là Sprint Backlog Item (hạng mục của Sprint Backlog)

## **Sprint Goal (Mục tiêu Sprint)**

Mục tiêu Sprint là bản tóm tắt những mục tiêu của Sprint, lý tưởng nhất đây là một nội dung ngắn gọn cô đọng.

## **Sprint Planning (Lập kế hoạch Sprint)**

Một sự kiện được đóng khung để khởi động một Sprint. Sự kiện này được chia làm hai phần. Trong phần đầu tiên, Product Owner trình bày với Nhóm các hạng mục Product Backlog có độ ưu tiên cao nhất. Nhóm Phát triển và Product Owner hợp tác để giúp Nhóm Phát triển xác định số lượng hạng mục Product Backlog mà họ có thể chuyển thành tính năng sản phẩm trong Sprint sắp diễn ra. Trong phần thứ hai, Nhóm Phát triển lên kế hoạch để thực hiện nhiệm vụ đã chọn thông qua việc thiết kế và phân tách các công việc nhằm biết được cách để đạt Mục tiêu Sprint.

## **Sprint Retrospective (Cải tiến Sprint)**

Một sự kiện trong Scrum. Sự kiện này được hỗ trợ bởi ScrumMaster để toàn bộ Nhóm Phát triển thảo luận về Sprint vừa kết thúc nhằm tìm ra những thay đổi để có thể làm cho Sprint tiếp theo trở nên thú vị và năng suất hơn.

## **Sprint Review (Sơ kết Sprint)**

Một sự kiện trong Scrum, được đóng khung trong hai giờ đồng hồ (đối với một Sprint hai tuần) diễn ra ở cuối mỗi Sprint để Nhóm Phát triển phối hợp với Product Owner và các bên liên quan nhằm thanh tra kết quả của Sprint. Sự kiện này thường bắt đầu bằng việc rà soát lại những hạng mục Product Backlog đã được hoàn thành, một phiên thảo luận về các cơ hội, hạn chế và rủi ro, và một phiên thảo luận về những thứ tốt nhất mà chúng ta nên làm tiếp theo (dẫn đến thay đổi trên Product Backlog). Chỉ những tính năng của sản phẩm đã được hoàn thành thì mới được trình diễn.

## **Stakeholder (Bên liên quan)**

Những người có quyền lợi trong kết quả của dự án, có thể là vì họ đã đầu tư, hoặc sẽ dùng, hoặc sẽ ảnh hưởng đến họ.

## **Team (Nhóm)**

Cách viết tắt của Development Team trong một số tài liệu. Một đội liên chức năng bao gồm những người chịu trách nhiệm tự quản lí để phát triển một phần tăng trưởng sản phẩm trong mỗi Sprint.

## **Test-Driven Development (Phát triển Hướng Kiểm thử)**

Thường được gọi tắt là TDD, đây là một phương pháp phát triển phần mềm mà các hoạt động lập trình, kiểm thử và thiết kế được đan xem vào nhau.

## **Time-box (Khung thời gian)**

Một khoảng thời gian không được phép kéo dài thêm để thực hiện một sự kiện hoặc hoạt động nào đó. Ví dụ, một buổi Scrum Hằng ngày được đóng khung trong 15 phút và bắt buộc kết thúc sau 15 phút bất kể kết quả như thế nào. Đối với các sự kiện, khung thời gian có thể ngắn hơn. Còn đối với các Sprint thì chúng phải có độ dài chính xác như thế.

## **User Story (User Story)**

User Story là một tài liệu sơ giản mô tả yêu cầu sản phẩm với góc nhìn người dùng. Thông thường, User Story do khách hàng, hoặc đại diện của khách hàng viết, tuy nhiên nếu có sự cộng tác của Nhóm Phát triển thì nhóm và khách hàng sẽ có sự chia sẻ hiểu biết về sản phẩm tốt hơn.

## **Velocity (Tốc độ)**

Tốc độ được tính bằng số lượng đơn vị được hoàn thành trong mỗi Sprint. Nếu bạn dùng đơn vị là điểm (point), thì tốc độ chính là số điểm mà nhóm hoàn thành được trong một Sprint. Qua thời gian, tốc độ của nhóm có thể sẽ tương đối ổn định. Đó là tiền đề quan

trọng để nhóm có thể phỏng đoán khối lượng công việc của nhóm trong mỗi Sprint.

# Phụ lục 2 Danh mục kiểm tra

## Kiểm tra hằng ngày

- ☐ Đã làm rõ yêu cầu sản phẩm cho Nhóm Phát triển (nếu có)
- ☐ Đã làm mịn các hạng mục của Product Backlog, đặc biệt các hạng mục ở trên

## Kiểm tra theo Sprint

- ☐ Đã làm việc với các bên liên quan để cập nhật những kỳ vọng mới về sản phẩm
- ☐ Đã sắp xếp các hạng mục trong Product Backlog đảm bảo tối ưu hóa ROI
- ☐ Đã làm mịn các hạng mục của Product Backlog
- ☐ Đã xác định tiêu chí chấp nhận cho các hạng mục sẽ phát triển trong Sprint tới
- ☐ Đã tham gia Lập kế hoạch Sprint để trình bày về tổng quan sản phẩm, từng hạng mục sẽ làm trong Sprint tới và giải đáp mọi thắc mắc về yêu cầu trong Lập kế hoạch Sprint
- ☐ Đã tham gia và đưa ra phản hồi về sản phẩm trong Sơ kết Sprint

## Danh mục kiểm tra của ScrumMaster

### Công việc hằng ngày

- ☐ Nhóm của bạn có ở trạng thái tốt không?
- Mục tiêu rõ ràng (những kỳ vọng và quy định phải rõ ràng, và mục tiêu có thể đạt được, phù hợp với kỹ năng và khả năng của mỗi

người).

- Tập trung và có trọng điểm, tập trung cao độ vào một lĩnh vực nhất định cần được chú ý.
  - Không có cảm giác tự ti, đề cao hành động và nhận thức.
  - Phản hồi trực tiếp và ngay lập tức (nhanh chóng nhìn thấy các thành công và thất bại của một chuỗi hoạt động, nhờ thế có thể điều chỉnh hành vi nếu cần thiết).
  - Cân bằng giữa cấp độ khả năng và thử thách (hoạt động đưa ra không quá khó cũng không quá dễ).
  - Mỗi người đều có khả năng tự kiểm soát trong mỗi tình huống hay hoạt động.
  - Mỗi hoạt động đều hiển nhiên đem lại kết quả, vì thế không cần quá nhiều cố gắng trong hành động.
- ☐ Các thành viên có thích nhau không, có thư giãn cùng nhau không, và có vui mừng trước thành công của những thành viên khác không?
  - ☐ Các thành viên nhóm có cùng nhau giữ cho mỗi người đều phải đạt được những tiêu chuẩn cao, và thúc đẩy mỗi người đều phát triển?
  - ☐ Có vấn đề hoặc cơ hội nào mà nhóm đang không thảo luận cùng nhau vì những vấn đề hoặc cơ hội đó có thể gây ra tình trạng quá khó chịu trong nhóm không?
  - ☐ Nhóm có tập trung liên tục vào các mục tiêu Sprint không? Có thể bạn nên thực hiện một bước kiểm tra giữa Sprint để có thể tái rà soát các tiêu chuẩn chấp thuận của các hạng mục trong Product Backlog đã cam kết hoàn thành trong Sprint hiện tại.
  - ☐ Bảng phân công nhiệm vụ của Sprint có phản ánh đúng những công việc mà nhóm đang thực sự làm không? Cần nhận thức rõ

“vấn đề tiềm ẩn” của những công việc không được nêu ra và những nhiệm vụ có thể tốn hơn một ngày mới có thể hoàn thành. Những công việc không liên quan đến những cam kết trong Sprint sẽ là những trở ngại cho việc hoàn thành các cam kết đó.

- ☐ Bảng phân công nhiệm vụ của nhóm bạn có cập nhật không?
- ☐ Các thành viên nhóm có biết về các công cụ tự quản lí của nhóm không, các công cụ đó có tiện dụng không?
- ☐ Những công cụ quản lí có được những người ngoài nhóm tôn trọng đúng mức không? Sự giám sát quá mức đối với các hoạt động thường nhật của những người bên ngoài nhóm có thể hủy hoại sự minh bạch và cơ chế tự quản lí trong nhóm.
- ☐ Các thành viên nhóm có tự nguyện nhận nhiệm vụ không?
- ☐ Sự cần thiết của việc hoàn trả nợ kĩ thuật, việc dần dần sẽ giúp cho việc lập trình trở nên dễ chịu hơn, đã được ghi nhận rõ ràng trong khái niệm hoàn thành chưa?
- ☐ Các thành viên nhóm có đang cùng nhau chịu trách nhiệm về mọi khía cạnh của sản phẩm chung (kiểm thử, tài liệu cho người dùng, v.v.) mà không quan tâm đến chức danh của mỗi người không?

## **Công việc theo Sprint**

- ☐ Product Backlog có được sắp xếp theo hiểu biết mới nhất của nhóm không?
- ☐ Các yêu cầu và mong muốn từ tất cả các bên liên quan có được ghi nhận trong Product Backlog không? Ghi nhớ: backlog là quan trọng.
- ☐ Khối lượng của Product Backlog có thể quản lí được không? Để duy trì số lượng hạng mục trong giới hạn có thể quản lí được, hãy giữ những hạng mục chi tiết ở trên cùng, còn các hạng mục trừu tượng nói chung ở dưới. Sẽ phản tác dụng nếu chúng ta phân tích

quá sâu vào những hạng mục ở phía dưới của Product Backlog. Những yêu cầu đặt ra sẽ thay đổi trong những cuộc trao đổi liên tục giữa sản phẩm đang phát triển và các bên liên quan hoặc khách hàng.

- Có yêu cầu nào (đặc biệt là những yêu cầu ở phía trên cùng của Product Backlog) có thể được thể hiện tốt hơn bằng những user story độc lập, có thể thương lượng được, có thể đánh giá được, có thể ước lượng được, nhỏ, và có thể kiểm thử được?
- Bạn đã giải thích cho Product Owner của bạn về nợ kỹ thuật và cách để tránh nó chưa? Một trong những giải pháp có thể là thêm việc viết kiểm thử tự động và tái cấu trúc vào định nghĩa "hoàn thành" cho mỗi hạng mục.
- Backlog có phải là một biểu đồ thông tin, mà các bên liên quan có thể lập tức nhận thấy được không?
- Nếu bạn đang sử dụng một công cụ tự động trong quản lý backlog, mọi người có biết cách để sử dụng nó dễ dàng không? Các công cụ quản lý tự động cũng dẫn tới nguy cơ trở thành sự tắc nghẽn thông tin nếu thiếu đi sự truyền tải thông tin chủ động từ ScrumMaster.
- Bạn có thể giúp truyền tải thông tin bằng các bản in cho mỗi người không?
- Bạn có thể giúp truyền tải thông tin bằng cách tạo ra các biểu đồ to và rõ ràng không?
- Bạn đã giúp Product Owner sắp xếp các hạng mục backlog vào các thời điểm phát hành thích hợp hoặc trong những nhóm ưu tiên chưa?
- Có phải mỗi người trong nhóm đều nhận thức được liệu kế hoạch phát hành còn phù hợp với thực tế không? Bạn có thể thử cho mọi người xem Biểu đồ tương quan sản phẩm/phát hành với thời gian thực hiện sau khi các hạng mục đã được xác nhận "hoàn thành"



trong mỗi cuộc họp Sơ kết Sprint. Biểu đồ thể hiện tỉ lệ các hạng mục Product Backlog đã được hoàn thành và những hạng mục mới được thêm vào để cho phép phát hiện sớm những biến động trong quy mô hoặc kế hoạch thực hiện.

☐ Product Owner của bạn đã điều chỉnh kế hoạch phát hành sau buổi họp Sơ kết Sprint gần nhất chưa? Chỉ có số ít Product Owner đã bàn giao sản phẩm được kiểm thử đầy đủ đúng hạn sắp xếp lại kế hoạch phát hành mỗi Sprint. Điều này có thể sẽ khiến cho một vài sản phẩm cần được phát hành sau vì có những việc quan trọng hơn được phát hiện ra.

☐ Bạn đã từng thử nhiều cách thức và địa điểm cho các buổi họp Cải tiến Sprint chưa?

### **Danh mục kiểm tra Lập kế hoạch Sprint**

☐ Buổi lập kế hoạch có diễn ra đúng giờ?

☐ Có thành viên nào thiếu không? Lý do vì sao?

☐ Product Backlog có sẵn sàng trước buổi lập kế hoạch? Product Owner có sơ kết lại tình hình sản phẩm hiện tại đầu buổi lập kế hoạch?

☐ Product Owner có đưa ra mong muốn mục tiêu Sprint đầu buổi lập kế hoạch?

☐ Khả năng của Nhóm Phát triển đã được tính?

☐ Có những vấn đề, sự kiện nào đặc biệt ảnh hưởng tới khả năng của nhóm không (ví dụ nghỉ hè, thành viên nào có việc riêng)?

☐ Nhóm Phát triển và Product Owner có rà soát những hạng mục có thể sẽ phát triển trong Sprint?

☐ Những hạng mục có thể sẽ phát triển trong Sprint đã có tiêu chí chấp nhận?

- ☐ Nhóm Phát triển đã phân ra các hạng mục sẽ phát triển thành các công việc đưa vào Sprint Backlog?
- ☐ Các hạng mục trong Sprint Backlog đã được ước lượng? Product Owner có hài lòng với cam kết của Nhóm Phát triển?
- ☐ Nhóm Phát triển có tin vào khả năng hoàn thành cam kết? Có điều gì cần lưu ý trong Sprint?
- ☐ Buổi Lập kế hoạch Sprint có đảm bảo khung thời gian?
- ☐ Đã cập nhật các tạo tác như Sprint Backlog, Biểu đồ Burndown?

### **Danh mục kiểm tra Cải tiến Sprint**

- ☐ Buổi Sơ kết Sprint có thiếu thành viên nào trong Nhóm Scrum?
- ☐ Buổi Sơ kết Sprint có diễn ra đúng thời gian?
- ☐ Nhóm Scrum có trình bày Mục tiêu Sprint?
- ☐ Nhóm Phát triển chỉ trình bày những hạng mục đã hoàn thành?
- ☐ Product Owner và các bên liên quan có kiểm tra sản phẩm?
- ☐ Product Owner có đưa ra quyết định chấp nhận hoặc không chấp nhận phần tăng trưởng?
- ☐ Product Owner và các bên liên quan có đưa ra phản hồi về sản phẩm?
- ☐ Buổi Sơ kết Sprint có giữ đúng khung thời gian?

### **Danh mục kiểm tra Sơ kết Sprint**

- ☐ Nhóm Phát triển có mời thêm thành viên khác tham gia?
- ☐ Có thành viên nào của Nhóm Phát triển thiếu không? Lý do vì sao?

- Địa điểm, các văn phòng phẩm đã được chuẩn bị? Buổi Cải tiến có diễn ra đúng giờ?
- Mọi thành viên đã rõ mục đích, quy tắc và cách thức thực hiện buổi Cải tiến?
- Có rà soát các hành động cải tiến ở những Sprint trước? Đã chuẩn bị các dữ liệu cần thiết cho việc cải tiến?
- Có thành viên nào không tham gia tích cực vào buổi làm việc? Tại sao?
- Có hành động cụ thể được tìm ra?
- Buổi Cải tiến Sprint có giữ đúng khung thời gian?

# Bảng chỉ mục

5WHYs 69, 100, 212

## A

adaptation 51

Agile 3, 9, 10, 13, 16, 17, 18, 20, 23, 26, 27, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 42, 44, 48, 62, 63, 67, 110, 154, 155, 161, 169, 170, 176, 177, 190, 205, 212, 213, 214, 226

Agile Coach 3, 9, 10, 13, 16, 17, 18, 20, 23, 26, 27, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 42, 44, 48, 62, 63, 67, 110, 154, 155, 161, 169, 170, 176, 177, 190, 205, 212, 213, 214, 226

agility 68

artifact 74, 114

ATDD 169, 184, 212

## B

BDD 169, 184, 212

bên liên quan 14, 27, 59, 72, 74, 75, 76, 81, 101, 103, 114, 138, 159, 172, 213, 215, 216, 218, 219

bỏ phiếu chấm 108, 109

Brainstorming 110

Burndown 66, 69, 74, 95, 97, 111, 140, 141, 142, 144, 145, 146, 147, 150, 151, 212

## C

Cải tiến Sprint 48, 50, 69, 72, 75, 85, 86, 88, 104, 105, 106, 107, 108, 109, 111, 215, 216, 219

CD 212, 213

chó chăn cừu 56

chồng lấp 54, 55

CI 188, 200, 212

CMMI 26

command-and-control 157, 204

công cụ 10, 21, 36, 38, 39, 40, 42, 48, 51, 62, 66, 69, 71, 72, 74, 77, 83, 94, 113, 114, 115, 128, 133, 140, 144, 150, 151, 154, 158, 164, 167, 171, 184, 186, 195, 200, 201, 203, 207, 209, 210, 212, 213, 214, 217, 218

Continuous Integration 188

cross-functional 212

Customer-Centric 47

Cynefin 34, 40

## **D**

Daily Scrum 50, 208, 212

Dashboard 158

Design Thinking 36, 186, 187

Development Team 212, 215

DevOps 213

Done 213

dự án 14, 15, 16, 18, 19, 22, 24, 26, 27, 28, 29, 32, 33, 34, 36, 37, 40, 42, 45, 46, 50, 51, 59, 63, 80, 83, 88, 110, 118, 153, 154, 155, 156, 157, 158, 159, 174, 197, 200, 204, 205, 207, 209, 214, 215

dựa theo kế hoạch 33

Định nghĩa Hoàn thành 26, 57, 113, 131, 133, 134, 135, 137, 138, 139, 167, 210, 213

## **E**

eduScrum 45

empirical process control 37

Estimated 124, 213

eXtreme Programming 213

## **F**

framework 38, 42, 203

function point 20

## **G**

Giá trị Scrum 51

Glad-Sad-Mad 105, 108

## **H**

hạng mục Product Backlog 66, 68, 89, 91, 92, 95, 114, 120, 122, 123, 124, 125, 128, 129, 171, 177, 178, 210, 214, 215

hoàn thành 15, 19, 22, 26, 32, 43, 47, 50, 52, 53, 55, 56, 57, 58, 59, 62, 67, 71, 72, 77, 78, 90, 91, 95, 97, 101, 102, 103, 114, 125, 129, 132, 140, 141, 143, 148, 149, 151, 176, 178, 186, 212, 213, 214, 215, 217, 218, 219

Hofstede 160, 161

huấn luyện 25, 65, 67, 69, 138, 167, 200

## **I**

Impediment Backlog 69, 99

Increment 50, 57, 114, 206, 207, 212, 213

incremental 37, 40, 42, 62

inspection 51

## **K**

Kanban 38, 129, 213, 214

Khởi nghiệp Tinh gọn 36

khung thời gian 56, 66, 86, 88, 89, 90, 97, 98, 99, 103, 104, 106, 108, 111, 215

KLOC 20

## **L**

làm mịn Product Backlog 91, 123, 210, 214

Lập kế hoạch Sprint 48, 75, 85, 86, 88, 90, 91, 95, 97, 111, 123, 126, 140, 198, 210, 214, 215, 216, 219

Lập trình cặp 169, 185

Lean 36, 38, 161, 213

LeSS 205

Liên chức năng 63, 78, 212

light weight 16

Limit WIP 82, 213

luồng 200, 209, 213

## **M**

minh bạch 14, 25, 51, 52, 58, 66, 71, 74, 77, 80, 83, 86, 114, 119, 132, 133, 157, 158, 194, 197, 200, 204, 209, 217

mục tiêu Sprint 71, 97, 140, 148, 217, 219

## **N**

Nexus 203, 205, 207, 208, 209, 210

nguyên nhân gốc rễ 100, 109

Nhóm Phát triển 32, 48, 50, 52, 56, 59, 61, 62, 65, 66, 67, 68, 69, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 88, 89, 90, 91, 92, 95, 97, 98, 99, 100, 101, 103, 104, 106, 111, 115, 122, 125, 126, 130, 131, 134, 135, 138, 140, 141, 147, 148, 149, 151, 159, 172, 174, 178, 188, 197, 212, 213, 214, 215, 216, 219

Nhóm Scrum 9, 48, 59, 61, 62, 63, 65, 67, 77, 83, 90, 91, 94, 103, 114, 115, 123, 135, 138, 151, 155, 156, 158, 174, 176, 177, 190, 207, 208, 212, 219

nợ kĩ thuật 71, 72, 217, 218

## **P**



Pair-Programming 38

Personal Kanban 38, 213

phân đoạn 24, 42, 43, 53, 59, 62, 214

phân tán 11, 162, 191, 192, 193, 194, 195, 196, 197, 198, 199, 201, 204, 209, 210

phần tăng trưởng 32, 46, 58, 77, 78, 80, 88, 101, 103, 114, 131, 132, 212, 214, 215, 219

phát hành 14, 15, 27, 32, 57, 58, 72, 88, 130, 137, 138, 147, 148, 149, 212, 213, 218

Phát triển Hướng Hành vi 184, 212

Phát triển Hướng Kiểm thử 179, 184, 212, 215

Phát triển Hướng Kiểm thử Chấp nhận 184, 212

Phát triển Phần mềm Tinh gọn 213

phức tạp 34, 37, 42, 52, 80, 98, 136, 204, 214

plan-driven 33

Planning Poker 95, 122, 176, 177, 178

PM 70

point 20, 92, 95, 176, 178

Product Backlog 32, 50, 57, 66, 68, 71, 72, 73, 74, 75, 76, 77, 81, 83, 89, 91, 92, 95, 97, 101, 102, 110, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 128, 129, 149, 151, 171, 174, 177, 178, 206, 207, 210, 212, 214, 215, 216, 217, 218, 219

Product Owner 27, 48, 50, 56, 61, 62, 63, 65, 66, 67, 68, 69, 72, 73, 74, 75, 76, 77, 81, 83, 88, 89, 90, 91, 92, 95, 97, 98, 101, 102, 103, 104, 111, 119, 121, 135, 145, 146, 150, 151, 154, 156, 158, 159, 162, 171, 172, 174, 197, 198, 201, 207, 212, 213, 214, 215, 216, 218, 219

## **Q**

quản lí vật 56

Quản trị dự án 153, 158

quy mô lớn 11, 80, 155, 203, 204, 205, 210

Quy tắc làm việc 83

quy trình 16, 21, 22, 25, 29, 35, 38, 39, 40, 50, 62, 67, 100, 104, 136, 138, 155, 158, 209, 214

## **R**

Release Burndown 145, 146, 147, 150, 212

ROI 32, 47, 74, 75, 216

rủi ro 15, 37, 46, 47, 52, 53, 58, 74, 132, 212, 215

## **S**

SAFe 205

Scrumban 31, 214

ScrumBut 157, 158

Scrum Hằng ngày 42, 48, 66, 69, 85, 86, 88, 98, 99, 100, 111, 125, 212, 215

ScrumMaster 48, 50, 56, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 83, 89, 98, 99, 100, 101, 104, 105, 111, 135, 151, 154, 157, 158, 159, 162, 197, 198, 199, 207, 212, 214, 215, 216, 217, 218

Scrum of Scrums 205

Scrum phân tán 11

Self-organized 214

Servant Leader 65

SMART 94

Sơ kết Sprint 48, 50, 57, 69, 72, 75, 85, 86, 88, 101, 102, 103, 104, 111, 210, 213, 215, 216, 218, 219

SpeedBoat 105, 107, 109

Sprint 32, 41, 42, 48, 50, 52, 53, 56, 57, 59, 66, 69, 71, 72, 74, 75, 77, 78, 80, 85, 86, 88, 89, 90, 91, 92, 94, 95, 97, 101, 102, 103, 104, 105, 106, 107, 108, 109, 111, 113, 114, 116, 123, 125, 126, 128, 129, 130, 131, 132, 137, 138, 140, 141, 142, 143, 144, 146, 147, 148, 149, 151, 155, 158, 159, 174, 177, 178, 195, 198, 204, 206, 207, 208, 209, 210, 212, 213, 214, 215, 216, 217, 218, 219

Sprint Backlog 42, 48, 50, 66, 95, 97, 113, 114, 125, 126, 128, 129, 138, 140, 141, 144, 151, 177, 178, 207, 209, 212, 214, 215, 219

Sprint Burndown 125

Sprint Goal 53, 215

Sprint Planning 215

Sprint Retrospective 50, 215

Stakeholder 215

standup meeting 24

## T

tầm nhìn 51, 68, 73, 163, 197

Tảng băng trôi 164

tăng trưởng 13, 32, 37, 40, 41, 42, 46, 48, 57, 58, 77, 78, 80, 88, 101, 103, 113, 114, 130, 131, 132, 195, 204, 206, 212, 213, 214, 215, 219

tạo tác 10, 37, 42, 74, 97, 113, 114, 133, 151, 195, 199, 204, 209, 214

TDD 38, 169, 179, 180, 182, 184, 190, 212, 215

technical debt 179

Thanh tra 48, 51, 59, 69, 207

thay đổi 15, 21, 22, 28, 29, 32, 33, 36, 40, 42, 46, 47, 50, 51, 52, 53, 56, 59, 66, 67, 70, 72, 80, 88, 89, 109, 149, 153, 154, 155, 157, 160, 163, 164, 167, 182, 183, 188, 201, 207, 212, 213, 214, 215, 218

thích nghi 46, 47, 51, 86, 101, 102, 104, 114, 160, 183, 192, 212

thí điểm 155

thuyết quản lí thực nghiệm 37

tích hợp liên tục 24, 66, 169, 209

tiến độ 14, 22, 46, 50, 51, 74, 77, 95, 100, 111, 114, 116, 140, 145, 148, 150, 200, 212

tiếp cận lặp 13

Time-box 215

Tinh gọn 36, 213

tổ chức học tập 105

Trao quyền 163

Triển khai Liên tục 212, 213

Triết lí Agile 38

trụ cột 37, 48, 51, 59, 157

trực quan 46, 114, 140, 144, 150

tuần tự 33, 53, 54

Tuckman 64

tự tổ chức 22, 47, 48, 61, 62, 77, 83, 133, 157, 206, 214

Tuyên ngôn Agile 17, 23, 26, 27, 30, 36, 40, 42, 62, 214

## **U**

ước tính 43, 81, 92, 100, 116, 122, 124, 128, 137, 140, 143, 147, 151, 175, 176, 177, 178, 213, 214, 215

User Story 110, 120, 137, 169, 171, 172, 173, 174, 175, 176, 215

waterfall 20, 53

## **X**

XP 16, 27, 31, 36, 179, 213

## **Giới thiệu Học viện Agile**

Học viện Agile được thành lập với mong muốn trở thành địa chỉ đào tạo tin cậy, chuyên sâu, toàn diện về Agile cùng những tri thức liên

quan đến phát triển sản phẩm và đổi mới quản lí trong lĩnh vực kinh tế sáng tạo.

Đội ngũ sáng lập gồm những người tiên phong trong việc truyền bá và phát triển cộng đồng Agile tại Việt Nam mong muốn xây dựng Học viện Agile thành đơn vị tiên phong trong việc nâng tầm năng lực sáng tạo và đổi mới với những giá trị về hạnh phúc đích thực, tình bằng hữu bền chặt, và không ngừng đổi mới sáng tạo.