

# 7

## SCRUM PHÂN TÁN

Trong chương này...

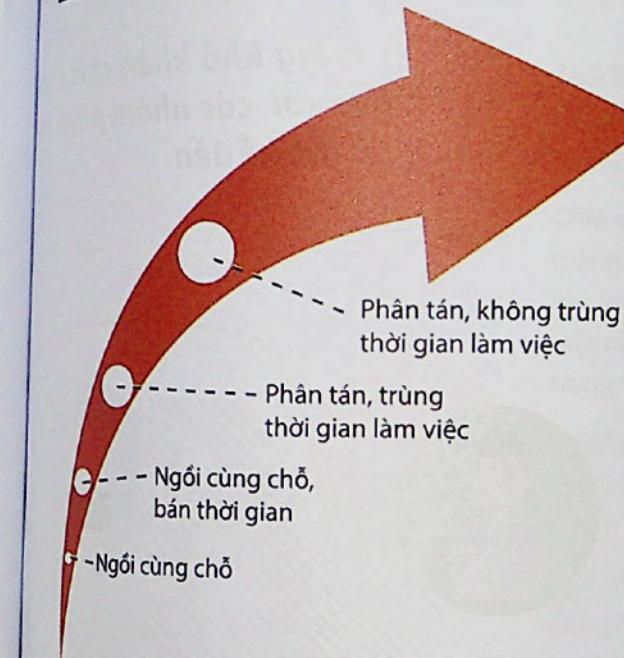
- Phân loại mức độ phân tán
- Các thử thách chính của nhóm phân tán và cách vượt qua
- Họp Tiền-kế-hoạch-hóa
- Lưu ý cho công tác hậu cần
- Thiết lập hạ tầng

“

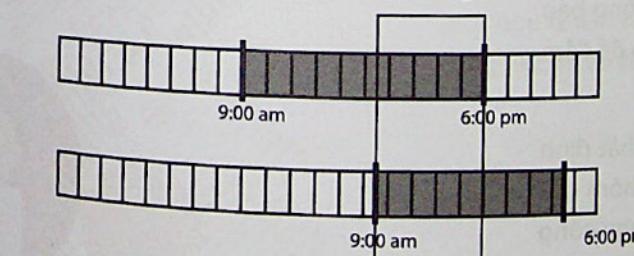
Biết một vài trong số các câu hỏi còn tốt hơn biết tất cả các câu trả lời.

James Thurber

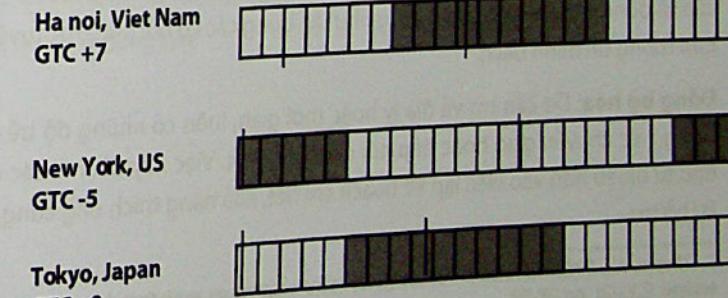
## LÀM VIỆC PHÂN TÁN



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



### Phân tán toàn bộ



Hiện nay vì nhiều lý do khiến các Nhóm Scrum phải làm việc phân tán. 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.

Ngay cả khi một nhóm ngồi cùng một chỗ thì họ vẫn làm việc với nhau phân nào là phân tán. Mỗi người đều có nhiệm vụ riêng của mình, nhịp làm việc và sinh hoạt riêng nên mặc dù cùng làm việc trong một không gian nhưng thời lượng làm việc trực tiếp với nhau chỉ gói gọn trong các hoạt động như: thảo luận nhóm, họp, lập kế hoạch, v.v..

Với những nhóm phân tán thực sự, do đặc thù lệch nhau về thời gian làm việc, hoặc cách xa nhau về địa lý, họ phải có phương án điều chỉnh cách làm việc cho phù hợp. Nhóm Scrum làm việc phân tán cũng phải lưu ý các thực tế này để thích nghi.

# THỬ THÁCH ĐỐI VỚI NHÓM PHÂN TÁN

Làm việc phân tán cũng có một số lợi ích nhất định. Có nghiên cứu cho thấy những tổ chức cho phép các nhân viên của mình làm việc từ xa tiết kiệm được chi phí về không gian làm việc và đầu tư trang thiết bị, nội thất; làm việc từ xa còn giúp gia tăng năng suất<sup>1</sup>. Những người làm việc từ xa được trao cơ hội để chủ động hơn trong công việc, thời gian làm việc với sự cân bằng cuộc sống<sup>2</sup>. Tuy vậy, không phải là không có những thách thức đáng kể. Những khó khăn chính yếu đối với các nhóm phân tán có thể kể đến như dưới đây.

**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 lý 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.

**Đồng bộ hóa:** Do cần trở về địa lý 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.

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



**Tích hợp:** Do các tạo 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 chuyển 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 lý 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.

Một khảo sát đăng trên HBR<sup>3</sup> cho thấy có một tỷ lệ lớn (52%) người làm việc phân tán cảm thấy đồng nghiệp không đối xử bình đẳng với họ. Họ cũng lo lắng về việc các đồng nghiệp nói xấu sau lưng, và không đấu tranh cho những quyền lợi của mình.

Những thách thức như vậy là không tránh khỏi với những Nhóm Scrum làm việc phân tán. Để hạn chế những khó khăn này các nhóm tìm kiếm và lựa chọn những công cụ, phương pháp phù hợp nhất với nhóm của mình. Một số công cụ này bạn có thể tìm hiểu ở phần **Công cụ cộng tác phân tán** trong chương này.

<sup>1</sup> Bloom, N., 2014. To raise productivity, let more employees work from home. Harvard business review, 92(1/2), pp.28-29.

<sup>2</sup> Dominus, S., 2016. Rethinking the work-life equation. The New York Times Magazine, pp.47-49.

<sup>3</sup> <https://hbr.org/2017/11/a-study-of-1100-employees-found-that-remote-workers-feel-shunned-and-left-out>

## GIAO TIẾP HIỆU QUẢ HƠN

Do 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 tinh 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.

### Giao tiếp “giàu” thông tin hơn

Giao tiếp “nghèo” thông tin hơn

- Giao tiếp mặt đối mặt
- Truyền hình trực tuyến với màn hình lớn độ nét cao
- Truyền hình trực tuyến độ nét thấp
- Gọi điện
- Nhắn tin
- Dùng wiki/tài liệu cộng tác
- Email

## 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 kỳ, 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 rất 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. Một số cách thức có thể giúp gia tăng và củng cố niềm tin trong Nhóm Scrum mà bạn có thể xem xét và áp dụng:

1. Kết nối thường xuyên và nhất quán phương thức kết nối.
2. Đặt lịch làm việc trực tiếp mặt-đối-mặt hoặc qua các phương tiện liên lạc giàu thông tin như video hay voice chat.
3. Đưa ra các kỳ vọng rõ ràng.
4. Ưu tiên việc tạo dựng và duy trì các mối quan hệ giữa những thành viên trong nhóm.



### Dùng 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 Sprint

• Chủ yếu chuẩn bị cho phần mục tiêu 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, Hangout 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 v.v.. 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, chia sẻ trực tuyến 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.



## NHÓM CÔNG CỤ GIAO TIẾP

 Slack là công cụ rất mạnh mẽ trong lĩnh vực này, và nó đã được chứng minh là một công cụ hấp dẫn và thú vị để giao tiếp nhóm, đặc biệt là đối với các nhóm làm việc từ xa. Các tính năng "xã hội" như biểu tượng cảm xúc, v.v.. giúp giao tiếp từ xa tự nhiên hơn.

 Stride là một ứng dụng mới về chat, nó có một số tính năng độc đáo, thực sự thú vị như decision making và focus mode. Ngoài ra, các cài đặt thông báo trạng thái ("đang họp" hoặc "đang họp", v.v..) đều hữu ích và thú vị, đặc biệt đối với những nhóm làm việc từ xa.

Tại Zapier, chúng tôi nhận thấy rằng sự minh bạch trong giao tiếp là chìa khóa để thành công cho làm việc từ xa. Hầu như tất cả các cuộc trò chuyện của chúng tôi đều diễn ra trên các kênh công khai, vì vậy bất kỳ ai cũng có thể trò chuyện và đọc về những gì đang diễn ra giữa nhóm và các phòng ban. Điều này đặc biệt hữu ích với các nhóm làm việc ở các múi giờ khác nhau. Khi các thành viên trong nhóm thức dậy, họ có thể dễ dàng hiểu được tình hình và biết những gì người khác đã trao đổi thay vì không biết chuyện gì xảy ra trong khi ngủ." - Wade Foster, CEO của Zapier-

 Skype là ứng dụng cho phép trò chuyện video và đàm thoại giữa máy tính, máy tính bảng, thiết bị di động, đồng hồ thông minh, v.v.. qua Internet. Skype cũng cung cấp dịch vụ gửi tin nhắn tức thời. Người dùng có thể gửi các tin nhắn văn bản, video và có thể trao đổi các tài liệu kỹ thuật số như hình ảnh, văn bản, v.v.. Skype còn cho phép triển khai video conference.

 Zoom đã chứng tỏ là công cụ đáng tin cậy nhất trên tất cả các hình thức kết nối Internet, đặc biệt là trong tình huống xử lý hàng trăm người tham gia vào một cuộc họp toàn công ty. Tính năng "gallery view" là bắt buộc đối với các cuộc họp từ xa có nhiều người tham dự.

 appear.in Appear.in là công cụ giúp nhanh chóng tham dự vào một buổi truyền hình trực tuyến và bạn có thể tùy chỉnh các URL, do đó rất phù hợp để tiến hành các cuộc họp 1:1.Thêm vào đó nó cũng cung cấp các tính

## CÔNG CỤ QUẢN TRỊ DỰ ÁN, LUÔNG CÔNG VIỆC



Jira là một sản phẩm theo dõi vấn đề độc quyền được phát triển bởi Atlassian, cho phép quản lý dự án theo dõi lỗi và nhanh chóng. Tên sản phẩm là một sự cắt ngắn của Gojira, từ tiếng Nhật cho Godzilla, là một tham chiếu đến một đối thủ cạnh tranh, Bugzilla.



Redmine là một công cụ theo dõi vấn đề và quản lý dự án dựa trên web miễn phí và mã nguồn mở. Redmine cho phép người dùng quản lý nhiều dự án và các tiểu dự án liên quan. Nó có tính năng cho mỗi wiki dự án và diễn đàn, theo dõi thời gian và kiểm soát truy cập dựa trên vai trò linh hoạt. Nó bao gồm một biểu đồ lịch và Gantt để hỗ trợ trình bày trực quan các dự án và thời hạn của chúng. Redmine tích hợp với các hệ thống kiểm soát phiên bản khác nhau và bao gồm một trình duyệt kho lưu trữ và trình xem khác.

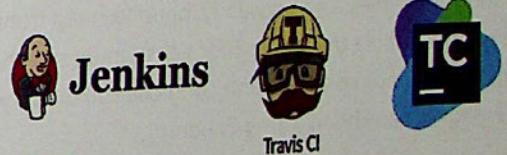


Khi bạn cần xem tiến độ, cập nhật trạng thái nhanh và tất cả các tài nguyên có liên quan đến công việc của một dự án hoặc nhóm, Trello là văn phòng ảo của bạn. Tất cả các cập nhật thông tin trong thời gian thực và sống ở đó 24/7, vì vậy các thành viên trong nhóm có thể nhận bối cảnh, thông tin liên lạc và trạng thái của bất kỳ yêu cầu, dự án, chương trình họp hoặc các mục khác một cách thuận tiện.

## HỆ THỐNG TÍCH HỢP LIÊN TỤC

Một số phần mềm phổ biến phục vụ triển khai Tích hợp liên tục là Jenkins, Travis-CI, TeamCity, Bamboo, v.v...

Xem thêm phần Hệ thống Tích hợp Liên tục trong Chương 9.





## Những hiểu lầm phổ biến

### 1. Scrum chỉ dành cho các nhóm làm việc tập trung tại một địa điểm.

Scrum nhấn mạnh vào cộng tác, giao tiếp mặt-đối-mặt, minh bạch thông tin, v.v.. Điều này không đồng nghĩa với việc nhóm phải cùng làm việc tại một địa điểm. Hiện nay có rất nhiều công cụ và phương pháp đảm bảo được những nguyên tắc kể trên của Scrum mà không đòi hỏi nhóm phải tập trung tại một không gian thời gian với nhau.

### 2. Nhóm Scrum phân tán ảnh hưởng lớn đến chất lượng sản phẩm.

Việc đảm bảo chất lượng không phụ thuộc vào sự tập trung hay phân tán của nhóm phát triển. Scrum nói chung cũng như trong phát triển phần mềm nói riêng có nhiều phương pháp và quy trình giúp đảm bảo và nâng cao chất lượng sản phẩm. Vấn đề chất lượng sản phẩm phụ thuộc vào sự tuân thủ quy trình hay vận hành thuần thực các công cụ, năng lực cộng tác của nhóm mà thôi.

### 3. Scrum không phù hợp cho các dự án phân tán lớn.

Dự án càng lớn, càng kéo dài thì càng khó dự đoán và rất nhiều sự thay đổi, nói chung là càng phức tạp. Scrum ra đời nhằm vào việc giảm thiểu những vấn đề trên. Các nguyên lý, phương pháp trong Scrum giúp sớm phát hiện và thích ứng được với sự phức tạp khó lường này. Xem lại phần nói về tỷ lệ thành công của các dự án và phần "Khi nào Agile, khi nào không" trong Chương 1.

### 4. Người làm việc từ xa là những người lười biếng.

Có một nhận định rằng nếu bạn không thể nhìn thấy một người nào

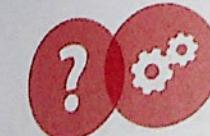
đó đang ngồi ở bàn làm việc của họ, thì có nghĩa là họ sẽ không làm gì cả. Bạn sẽ sớm thấy nhận định này là không đúng đắn. Bất kỳ vị trí nào cũng có thể lười biếng nếu người quản lý không truyền đạt cho họ kỳ vọng, mục tiêu và thời hạn. Với những người làm việc từ xa, nhà quản lý thay vì giám sát thời gian làm việc của họ thì hãy đánh giá kết quả, chất lượng công việc và sự đúng hạn của người đó.

### 5. Làm việc từ xa hủy hoại văn hóa doanh nghiệp.

Tất nhiên với các nhóm làm việc phân tán sẽ không có những giây phút kiểu như ngồi ăn, cafe cùng nhau, tán gẫu nơi hành lang, v.v.. Tuy nhiên với một kế hoạch và sự chuẩn bị chu đáo, nhóm của bạn vẫn có những giây phút bên nhau "từ xa".

### 6. Những người làm việc từ xa là những người luôn sẵn sàng tiếp bạn bất cứ khi nào.

Nhiều người thường giả định rằng bởi vì những người đó luôn ở nhà và họ luôn sẵn sàng trả lời câu hỏi công việc nhanh chóng bất cứ lúc nào. Tất nhiên điều này là không đúng. Một trong những cách làm tốt nhất từ xa là làm việc nghiêm túc, cũng giống như việc họ làm việc trong văn phòng. Người làm việc từ xa được khuyến khích quyết định (và giao tiếp) vào những thời điểm họ sẵn sàng, nghỉ ngơi đúng giờ, và họ vẫn có cuộc sống riêng của mình và cần thoát khỏi không gian làm việc để tham gia vào cuộc sống gia đình của mình. Việc phân tách rạch ròi công việc của công ty và công việc gia đình là khả thi, ngay cả khi văn phòng làm việc ở trong nhà bạn!



## 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 để giải quyết 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?
10. Bạn sẽ chọn cách thức nào để tiến hành hiệu quả buổi Scrum Hàng ngày với nhóm Scrum phân tán?

# 8

## SCRUM 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
- Scrum quy mô lớn với LeSS

“

*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 trong tổ chức. 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 tổ chức. Việc vận dụng Scrum ở quy mô lớn thường được đặt ra khi tổ chức của bạn đã thành thục 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 vẫn phải vấn đề đồng bộ hóa 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 hóa, 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 lý ra lệnh-kiểm soát.

Các vấn đề và chiến lược thích ứng của nhóm phân tán được đề cập tới trong Chương 7 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 lý mới, gồm:

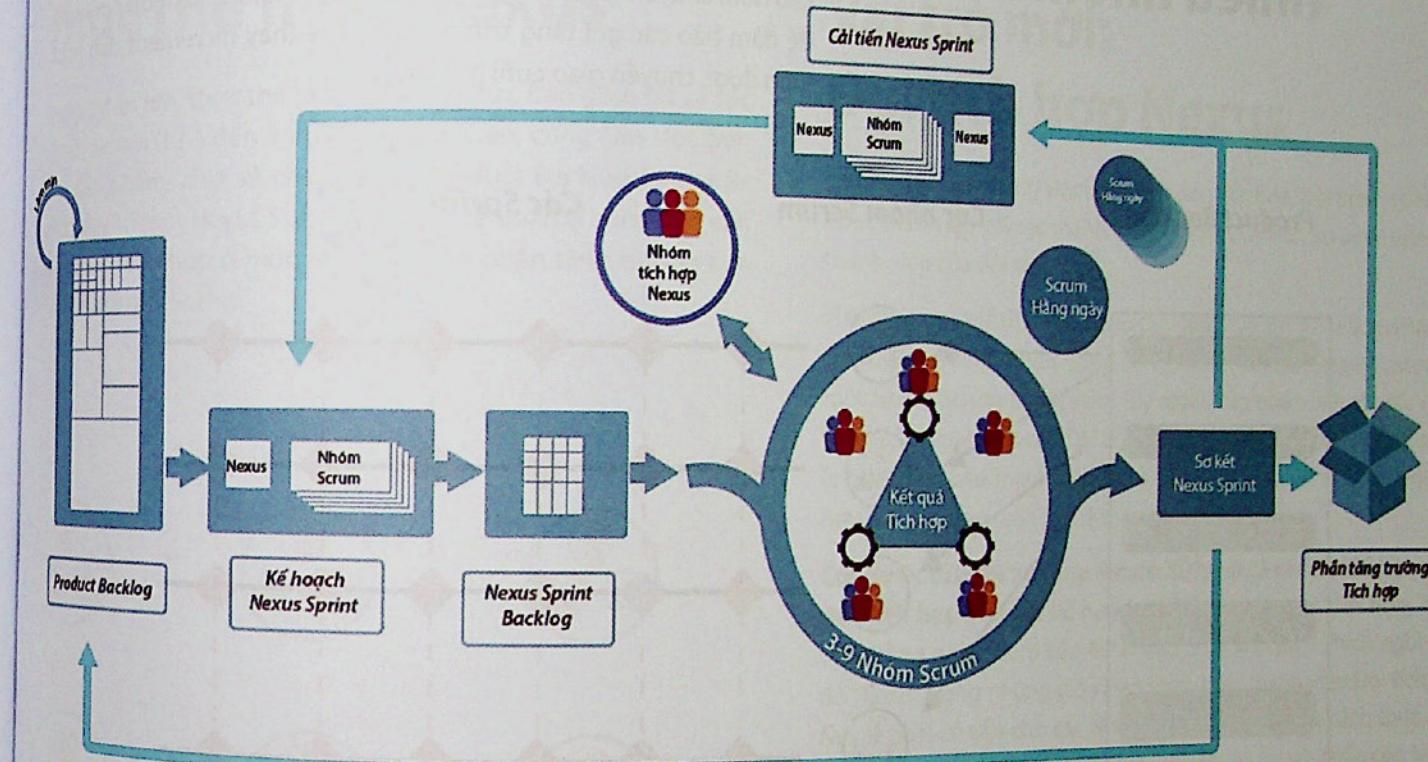
- **Yêu cầu:** các yêu cầu cần được quản lý tập trung, cập nhật liên tục, chia sẻ và minh bạch để các nhóm dễ đồng bộ hóa công việc phát triển.
- **Kiến thức và hiểu biết chung:** thông qua các cơ chế họp và chia sẻ thông tin tập trung, các kiến thức về sản phẩm/công nghệ/tiêu chuẩn sẽ phải được chia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

Hiện nay có nhiều cách tiếp cận đối với việc áp dụng Scrum ở quy mô lớn để giải quyết các vấn đề kể trên. Chúng ta có thể nhắc đến những phương pháp phổ biến như: LeSS, SAFe, Nexus,... và gần đây là Scrum@Scale. Trước đó một biện pháp đã được biết đến từ lâu trong cộng đồng những người thực hành Scrum là Scrum of Scrums. Tất cả các khung làm việc này đều xoay quanh những nhóm liên chức năng, tự quản. Các nhóm được thiết lập dựa trên các yêu cầu của Product Backlog có thể triển khai được thành những gói tăng trưởng nhỏ nhất mà được xây dựng độc lập với nhau. Các nhóm cũng được kỳ vọng sẽ tập trung vào thực hiện tốt các kỹ thuật như tích hợp liên tục, kiểm thử hồi quy tự động, v.v.. Vào cuối mỗi Sprint, mỗi nhóm phải chuyển giao được một gói tăng trưởng sản phẩm có khả năng triển khai được. Những khung làm việc này cũng khuyến khích sử dụng các nguyên tắc Lean để tối ưu hóa luồng công việc.

Trong khuôn khổ của cuốn cẩm nang mang tính nhập môn này, chúng tôi lựa chọn giới thiệu đến bạn đọc hai khung làm việc Scrum quy mô lớn thuộc loại đơn giản nhất là Nexus và LeSS (Large Scale Scrum). Nexus được Ken Swchaber, đồng tác giả Scrum soạn ra và là tài liệu chính quy đào tạo Scrum quy mô lớn trong khuôn khổ tổ chức Scrum.org; còn LeSS là phương pháp được Craig Larman và Bas Vodde phát triển, và được tiếp nhận rộng rãi vì sự đơn giản của nó.

## KHUNG NEXUS CHO SCRUM QUY MÔ LỚN

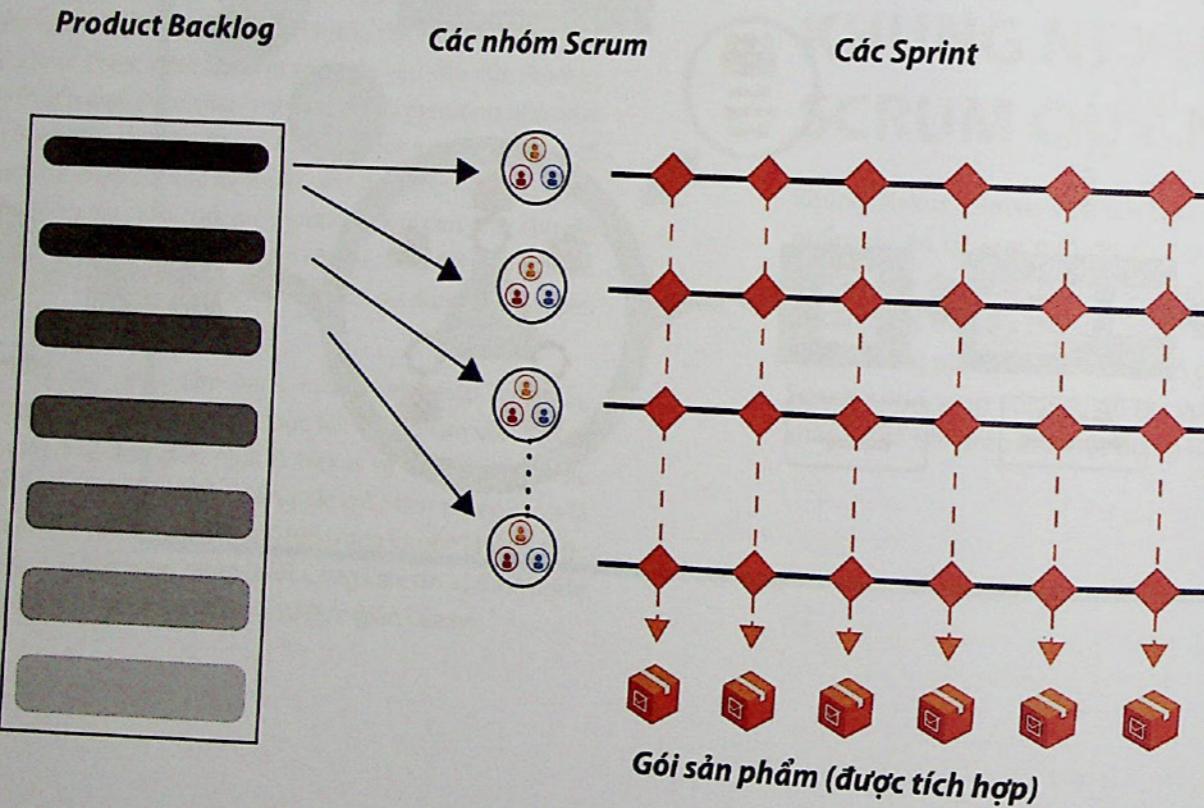
Khung Nexus (Nexus Framework) dành cho các sản phẩm/dự án có quy mô lớn từ 3-9 nhóm Scrum. Bản hướng dẫn định nghĩa chi tiết cách làm được duy trì và cập nhật trên Scrum.org. Về cơ bản, Nexus giữ cho được những giá trị và cách thức tổ chức công việc của Scrum, song song với việc nỗ lực vượt qua những khó khăn đã kể bên trên khi một dự án lớn vận hành.



## 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.

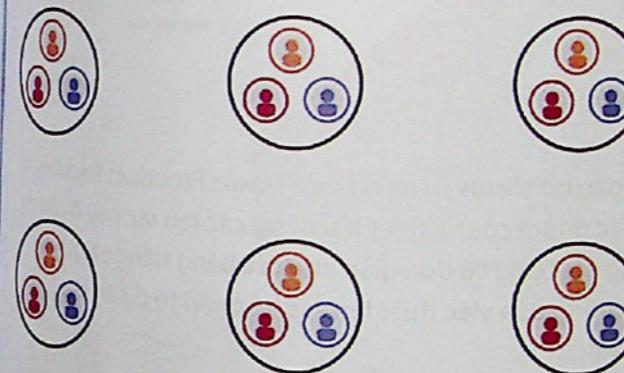
Các nhóm sẽ tự tổ chức để hiện thực hóa các hạng mục có độ ưu tiên cao hơn, chuyển giao các gói tăng trưởng nhỏ, và tích hợp chúng lại để đảm bảo các gói tăng trưởng tổng thể (hay Increment tích hợp) luôn được chuyển giao cuối mỗi Sprint.



## 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 phần tăng trưởng tích hợp ở cuối mỗi Sprint.



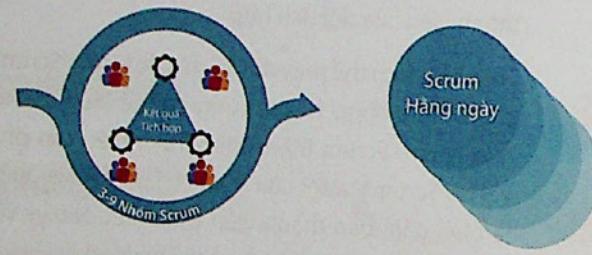
### Vai trò mới: Đội tích hợp Nexus

Để 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 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



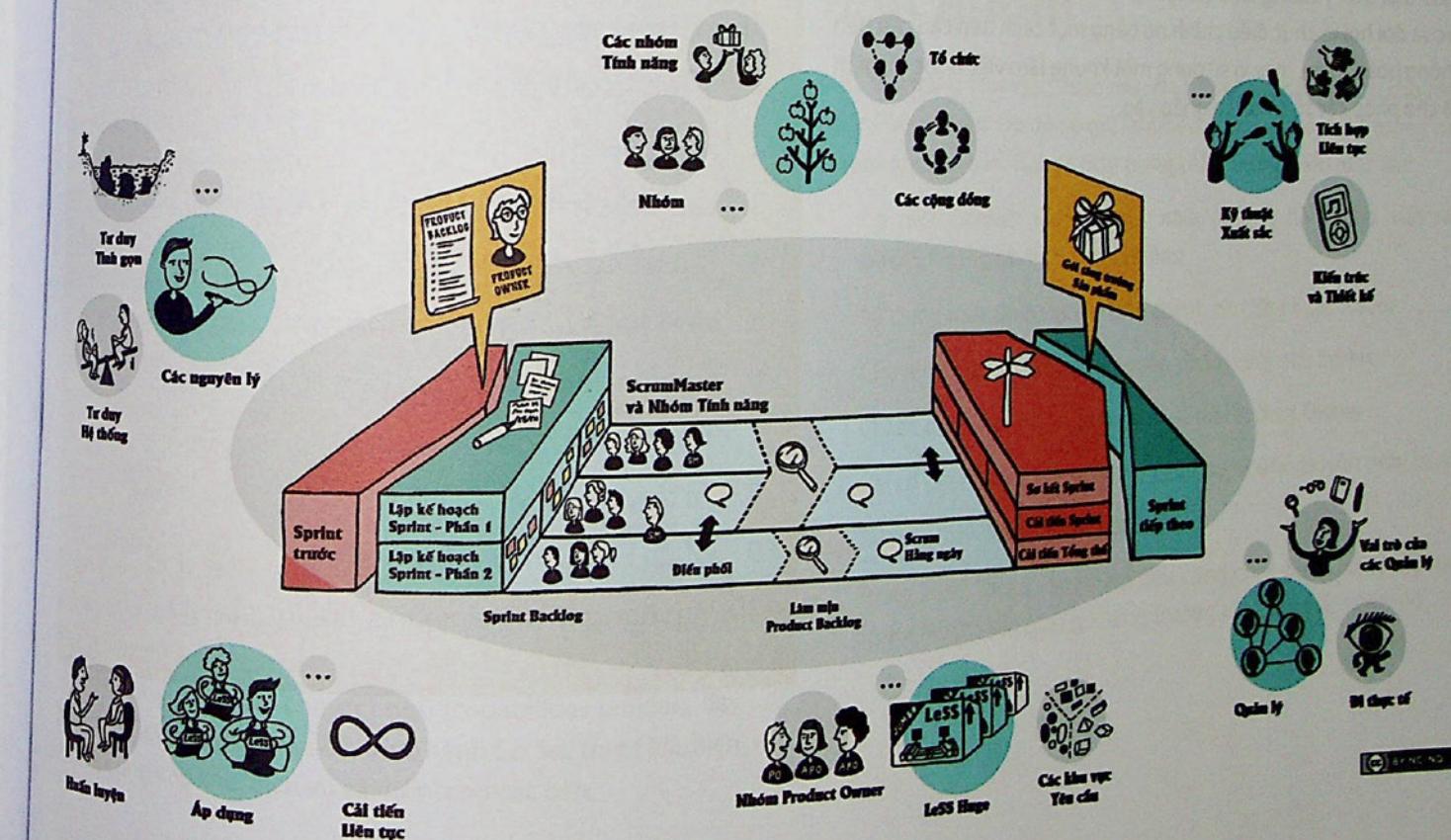
Sprint

## 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.

## SCRUM QUY MÔ LỚN VỚI LESS



LeSS (Large-Scale Scrum) là một khung làm việc Agile để mở rộng Scrum cho nhiều nhóm. Khung làm việc này được đúc rút từ những kinh nghiệm của Bas Vodde và Craig Larman khi triển khai Agile ở quy mô lớn cho nhiều loại hình công ty, sản phẩm và lĩnh vực công nghệ khác nhau trong mười năm qua. LeSS có hai phiên bản: LeSS dành cho 8 Nhóm Scrum làm việc cùng nhau và LeSS Huge dành cho vài nghìn người làm việc trên cùng một sản phẩm. Trong phạm vi tài liệu này, chúng ta chỉ khảo sát sơ bộ phiên bản dành cho quy mô nhỏ hơn, hướng dẫn chi tiết có tại trang chủ của LeSS (<https://less.works>).

Tổng quan LeSS Framework,Ảnh:[less.works](https://less.works).

Chương 8: Scrum với quy mô lớn

LeSS khác so với các khung làm việc Agile cho diện rộng khác ở chỗ nó cung cấp một khung làm việc tối giản cho phép thực hành Scrum trên quy mô lớn, cho phép các nhóm và tổ chức tiến hành thanh tra/đánh giá việc thực hiện dựa trên kinh nghiệm và bối cảnh của họ. LeSS dựa trên ý tưởng việc cung cấp quá nhiều quy tắc, vai trò, tạo tác và đòi hỏi tổ chức điều chỉnh nó bằng một cách tiếp cận căn bản không hoàn thiện, thay vì sử dụng một khung làm việc được tối giản và cho phép các tổ chức lắp ghép vào.

### Nguyên tắc LeSS

Bộ nguyên tắc LeSS hết sức tối giản, gồm 10 điều quy định những thứ quan trọng nhất định hướng hoạt động toàn bộ đội ngũ. Chúng bao gồm:

1. Scrum ở quy mô lớn là Scrum
2. Minh bạch
3. Làm việc ít hơn, hoàn thành nhiều hơn
4. Tập trung vào sản phẩm
5. Lấy khách hàng làm trung tâm
6. Cải tiến liên tục hướng đến sự hoàn hảo
7. Tư duy tinh gọn
8. Tư duy hệ thống
9. Quản lý quy trình thực nghiệm
10. Vận dụng lý thuyết hàng đợi (queue theory)

## Cấu trúc và quản trị

LeSS lưu ý mọi người tổ chức xoay quanh giá trị khách hàng (Customer Value). Các nhóm cần giữ nguyên tắc tổ chức liên- chức-năng và tự quản. Các nhóm tổ chức dưới dạng Feature Team có đặc trưng bền vững, liên chức năng, liên thành phần (cross-component) để hoàn thành nhiều yêu cầu hoàn chỉnh của khách hàng. Cấu trúc của doanh nghiệp nên phẳng, xoay quanh các nhóm tự quản thay vì phân ra thành các phòng ban chức năng. Các nhóm này cần được chuyên tâm, liên chức năng, cùng ngồi làm việc gắn kết tại chỗ và bền vững như những tổ chức nhỏ hơn. Các nhóm khuyến khích cá nhân có nhiều kỹ năng, tự ra quyết định, và tự quản lý lấy các sự phụ thuộc vào các yếu tố bên ngoài. Bên cạnh các nhóm làm việc tạo ra kết quả, LeSS khuyến khích tổ chức các cộng đồng học tập (Communities of Practices) để thúc đẩy việc học tập và chia sẻ trong tổ chức.

Trong một tổ chức LeSS, nhà quản lý làm việc như các ScrumMaster, phân chia rõ vai trò. Nhà quản lý trao quyền quyết định làm gì cho các Product Owner, trao quyền quyết định làm như thế nào cho các Nhóm, còn mình thì tập trung vào xây dựng năng lực cho đội ngũ (capabilities builder), dạy lại cách thức giải quyết vấn đề, thực hành Go See (hiện địa, hiện vật) và cung cấp các "dịch vụ cải tiến" cho các bên.

## Những điểm giống với Scrum áp dụng cho một nhóm

LeSS là khung làm việc được mở rộng từ phiên bản Scrum cho một nhóm tiêu chuẩn. Do đó nó giữ lại nhiều phương pháp và ý tưởng của Scrum nguyên bản. Bạn sẽ thấy trong LeSS những điều như sau:

- Có một Product Backlog duy nhất (nhưng dành cho một sản phẩm, không phải cho một nhóm)
- Sử dụng một Định nghĩa Hoàn thành cho tất cả các nhóm
- Chuyển giao Gói tăng trưởng sản phẩm cuối mỗi Sprint
- Chỉ có duy nhất một người nắm vai trò Product Owner
- Chỉ có các nhóm liên chức năng (không nhóm chuyên môn hóa)
- Tại một thời điểm chỉ có một Sprint
- Trong LeSS, tất cả các Nhóm đều cùng chạy nước rút trong một Sprint để chuyển giao gói tăng trưởng khi Sprint đó kết thúc.

# Những điểm khác trong LeSS

## Lập kế hoạch Sprint - Phần 1

Ngoài Product Owner còn có sự tham gia của các thành viên đến từ tất cả các nhóm. Các thành viên nhóm tự quản trong việc quyết định phân chia các Hạng mục Product Backlog. Họ cũng thảo luận để tìm kiếm các cơ hội chia sẻ, cộng tác giữa các nhóm, đặc biệt là giữa những nhóm có Hạng mục Backlog liên quan với nhau.

## Lập kế hoạch Sprint - Phần 2

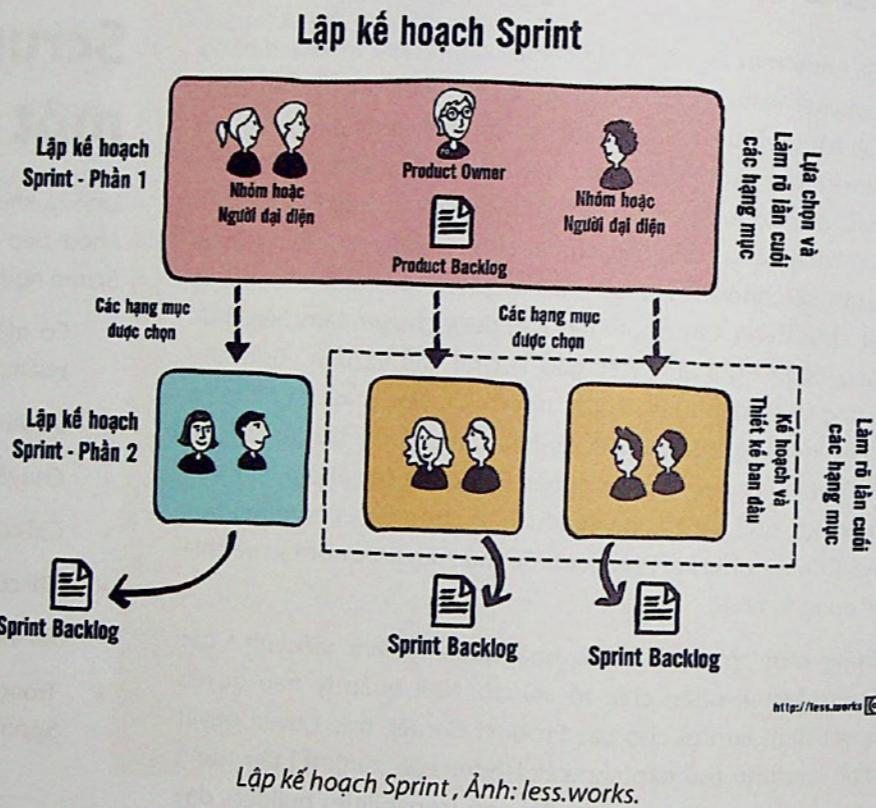
Phần này các nhóm tổ chức độc lập và thường song song với nhau. Đôi khi để phối hợp hay học tập đơn giản các nhóm có thể tổ chức Phần 2 của buổi Lập kế hoạch Sprint trong cùng một không gian nhưng ở hai khu vực tách biệt.

### Scrum Hàng ngày

Được tổ chức độc lập giữa các nhóm, tuy nhiên thành viên của Nhóm A có thể quan sát buổi Scrum Hàng ngày của Nhóm B để tăng cường việc chia sẻ thông tin.

### Sự điều phối

Chỉ thông qua Nói chuyện, Cộng tác thông qua mã nguồn, Tham quan, Open Space và các Cộng đồng.

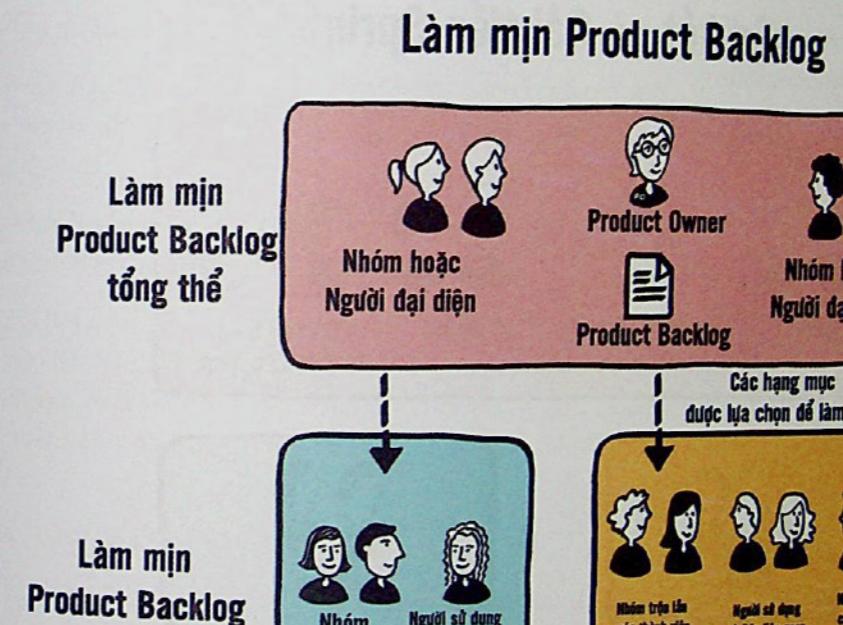


## Làm mịn Product Backlog tổng thể

Hoạt động làm mịn Product Backlog (PBR) tổng thể là tùy chọn với sự tham gia của Product Owner và thành viên đến từ tất cả các nhóm. Mục đích chính là quyết định xem nhóm nào phù hợp phát triển hạng mục nào rồi thực hiện như một buổi Làm mịn Product Backlog của một Nhóm Scrum. Đây cũng là cơ hội gia tăng sự gắn kết giữa Product Owner và tất cả các nhóm.

## Làm mịn Product Backlog

LeSS yêu cầu hoạt động này được tổ chức giống như hoạt động làm mịn Product Backlog khi triển khai Scrum trong một nhóm. Khác biệt duy nhất ở đây là hoạt động này có sự tham gia của tất cả các nhóm để tăng cường học tập và phối hợp với nhau.



## Short-ISH

Các nhóm Làm mịn Product Backlog 5-10% thời gian của một Sprint

<http://less.works> BY-NC

Chương 8: Scrum với quy mô lớn

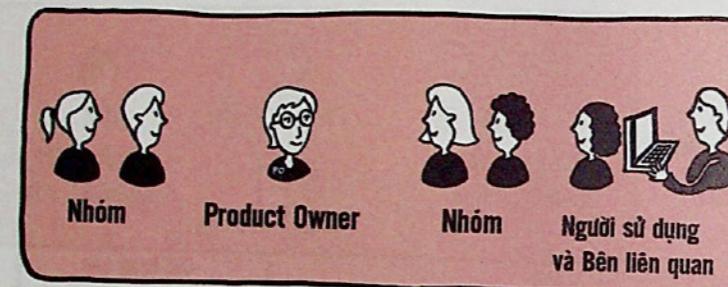
## Sơ kết Sprint

Sự kiện này có sự tham dự của Product Owner, các thành viên của tất cả các nhóm và khách hàng\người dùng và các bên liên quan khác. Phần trình diễn Phần tăng trưởng và các hạng mục mới được xem xét triển khai như một “phiên chợ” hoặc “báo cáo khoa học”: Trong một phòng lớn chia thành nhiều khu vực khác nhau, mỗi nhóm trình diễn, thảo luận về gói tăng trưởng mà mình phát triển ở một khu.

## Cải tiến Tổng thể

Sự kiện này không có trong Scrum chuẩn, mục đích của nó là tìm kiếm các cải tiến ở mức tổng thể thay vì chỉ tập trung vào một nhóm. Thời lượng là tối đa 45 phút/một tuần của Sprint. Sự kiện này có sự tham dự của Product Owner, ScrumMaster và đại diện luân phiên đến từ mỗi Nhóm.

## Sơ kết Sprint



## Cải tiến Sprint mức Nhóm



## Cải tiến Tổng thể



Sơ kết Sprint và Cải tiến Sprint trong LeSS,Ảnh: less.works.

<http://less.works>



## Những hiểu lầm phổ biến

### 1. Công ty lớn thì tổ chức phải quan liêu mới hiệu quả.

LeSS đã chỉ ra rằng điều này không đúng. Nhiều công ty có quy mô hàng nghìn người nhưng vẫn giữ được sự linh hoạt và vận dụng các nguyên tắc làm việc của Scrum.

### 2. Việc mở rộng Scrum tức là chỉ cần thêm các nhóm, và thêm giai đoạn thử nghiệm là đủ.

Như chúng ta thấy, một nỗ lực triển khai Scrum cho quy mô lớn đòi hỏi nhiều hơn thế. Nó đòi hỏi chúng ta phải có những nguyên tắc mới, cách tổ chức mới, một nền tảng quản trị mới. Đó là lý do chúng ta nên tìm kiếm đến những framework đã được minh chứng như LeSS hoặc Nexus.

### 3. LeSS là tốt nhất cho Scrum ở quy mô lớn.

Thực tế là không có phương pháp nào tốt nhất cho mọi tổ chức, mọi ngữ cảnh. Bạn sẽ phải tìm hiểu kỹ điều kiện của mình, cũng như hiểu biết sâu sắc một khung làm việc mới trước khi quyết định cái gì là phù hợp nhất cho mình.

### 4. Việc triển khai Scrum ở quy mô lớn có thể hoàn thành trong thời gian ngắn.

Còn tùy bạn định nghĩa như thế nào là xong. Bạn hãy đọc lại các ví dụ trong Chương 6 để thấy sự khó khăn trong nỗ lực mở rộng áp dụng Scrum tại doanh nghiệp. Bạn sẽ cần phải có được đầu tư thích đáng, trong thời gian đủ lâu mới tạo được kết quả đủ lớn.

### 5. Chỉ cần bộ phận phát triển sản phẩm của tôi Agile là được, còn lại thì cứ như cũ.

Thực tế là khi triển khai Agile ở quy mô lớn, tổ chức cũng cần phải thích ứng theo. LeSS gợi ý bạn nên tổ chức lại công ty xoay quanh các nhóm tự quản, những cộng đồng học tập. Vai trò của các nhà quản lý cũng phải thay đổi để xây dựng một hệ thống quản trị kiểu mới. Chỉ có cách tư duy hệ thống như vậy mới giúp bạn đạt được hiệu quả cao nhất. Không cải tiến cục bộ, hãy nghĩ đến toàn cảnh.



## 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, kỹ 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 Scrum ở quy mô lớn?
11. LeSS và Nexus khác nhau chỗ nào?
12. Bạn sẽ lựa chọn một khung làm việc dựa trên căn cứ nào?

# 9

## CÁC CHỦ ĐỀ NÂNG CAO

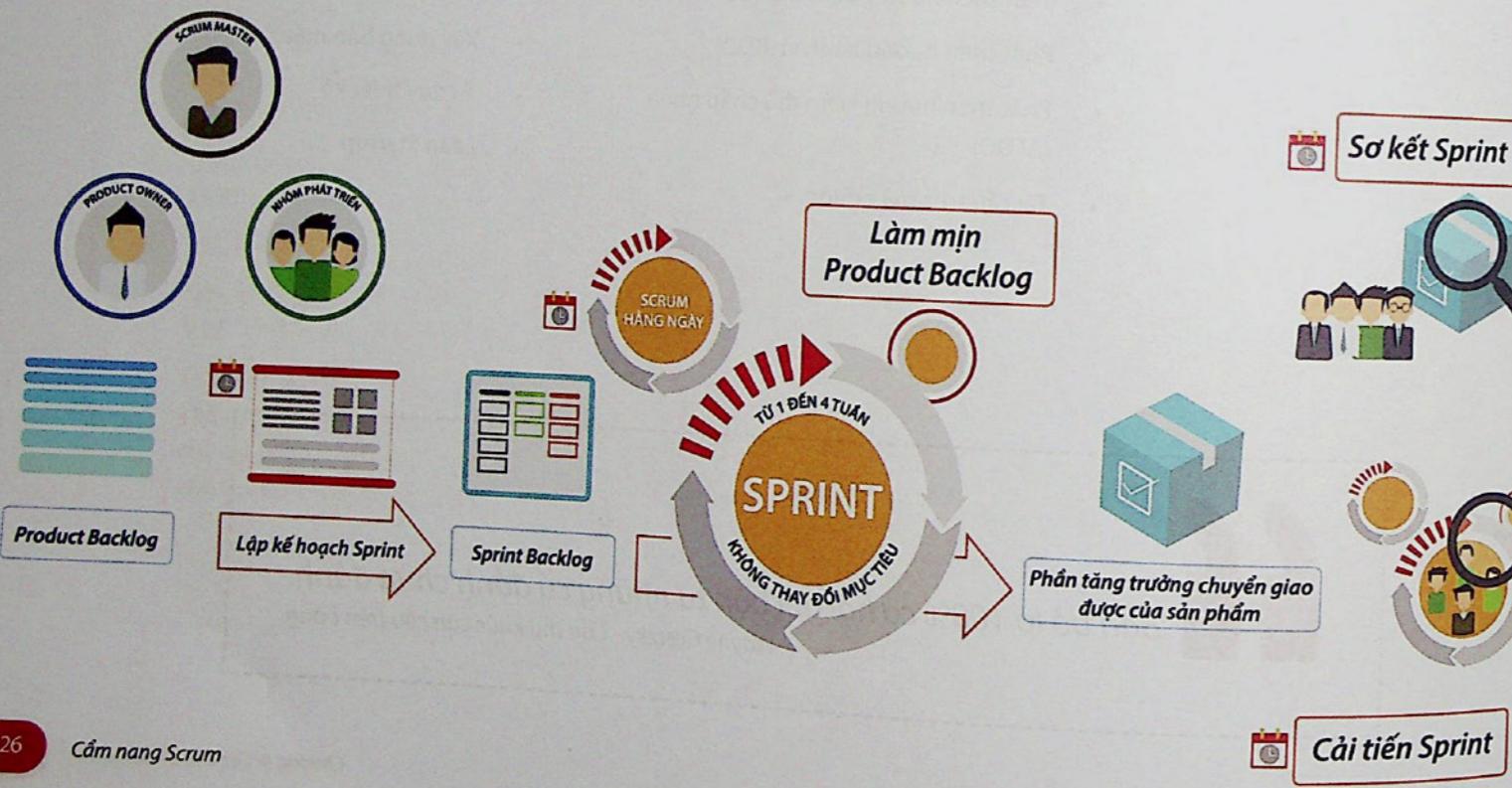
Trong chương này...

- Quản lý yêu cầu người dùng 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
- CI, CD và DevOps
- Luồng Tích hợp liên tục
- Xây dựng bản mẫu
- Tư duy thiết kế
- Lean Startup

“

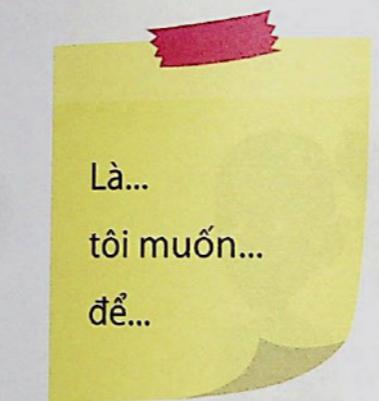
Bạn bỏ lỡ 100% cơ hội ghi bàn từ những cú đánh chưa đánh.  
Wayne Gretzky - Cầu thủ khúc côn cầu trên băng

Từ Chương 1 đến Chương 8, chúng ta đã thảo luận hầu như mọi ngóc ngách của Scrum. Có thể bạn sẽ hỏi “Thế sau Scrum thì là gì nữa?”. Sự thật thiếu sót nếu chúng ta không đề cập những thứ thường được các nhóm Scrum “cầm thêm” vào khung Scrum hoặc kết hợp cùng để nâng cao năng lực tổng thể. Quá trình phát triển hơn hai thập kỷ của những phương pháp Agile đã để lại cho các nhóm ngày nay hàng loạt kỹ thuật và công cụ hữu dụng trong tất cả các khâu, từ thu thập và quản lý yêu cầu người dùng, phân tích, thiết kế, phát triển sản phẩm cho đến trải nghiệm người dùng và tự động hóa.



Chương này giới thiệu những chủ đề nâng cao để giúp bạn tìm hiểu thêm. Phân nhiều các công cụ kỹ thuật trong chương này là dùng trong phát triển phần mềm nói riêng. Một số có thể vận dụng được ngay, số khác đòi hỏi bạn phải nghiên cứu thêm để vận dụng. Cũng có một số kỹ thuật có thể áp dụng trong các lĩnh vực khác, chẳng hạn như User Story, Ước tính linh hoạt, Tư duy thiết kế, Lean Startup,... 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 kỹ thuật nâng cao hơn là lúc bạn đã lên một tầm cao mới trong năng lực vận dụng Scrum.

## QUẢN LÝ YÊU CẦU NGƯỜI DÙNG VỚI USER STORY



Ví dụ:

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



User story template

Các hạng mục Product Backlog mô tả yêu cầu sản phẩm, Scrum không quy định cụ thể kỹ thuật cũng như hình thức để thể hiện các yêu cầu này, tuy nhiên trong thực tế kỹ 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 được viết bởi khách hàng, hoặc đại diện của khách hàng – những người thực sự hiểu nghiệp vụ và biết được tốt nhất các nhu cầu của mình đối với sản phẩm. User Story không đơn thuần là công cụ mô tả yêu cầu, mà còn là công cụ để giao tiếp, chia sẻ và cái “phanh hầm” trong phát triển. Scrum quy định Product Owner sở hữu các story (qua Product Backlog), nhưng đó không phải công việc chỉ riêng Product Owner làm.

Nhiều nhóm viết User Story trên Index Card để dễ thảo luận, nhiều nhóm khác lại thích dùng các phần mềm hỗ trợ (thường là các công cụ hỗ trợ tạo lập và quản lý Product Backlog, xem Chương 5) để tạo và quản lý 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 mua hàng online

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

## Làm rõ một User Story

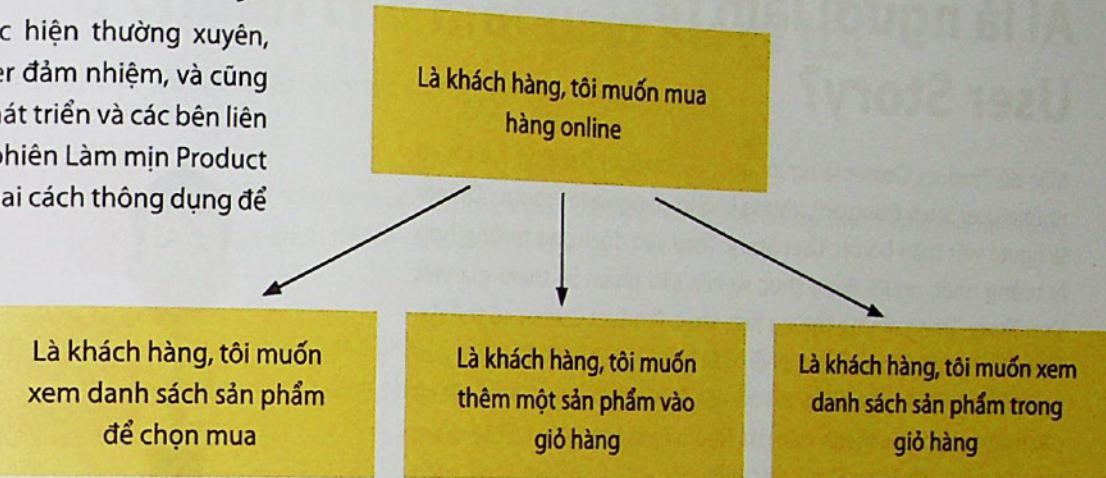
Việc làm rõ User Story được thực hiện thường xuyên, thông thường là do Product Owner đảm nhiệm, và cũng có thể có sự tham gia của Nhóm Phát triển và các bên liên quan. Ngoài ra có thể tổ chức các phiên Làm mịn Product Backlog để thực hiện việc này. Có hai cách thông dụng để chi tiết hóa một User Story.



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

Tính ngắn gọn của User Story có hai lợi ích chính, thứ nhất là nhóm không mất quá nhiều thời gian cho việc phải viết quá chi tiết những yêu cầu của sản phẩm ngay từ đầu như cách làm truyền thống, trong khi các tài liệu tinh này không đảm bảo được các bên sẽ hiểu đúng ý muốn của nhau, thay vào đó các nhóm chỉ cần mô tả đủ các tính năng chính để bắt đầu làm việc. Lợi ích thứ hai là giúp gia tăng giao tiếp thường xuyên giữa các bên, nhất là giữa Product Owner và Nhóm Phát triển. Giao tiếp thường xuyên giữa Product Owner và Nhóm Phát triển là cách tốt nhất để hạn chế những hiểu lầm khi mô tả các yêu cầu sản phẩm.

Ngôn ngữ “đời thường” được sử dụng trong User Story giúp cho những người không am hiểu về kỹ thuật (như khách hàng hoặc người dùng) có thể tự diễn đạt được mong muốn của mình. Khác với các cách mô tả yêu cầu truyền thống, thông thường các tài liệu này rất chi tiết và chứa nhiều mô tả kỹ thuật, gây khó khăn cho việc giao tiếp giữa một bên là kỹ thuật và một bên là phi kỹ thuật.



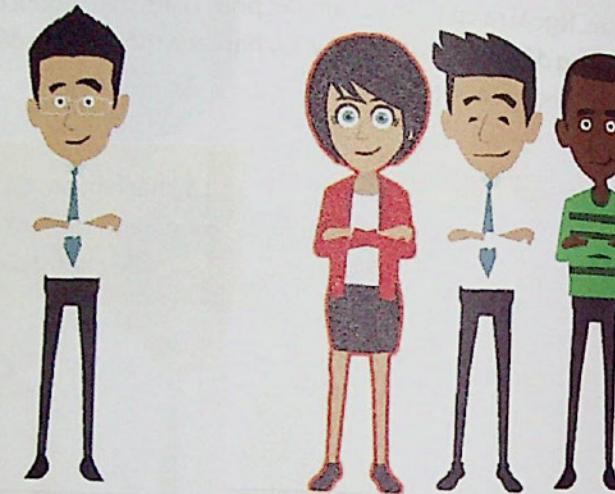
**Cách thứ nhất là phân tách một User Story thành các User Story khác nhau.**

**Cách thứ hai để chi tiết hóa một User Story đó là thêm tiêu chí chấp nhận vào cho nó.**

- Là khách hàng, tôi muốn xem danh sách sản phẩm để chọn mua.
- Tiêu chí chấp nhận:**
- Hiển thị ảnh, tên, giá sản phẩm
  - Sản phẩm mới được hiển thị trước, sản phẩm cũ được hiển thị sau
  - Mỗi trang hiển thị 20 sản phẩm

# AI LÀ NGƯỜI LÀM RA USER STORY?

Mặc dù Product Owner là người quản lý Product Backlog (và tất cả những User Story trong đó), nhưng không có nghĩa là Product Owner là người viết toàn bộ các User Story. Thay vào đó, trong trường hợp lý tưởng nhất, người dùng thực sự của sản phẩm sẽ tham gia 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 có thể tham gia vào việc viết User Story. Đặc biệt, đối với các nhóm phát triển sản phẩm khởi nghiệp, trong đó các thành viên Nhóm Phát triển cũng chính là những người đưa ra ý tưởng về sản phẩm, am hiểu tầm nhìn của sản phẩm và có tri thức về lĩnh vực mà mình đang tham gia, Nhóm Phát triển đóng vai trò quan trọng trong việc mô tả các tính năng của sản phẩm.



## VIẾT USER STORY KHI NÀO?

Hoạt động viết User Story 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 triển khai 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 (User Story Writing Workshop). 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 thông qua hoạt động làm mịn Product Backlog.

## 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

Negotiable – Đàm phán được

Valuable – Đáng giá

Estimable – Ước tính được

Sized appropriately  
– Kích thước phù hợp

Testable – Kiểm thử được

Hạn chế sự phụ thuộc lẫn nhau giữa các User Story, nhờ đó mà nhóm dễ dàng hơn trong việc lựa chọn thứ tự triển khai các User Story.

User Story không phải là một bản mô tả chi tiết và chật chẽ một tính năng của sản phẩm. Thảo luận và đàm phán là cách để hiểu đúng các nhu cầu thực tế và đưa ra các điều chỉnh hợp lý.

User Story cần có giá trị đối với người dùng, không phải đối với những người khác (chẳng hạn như nhà phát triển). Một cách để thể hiện được giá trị của User Story là viết đầy đủ ý nghĩa của User Story đó (về ở phía sau từ “để” trong định dạng “Là...tôi muốn...để...”). Rất nhiều nhóm thường bỏ qua mất về quan trọng này trong các User Story.

Không cần phải ước tính chính xác, nhưng cần có một giá trị ước tính để phục vụ cho việc lập kế hoạch. Muốn ước tính được thì User Story cần phải cung cấp đủ các thông tin cần thiết cho Nhóm Phát triển. Xem thêm phần Ước tính Linh hoạt.

Những User Story sắp được đưa vào sản xuất cần có kích thước nhỏ (đồng nghĩa với việc được mô tả rõ ràng hơn), những User Story chưa được đưa vào sản xuất trước mắt có thể có kích thước lớn hơn.

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. Kiểm thử được cũng là cách để đạt được sự đồng thuận giữa Nhóm Phát triển, Product Owner và các bên liên quan.

# 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. Kỹ thuật này được Nhóm Scrum sử dụng trong các phiên Xây dựng Product Backlog, Lập kế hoạch Sprint và Làm mịn Product Backlog. 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.

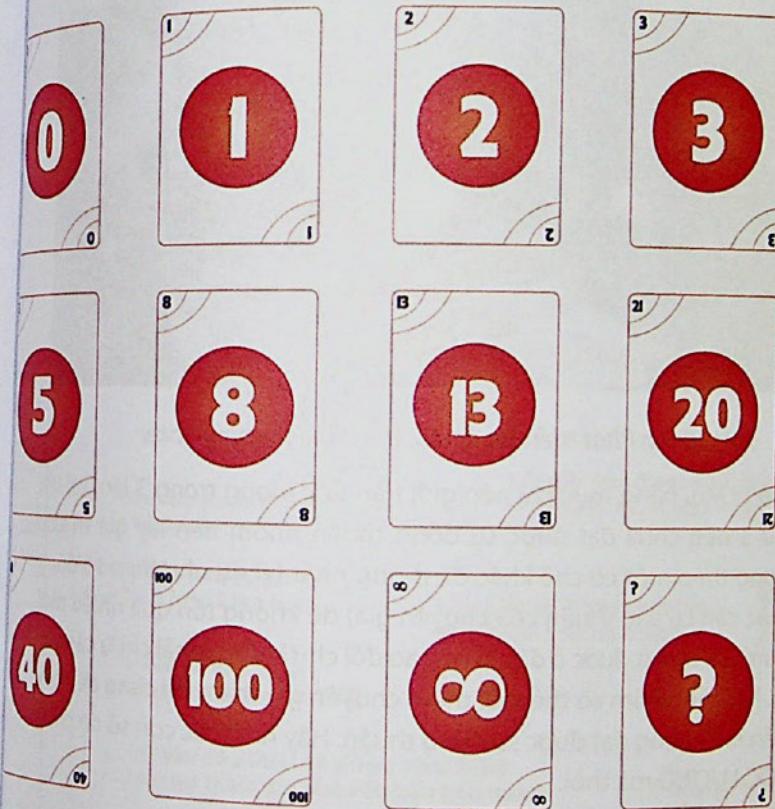
Ưu điểm đầu tiên của phương pháp ước tính linh hoạt đó là dựa trên dữ liệu thực tế của nhóm nên sẽ cho độ chính xác ổn định theo thời gian. Các nhóm sẽ căn cứ vào năng lực sản xuất của mình trong các khoảng thời gian gần nhất để thực hiện ước tính, dựa vào dữ liệu trong quá khứ thì chính xác hơn so với trường hợp dựa vào các tiên đoán.

Ưu điểm thứ hai của phương pháp ước tính linh hoạt đó là dễ thực hiện và tiết kiệm thời gian hơn so với các cách ước tính truyền thống. Việc so sánh tương đối giữa các hạng mục trong cùng một nhóm và cùng một sản phẩm sẽ dễ dàng hơn nhiều so với việc phải đưa ra một con số tuyệt đối về mặt thời gian như cách làm truyền thống. Nhờ vậy, các nhóm thường dễ dàng đi đến thống nhất và hoàn thành được việc ước tính trong một khoảng thời gian ngắn hơn.

Các nhóm Agile thành thục thường sử dụng đơn vị **Điểm Story** (story point) để xác định độ lớn cho các User Story hơn là sử dụng các đơn vị đo thời gian tuyệt đối như là ngày hoặc giờ.

## 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 hiệu quả được sử dụng phổ biến để thực hiện ước tính trong các nhóm 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 hỗ trợ tốt cho việc lập kế hoạch của nhóm. Nhóm Scrum có thể sử dụng Planning Poker để ước tính các hạng mục Product Backlog hoặc các công việc trong Sprint Backlog.

Để thực hiện kỹ thuật này, nhóm 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 bộ poker cho mình hoặc là mua các loại có sẵn trên thị trường. Planning Poker giúp cho hoạt động ước tính của các thành viên trở nên vui vẻ và dễ dàng hơn so với việc ước tính truyền thống nặng nề.



Planning Poker

# 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 thì việc ước lượng cho hạng mục đó đã xong. Ghi lại số điểm của hạng mục đó tương ứng với giá trị của cây poker được chọn.
- **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.



Hướng dẫn Ước tính Linh hoạt với Planning Poker



Một Nhóm Phát triển đang ước tính với Planning Poker

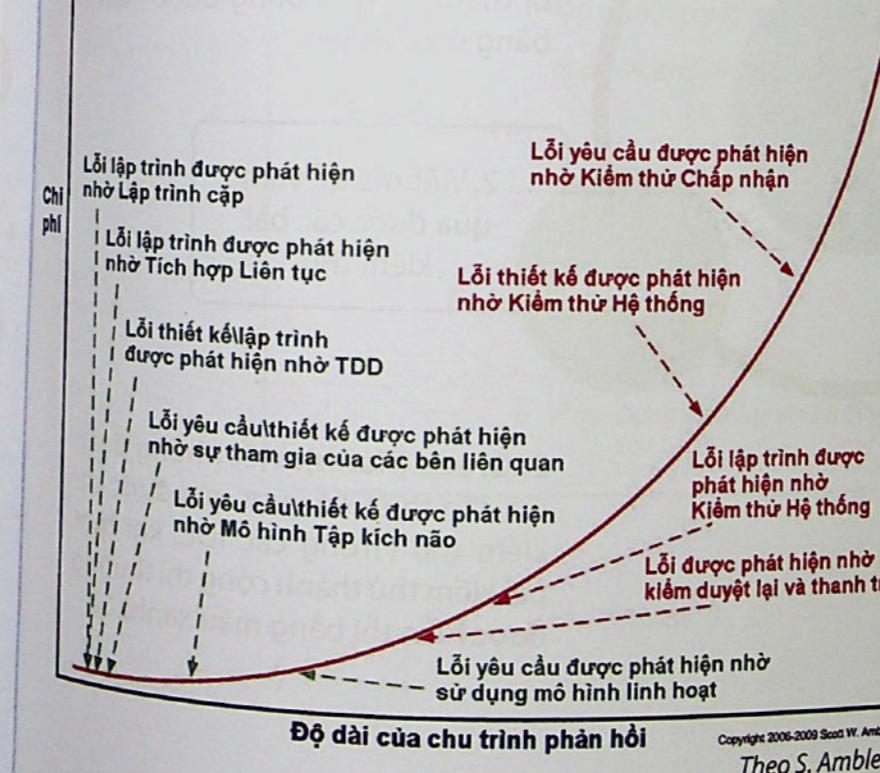
**Lưu ý:** Mỗi hạng mục chỉ nên giới hạn ước lượng trong 3 lần. Tới lần thứ 3 nếu chưa đạt được sự đồng thuận nhóm nên lấy giá trị ước lượng theo một cơ chế khác đã thống nhất (ví dụ như theo số đồng hoặc căn cứ vào ý kiến của chuyên gia) để không tốn quá nhiều thời gian. Giá trị đạt được ở đây là sự trao đổi chứ không phải chỉ là 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.



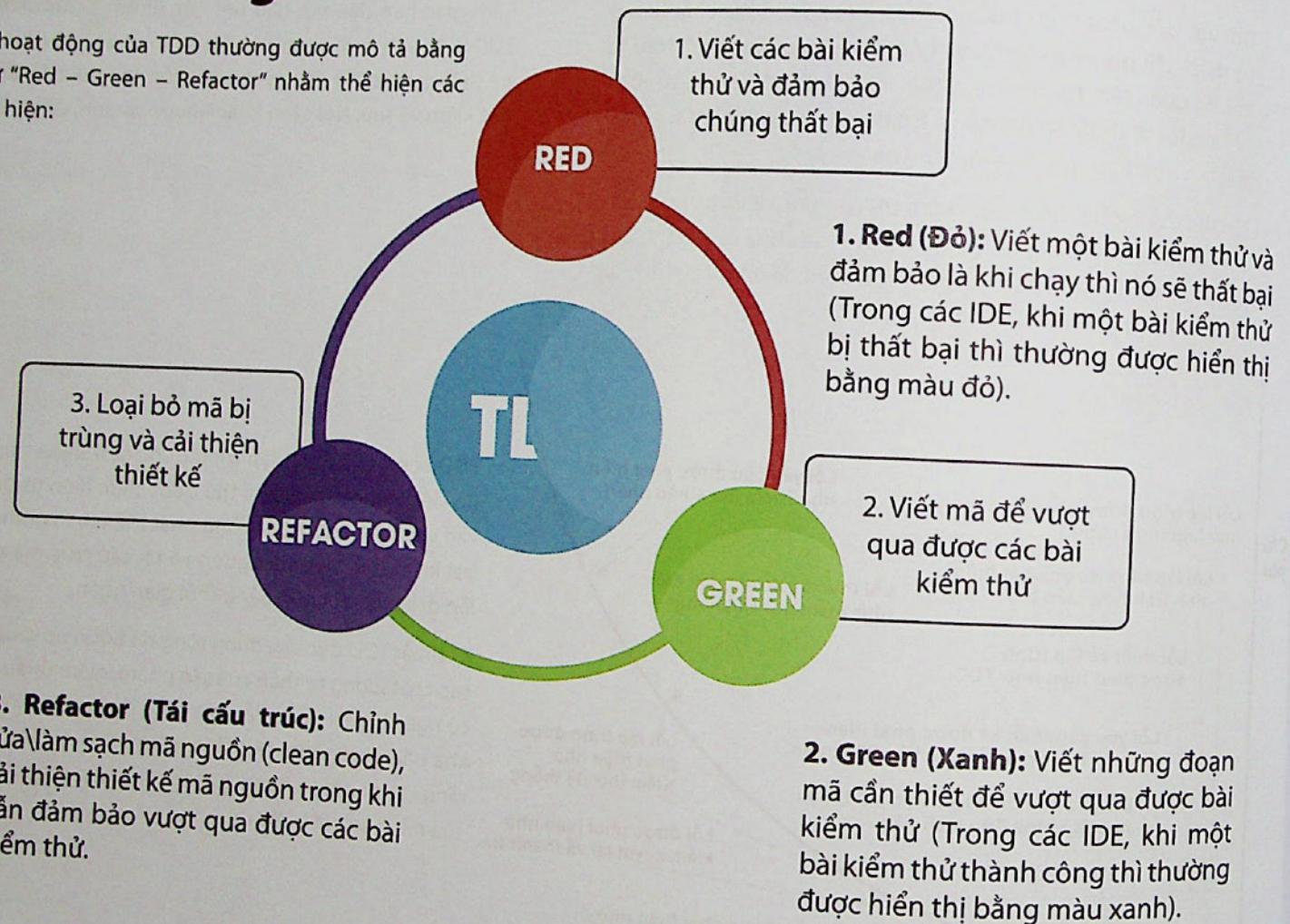
Cảm giác ban đầu với hầu hết các nhóm Scrum khi nhìn vào TDD là “tốn thời giờ”, tuy nhiên, khi nhìn từ góc độ dài hạn và toàn hệ thống, thì càng đầu tư cho cái “tốn thời giờ” ban đầu thì càng tiết kiệm về sau. Nói cách khác “muốn nhanh, cứ phải từ từ”.

TDD là một kỹ thuật lập trình 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, viết mã nguồn và tái cấu trúc mã nguồn) được lặp đi lặp lại trong khoảng thời gian ngắn.

Kỹ 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:



## 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 bao gồm một loạt các 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.*
- **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 những 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

dùng 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.

**K.I.S.S**  
Keep it simple stupid

## ATDD VÀ BDD

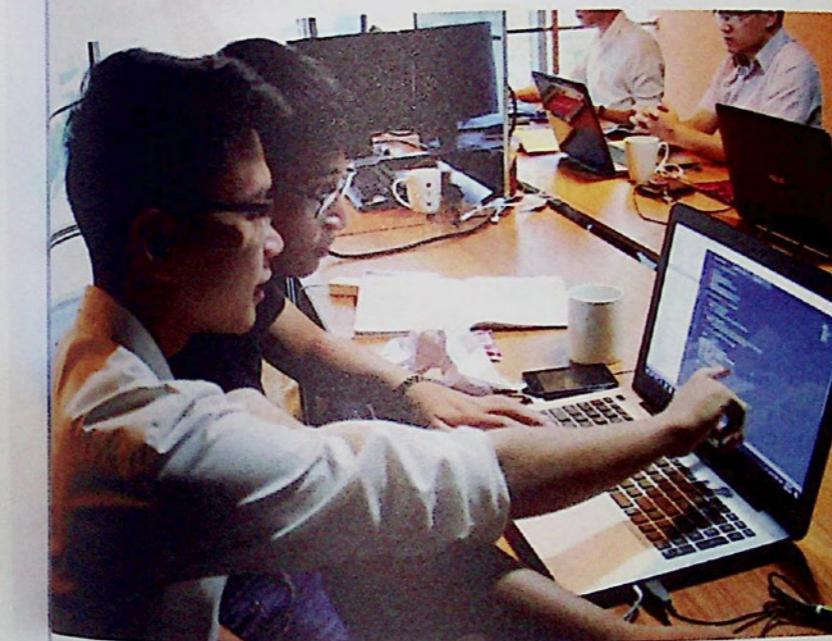
Một dạng mở rộng của TDD là Phát triển Hướng Kiểm thử Chấp nhận (Acceptance Test-Driven Development - ATDD). Đây là một kỹ thuật dựa trên sự cộng tác giữa khách hàng, nhà phát triển và kiểm thử viên, trong đó toàn bộ các thành viên tập trung thảo luận về các tiêu chí chấp nhận, sau đó chuyển chúng thành 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 Selenium, Cucumber, RSpec, Behat, SpecFlow, JBehave, Lettuce, Thucydidies, Spectacular, FitNesse, Concordion.



## LẬP TRÌNH CẶP

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 thông qua tranh luận. Tốc độ sản xuất có thể chậm hơn trong ngắn hạn nhưng xét toàn bộ hệ thống thì năng suất 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.

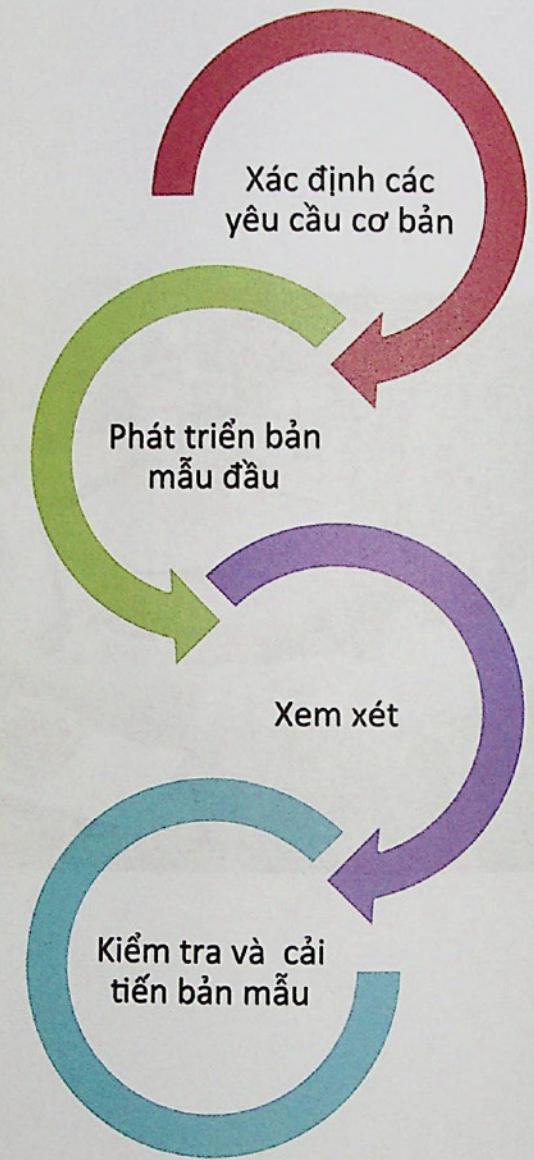
Lập trình cặp là một kỹ thuật rất tốt được sử dụng nhằm nâng cao chất lượng sản phẩm, giảm thiểu lỗi ngay từ rất sớm, gia tăng cộng tác trong nhóm, thúc đẩy học tập lẫn nhau và cổ xúy sở hữu và trách nhiệm tập thể.

## LÀM BẢN MẪU

Làm bản mẫu (Prototyping/Mockup) là kỹ thuật sớm đưa ra một phiên bản 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 phiên bản mẫu này của 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 mẫu này, chúng ta sẽ sớm có các phản hồi quan trọng và nhanh chóng đưa ra các điều chỉnh với chi phí thấp.

Các kỹ thuật và công cụ để làm bản mẫu tùy thuộc vào đặc điểm của từng 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.

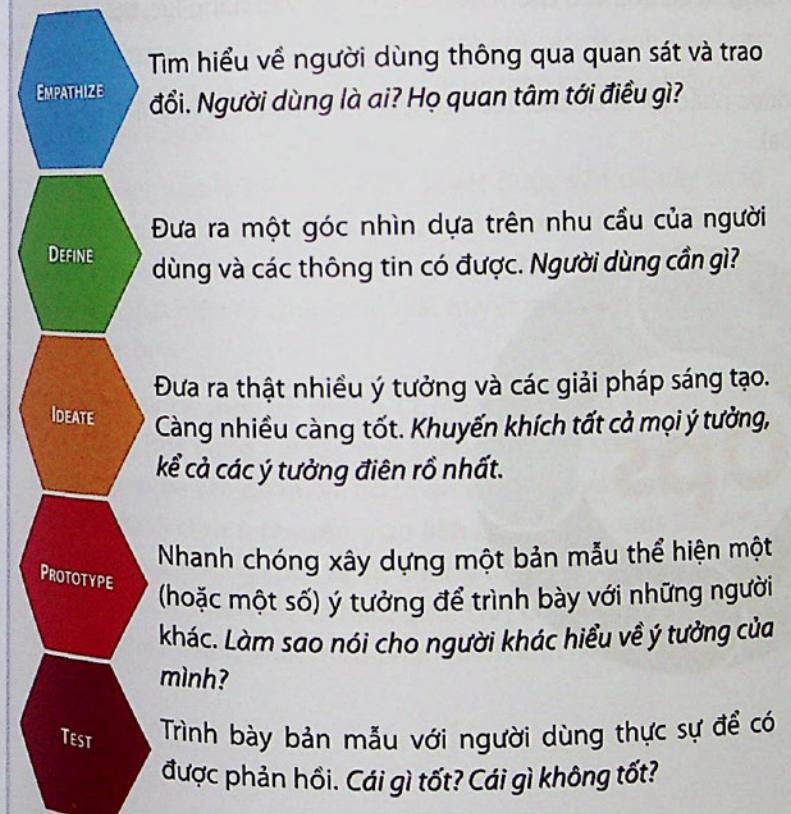
Trong các chu trình phát triển (như Design Thinking để cập dưới đây) các sản phẩm sáng tạo, việc làm mẫu là một phần vô cùng quan trọng.



## 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. Tên gọi Design Thinking xuất phát từ việc phương pháp này tương đồng với chiến thuật được sử dụng bởi các nhà thiết kế trong thực tế.

Cách thức xây dựng sản phẩm theo tư duy thiết kế bắt đầu bằng sự thấu cảm với người dùng để nắm bắt những nhu cầu sâu xa đôi khi không thể hiện được bằng các cách mô tả thông thường, từ đó



phát sinh các ý tưởng thiết kế và giải pháp để đáp ứng nhu cầu đó, rồi tạo lập các bản mẫu để hiện thực hóa ý tưởng, sau đó mang ra thử nghiệm để nhận về các phản hồi thực sự từ người dùng, làm mịn và tiếp tục cải tiến nhiều lần trước khi thành sản phẩm cuối tới tay người dùng, và không ngừng cải tiến trong các phiên bản tiếp theo của sản phẩm.

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).

Design Thinking là một quy trình để giải quyết vấn đề. Cách tiếp cận của Design Thinking là tập trung vào GIẢI PHÁP thay vì tập trung vào vấn đề. Design Thinking được ứng dụng trong khá nhiều lĩnh vực như kinh doanh<sup>1</sup>, giáo dục<sup>2</sup>, phát triển phần mềm và phần cứng<sup>3</sup>... Một Nhóm Scrum có thể kết hợp Scrum với Design Thinking để gia tăng sức mạnh cho việc tạo dựng những giải pháp sáng tạo.

1 Myerson, Jeremy. *IDEO: Masters of Innovation*. New York: teNeues, 2001.  
2 Archer L. B. et al. (1979) "Design in General Education". London: The Royal College of Art.  
3 <https://www.designorate.com/design-thinking-case-study-innovation-at-apple/>

# DevOps

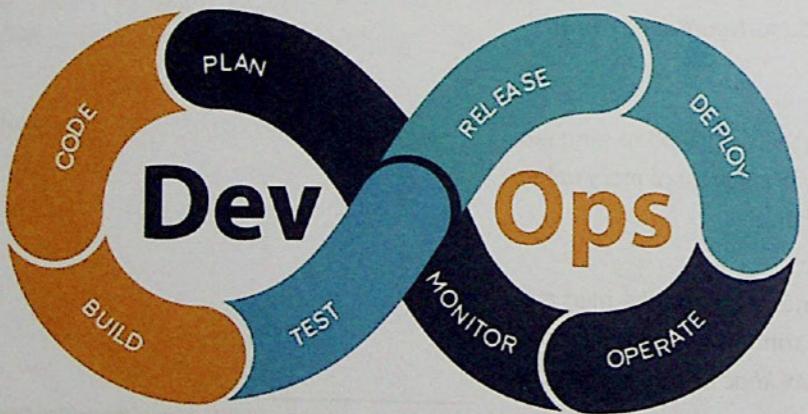
DevOps là một văn hóa và phương pháp nhấn mạnh sự cộng tác và giao tiếp giữa bộ phận phát triển sản phẩm và các bộ phận vận hành. Tên gọi DevOps xuất phát từ chữ viết tắt kết hợp của hai từ Development (Phát triển) và Operations (Vận hành). Việc nhanh chóng đưa các sản phẩm từ giai đoạn phát triển vào giai đoạn vận hành sẽ đáp ứng tốt hơn các yêu cầu cấp thiết của hoạt động kinh doanh.

Có một công thức nói lên tính chất này:

## DevOps = Development + Operations

Do DevOps là một sự thay đổi về văn hóa và hợp tác giữa nhóm phát triển sản phẩm, vận hành và kiểm thử/kiểm định chất lượng nên không có một "công cụ DevOps" duy nhất mà có cả chuỗi công cụ để triển khai DevOps. Việc thay đổi văn hóa, thói quen, quy trình và công cụ là cần thiết để triển khai thành công DevOps. Thông thường, việc lựa chọn các công cụ để đưa vào chuỗi này phụ thuộc vào năng lực, bối cảnh, cơ sở hạ tầng, v.v... của các nhóm nói riêng và của tổ chức nói chung.

Thông thường khi thảo luận, chia sẻ về DevOps các công cụ phổ biến được nhắc tới là Docker (container), Jenkins (Tích hợp liên tục - CI), Puppet (Mã nguồn như là Cơ sở Hạ tầng - IaC) và Vagrant (nền tảng ảo hóa).



# TÍCH HỢP LIÊN TỤC

Tích hợp Liên tục (Continuous Integration, viết tắt CI) là kỹ 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 kỹ 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 hóa 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.

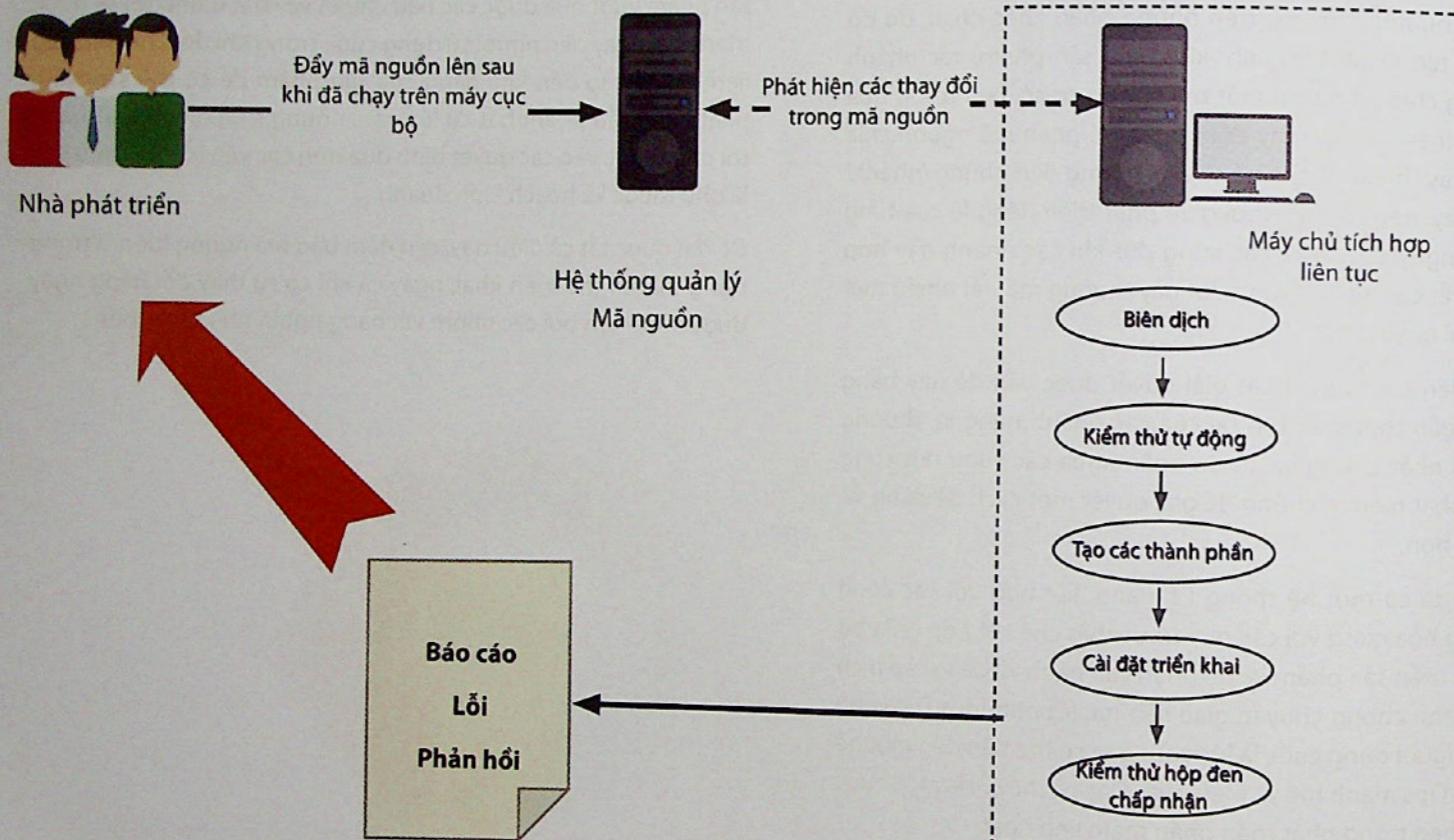
# TRIỂN KHAI LIÊN TỤC

Chuyển giao liên tục (Continuous Delivery) là một khái niệm khá gần gũi với Triển khai liên tục (Continuous Deployment), do đó trên thực tế nó thường được sử dụng nhầm lẫn thay thế cho nhau mà không để ý đến sự khác biệt chính: Triển khai liên tục tức là nếu sản phẩm vượt qua được các tiêu chuẩn về chất lượng thì sẽ được triển khai ngay đến người sử dụng cuối; Trong khi đó, Chuyển giao liên tục đề cập đến khả năng của sản phẩm để có thể sẵn sàng phát hành sản phẩm bất cứ lúc nào, nhưng thời điểm phát hành thì phụ thuộc vào các quyết định dựa trên các yếu tố khác, thường là phụ thuộc kế hoạch kinh doanh.

Để đạt được tất cả điều này, cần đảm bảo mã nguồn luôn ở trong trạng thái có thể triển khai, ngay cả khi có sự thay đổi hằng ngày được thực hiện bởi các nhóm với hàng nghìn nhà phát triển.

# LUÔNG TÍCH HỢP LIÊN TỤC

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



Một số phần mềm phổ biến phục vụ triển khai Tích hợp liên tục là Jenkins, Travis-CI, TeamCity, Bamboo, v.v...

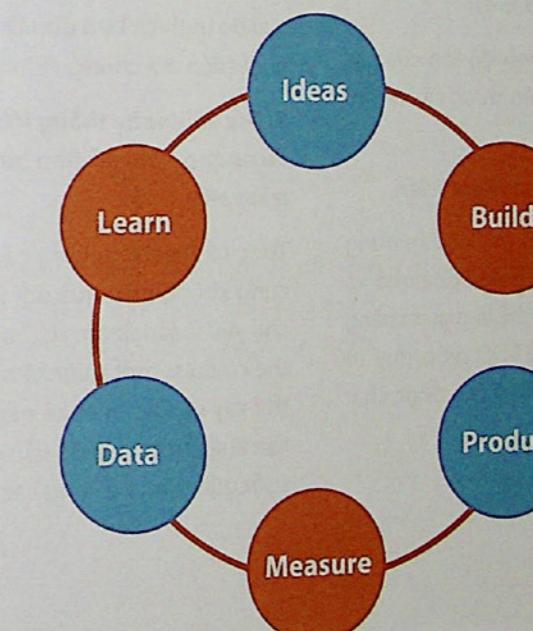
# LEAN STARTUP

Lean Startup (Khởi nghiệp Tinh gọn) là một phương pháp luận để phát triển các doanh nghiệp và sản phẩm dựa trên giả thuyết rằng nếu các công ty đầu tư thời gian của họ theo cách lặp đi lặp lại việc sớm khám phá ra các nhu cầu của khách hàng. Họ có thể giảm rủi ro thị trường, giảm đòi hỏi về vốn đầu tư dự án ban đầu và tăng cơ hội thành công cuối cùng.

Lean Startup được đề xuất lần đầu tiên vào năm 2008 bởi Eric Ries, sử dụng những trải nghiệm của ông trong việc thích ứng với các nguyên lý lean management đối với các công ty khởi nghiệp trong lĩnh vực công nghệ. Phương pháp này hiện đã được mở rộng để áp

dụng cho bất kỳ cá nhân, nhóm hoặc công ty nào muốn giới thiệu sản phẩm hoặc dịch vụ mới ra thị trường.

Lean Startup là sự kết hợp giữa phương pháp phát triển khách hàng (Customer Development), phát triển sản phẩm linh hoạt (Agile Development) và tạo dựng mô hình kinh doanh (Business Model Generation). Một nhóm Scrum nên biết thêm về Lean Startup để nắm được bức tranh lớn về mô hình kinh doanh, về cách thức kiểm định các giả thuyết về khách hàng. Từ đó mà nâng cao năng lực phát triển sản phẩm.





## Những hiểu lầm phổ biến

### 1. Phải trình bày toàn bộ các hạng mục Product Backlog theo định dạng User Story.

Thực tế: Scrum không quy định cụ thể cách thể hiện các hạng mục Product Backlog. User Story chỉ là một cách phổ biến được các nhóm Agile lựa chọn.

### 2. Các User Story chỉ được ước tính theo điểm (point).

Thực tế: Không có quy định cụ thể về đơn vị tính độ lớn của các User Story. Điểm (point) chỉ là một cách phổ biến được các nhóm Agile lựa chọn.

### 3. Triển khai TDD sẽ làm giảm hiệu quả của Nhóm Phát triển.

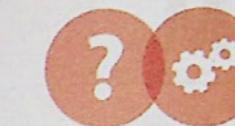
Thực tế: Điều này chỉ đúng trong các tình huống mà việc áp dụng TDD là không cần thiết (chẳng hạn như do sản phẩm quá nhỏ và hiển nhiên, thời gian sản xuất quá ngắn). TDD có thể làm giảm năng suất trong ngắn hạn, nhưng xét về dài hạn thì TDD giúp duy trì năng suất của nhóm về sau và đồng thời giúp nâng cao được chất lượng của sản phẩm.

### 4. Cần cố gắng sử dụng được càng nhiều kỹ thuật và công cụ thì càng tốt ngay tại thời điểm đầu tiên triển khai Scrum.

Thực tế: Việc sử dụng quá nhiều kỹ thuật và công cụ ngay tại thời điểm mà nhóm chưa hoạt động ổn định và hiệu quả sẽ gây rối cho nhóm và dễ dẫn đến tình huống không làm chủ được tình hình. Nên đưa dần các kỹ thuật và công cụ vào cho nhóm tùy theo từng giai đoạn thích hợp phụ thuộc vào năng lực của nhóm và tình hình phát triển nói chung.

### 5. Đối với các hệ thống lớn thì nhất thiết phải có một thiết kế đầy đủ ngay từ đầu để đảm bảo tính ổn định và thống nhất trong quá trình phát triển.

Thực tế: Việc xây dựng các thiết kế đầy đủ là không cần thiết và cũng không hiệu quả đối với cả những hệ thống lớn và nhỏ. Lý do chủ yếu là sản phẩm sẽ thay đổi rất nhiều trong tương lai, khó để có thể có được một thiết kế mà phù hợp cho tất cả các tình huống có thể xảy ra. Các thiết kế nên chỉ dừng ở mức vừa đủ để bắt tay vào sản xuất, trong khi đó vẫn đảm bảo các nguyên lý về thiết kế trong suốt quá trình bổ sung các tính năng mới và tái cấu trúc.



## CÂU HỎI ỨNG DỤNG

- Tại sao Nhóm Scrum cần áp dụng thêm những kỹ thuật Agile khác?
- Tại sao dùng User Story để mô tả yêu cầu khi nó quá ngắn không thể mô tả chi tiết?
- Ưu điểm của việc ước lượng linh hoạt so với ước lượng theo giờ?
- Tại sao khi chơi Planning Poker để ước lượng nỗ lực các thành viên lại cần suy nghĩ độc lập?
- Những giá trị mà TDD mang lại cho bạn là gì?
- Những khó khăn và cách vượt qua khi nhóm muốn áp dụng TDD?
- 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?
- Việc áp dụng lập trình cặp giải quyết được vấn đề gì ở nhóm của bạn?
- Làm bản mẫu có thể giúp nhóm bạn giải quyết được những vấn đề gì?
- Nhóm bạn có đang gặp vấn đề gì với việc tích hợp? Phương án giải quyết là gì?
- DevOps đem lại lợi ích gì cho nhóm hoặc tổ chức của bạn?
- Nhóm của bạn có những khó khăn nào khi triển khai một hệ thống Tích hợp Liên tục?

# 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**.

Denning, S. (2018). **The Age of Agile: How smart companies are transforming the way work get done**, Amyryllis Business

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. (2018). **Nexus Guide: The Definitive Guide to scaling Scrum with Nexus: The Rules of the Game**.

Schwaber, K., & Sutherland, J. (2017). **The Scrum guide-the definitive guide to Scrum: The rules of the game**.

Thoren M., (2017). **Agile People: A Radical Approach for HR & managers**, Lioncrest Publishing.

## 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 offshore 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 "**12th Annual State of Agile**" của VersionOne năm 2017.

## 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 hóa Hofstede (<https://www.geert-hofstede.com>)

eduScrum (<http://eduscrum.nl>)

Tài cấu trúc mã nguồn (<http://refactoring.com>)

Blog AgileBreakfast của Học viện Agile (<https://agilebreakfast.vn/>)

## Phụ lục 1 THUẬT NGỮ

### 5WHYs (5WHYs)

5Whys (hay Five Whys) là kỹ thuật dùng để điều tra nguyên nhân và các lựa chọn để giải quyết vấn đề, người điều tra sẽ hỏi 5 lần liên tiếp câu hỏi tại sao. Kỹ thuật này có nguồn gốc từ phương thức sản xuất Toyota hiện

nay đã được dùng phổ biến cho hầu hết các lĩnh vực.

#### Adaptation (Thích nghi)

Thích nghi là các hoạt động điều chỉnh được thực hiện bởi nhóm Scrum khi phát hiện các sai lầm trong quá trình phát triển. Các sai lầm này thường được phát hiện thông qua hoạt động thanh tra. Việc thích nghi có thể được thực hiện đối với sản phẩm, quy trình, công nghệ hay bất cứ phương diện nào khác. Các sự kiện của Scrum như: Lập kế hoạch Sprint, Sơ kế Sprint, Cài tiến Sprint và Scrum Hàng ngày đều hỗ trợ cho việc thích nghi được diễn ra thuận lợi.

#### 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.

#### Agile Estimation (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. Ước tính linh hoạt 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). Ước tính linh hoạt được thực hiện bằng cách so sánh độ lớn tương đối giữa các hạng mục với nhau dựa trên các dữ liệu thực tế đã trải qua.

Đơn vị thường được các nhóm sử dụng trong ước tính linh hoạt là điểm (point).

#### 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 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 kỹ thuật 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)

Thuật ngữ này thường đi kèm với từ chart (biểu đồ), trong Scrum có hai loại biểu đồ burndown được sử dụng phổ biến là Sprint Burndown và Release Burndown.

Biểu đồ Burndown thể hiện khối lượng công việc còn lại để hoàn thành kế hoạch. Nguồn dữ liệu thô được lấy từ Sprint Backlog và Product Backlog, khối lượng công việc còn lại được thể hiện ở trực tung còn thời gian (các ngày của Sprint, hoặc các Sprint) được thể hiện ở trực hoành.

### Continuous Integration (Tích hợp Liên tục)

Thuật ngữ này có tên ngắn gọn là CI. Đây là phương pháp mà nhóm liên tục ghép các phần của sản phẩm với nhau trong suốt thời gian phát triển 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 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.

### Continuous Delivery (Chuyển giao liên tục)

Chuyển giao liên tục là một khái niệm khá gần gũi với Triển khai liên tục, do đó trên thực tế nó thường được sử dụng nhầm lẫn thay thế cho nhau mà không để ý đến sự khác biệt chính: Triển khai liên tục tức là nếu sản phẩm vượt qua được các tiêu chuẩn về chất lượng thì sẽ được triển khai ngay đến người sử dụng cuối; Trong khi đó, Chuyển giao liên tục để cập đến khả năng của sản phẩm để có thể sẵn sàng phát hành sản phẩm bất cứ lúc nào, nhưng thời điểm phát hành thì phụ thuộc vào các quyết định dựa trên các yếu tố khác, thường là phụ thuộc kế hoạch kinh doanh.

### 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 kỹ năng, mà họ có thể chỉ có một số kỹ năng nhất định nhưng bổ sung đầy đủ cho nhau để tổng thể nhóm có được tất cả các kỹ năng cần thiết.

#### Daily Scrum (Scrum Hàng ngày)

Daily Scrum là buổi trao đổi ngắn của từng Nhóm Phát triển diễn ra hàng 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 nhóm và ScrumMaster có thể loại bỏ chúng. Có thể diễn ra các cuộc thảo luận ngay sau đó để điều chỉnh phần công việc tiếp theo nhằm tối ưu hóa Sprint.

#### Development Team (Nhóm Phát triển)

Nhóm Phát triển là một trong ba vai trò trong Nhóm Scrum, hai vai trò khác là ScrumMaster và Product Owner.

Nhóm Phát triển gồm tất cả những người đủ năng lực để chuyển giao phần tăng trưởng chất lượng (Increment) cuối mỗi Sprint. Nhóm Phát triển là liên chức năng.

#### DevOps (DevOps)

DevOps là một văn hóa và kỹ thuật trong kỹ nghệ phần mềm, trong đó cố gắng hợp nhất hoạt động phát triển và vận hành phần mềm. Khi triển khai kỹ 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) nhằm đáp ứng nhanh nhất các mục tiêu kinh doanh. 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 nổi lên tính chất này:

DevOps = Development + Operations.

#### 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 viết Định nghĩa Hoàn thành (Definition of Done) trước khi bắt đầu công việc.

#### Empirical Process (Quy trình thực nghiệm)

Quy trình thực nghiệm là quy trình mà kế hoạch và hành động dựa trên kinh nghiệm, không dựa trên yếu tố tiên lượng. Để quy trình thực nghiệm phát huy tác dụng, những tính chất sau phải được đảm bảo:

- Minh bạch (transparency)
- Thanh tra (inspection)
- Thích nghi (adaptation)

#### Estimated Work Remaining (Công việc Còn lại Được ước lượng)

Khối lượng công việc còn lại mà Nhóm Phát triển ước tính dành để làm việc trên một hạng mục nào đó. Ước tính này được cập nhật hàng ngày khi hạng mục đó đã được triển khai. Con số này là khối 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.

#### Glad-Sad-Mad (Glad-Sad-Mad)

Glad-Sad-Mad là một kỹ 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 mình cảm thấy hài lòng thì được phân loại vào mục Glad. Những hạng mục nào mà mình chưa cảm thấy hài lòng và có thể cải tiến được thì đưa vào mục Sad. Những hạng mục nào mà gây cảm trở nghiêm trọng và mình mong muốn loại bỏ nó ngay thì đưa vào mục Mad. Đây là một kỹ thuật được sử dụng phổ biến trong các buổi Cải tiến Sprint của Scrum.

#### Increment (Phần tăng trưởng)

Tính năng của sản phẩm được Nhóm Phát triển xây dựng trong từng Sprint có khả năng chuyển giao được. Phần tăng trưởng là cách gọi ngắn gọn của Phần tăng trưởng Tính năng Sản phẩm Có khả năng Chuyển giao được.

#### Increment of Potentially Shippable Product

#### Functionality (Phần tăng trưởng Tính năng Sản phẩm Có khả năng Chuyển giao được)

Tính năng của sản phẩm được Nhóm Phát triển xây dựng trong từng Sprint có khả năng chuyển giao được. Đây cũng là 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 ngắn gọn là Increment.

#### Inspection (Thanh tra)

Thanh tra là các hoạt động soi xét các tạo tác Scrum và tiến độ hướng đến mục tiêu để phát hiện các sai lệch không mong muốn. Hoạt động thanh tra được thực hiện thường xuyên nhưng không quá nhiều dẫn đến việc cản trở hoạt động sản xuất. Các sự kiện của Scrum như Lập kế hoạch, Sơ kết Sprint, Cải tiến Sprint và Scrum hàng ngày đều hỗ trợ cho việc thanh tra diễn ra được thuận lợi.

#### Kanban (Kanban)

Kanban là một phương pháp Agile được Taiichi Ohno

phát triển tại Toyota để nâng cao hiệu quả sản xuất. Kanban không chỉ được ứng dụng trong làm việc nhóm mà còn cho cả quản lý công việc cá nhân với tên gọi Kanban cá nhân (Personal Kanban).

Các nguyên lý của Kanban bao gồm:

- Trực quan hóa công việc
- Giới hạn số lượng công việc đang làm
- Tập trung vào luồng làm việc
- Cải tiến liên tục

Cần phân biệt Kanban với tư cách là một phương pháp, và Kanban như là một bảng công việc.

#### 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 từ 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. Họ thảo đã 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 đã có trước đó.

Thuật ngữ này thường được gọi ngắn gọn trong tiếng Việt là Tuyên ngôn Agile.

#### Planning Poker (Planning Poker)

Planning Poker là một kỹ thuật được sử dụng phổ biến trong các nhóm Agile để thực hiện ước tính. 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. 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.

#### Product Backlog (Product Backlog)

Một danh sách được sắp xếp chứa các yêu cầu với ước tính nỗ lực cần thiết để phát triển thành tính năng sản phẩm. Các hạng mục ở phía trên của Product Backlog được mô tả chi tiết hơn và 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 quản lý trong Product Backlog. Tất cả các hạng mục Product Backlog đều được ước tính. Độ chính xác của việc ước tính phụ thuộc vào mức chi tiết của hạng mục Product Backlog, các hạng mục ở phía trên của Product Backlog có thể được lựa chọn cho Sprint tiếp theo sẽ chi tiết và chính xác.

#### Product Backlog Refinement (Làm mịn Product Backlog)

Làm mịn Product Backlog là hoạt động thêm vào các chi tiết, ước lượng, và sắp xếp 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 soát cẩn thận. Có thể tách một hạng mục Product Backlog lớn thành các hạng mục Product Backlog nhỏ hơn.

#### Product Owner (Product Owner)

Là người chịu trách nhiệm về giá trị và hiệu quả của sản phẩm. Product Owner trực tiếp quản lý Product Backlog với mục tiêu tối ưu hóa giá trị của sản phẩm

và là người có tiếng nói cuối cùng trong việc đưa ra các quyết định liên quan đến sản phẩm.

#### Refactoring (Tái cấu trúc)

Tái cấu trúc là thay đổi ở cấu trúc bên trong mà không làm thay đổi hành vi bên ngoài của hệ thống. Hoạt động tái cấu trúc thường hướng đến việc nâng cao hiệu quả cộng tác của nhóm, khả năng duy trì và mở rộng của sản phẩm.

#### ROI (ROI)

ROI (Return On Invest) là tỷ suất hoàn vốn đầu tư hoặc tỷ lệ lợi nhuận. ROI được tính bằng tỷ lệ lợi nhuận ròng so với vốn đầu tư:

$$ROI = (\text{Doanh thu} - \text{Chi phí}) / \text{Chi phí}$$

#### Sailboat (Sailboat)

Sailboat là một kỹ thuật để tiến hành buổi cải tiến dựa trên hình ảnh ẩn dụ của một chiếc thuyền buồm. Nhóm xác định các neo (trở ngại) và gió (ý yếu tố tích cực) và chọn một vùng để cải tiến. Kỹ thuật này được sử dụng phổ biến trong các buổi Cải tiến Sprint của Scrum.

#### Scrum (Scrum)

Scrum là một khung làm việc dành cho phát triển, chuyển giao và duy trì các sản phẩm phức tạp. Tên gọi Scrum không phải là một từ viết tắt mà được lấy từ trò chơi bóng bầu dục, trong đó có một cơ chế nhằm đưa quả bóng đã ra ngoài quay trở lại sân.

#### 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, bao gồm:

- Product Backlog
- Sprint Backlog
- Phần tăng trưởng

### Scrumban (Scrumban)

Scrumban là một phương pháp Agile, cố gắng kết hợp giữa Scrum và Kanban nhằm tận dụng những ưu điểm của cả hai phương pháp này. Scrumban khuyến khích các đội phải liên tục cải tiến quy trình thông qua cơ chế của Kanban.

### ScrumBut (ScrumBut)

ScrumBut là thuật ngữ để chỉ đến những tình huống áp dụng Scrum nhưng lại làm khác đi ở một số quy tắc hoặc hoạt động so với những khuyến nghị được đưa ra trong Hướng dẫn Scrum. Một số ví dụ về ScrumBut: Tôi dùng Scrum nhưng không có ScrumMaster hoặc Product Owner; Tôi dùng Scrum nhưng Nhóm Phát triển không đảm bảo liên chức năng, vẫn phụ thuộc vào nhân sự ở các phòng ban khác; Tôi dùng Scrum nhưng không có Daily Scrum mà thay bằng Weekly Scrum.

### 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. ScrumMaster tạo ra môi trường và điều kiện làm việc tốt nhất cho nhóm Scrum, thúc đẩy sự cộng tác và cải tiến nhằm duy trì nhóm hoạt động ở hiệu suất cao.

### 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. Nhóm tự tổ chức không cần đến sự quản lý hoặc điều phối của bên ngoài mà vẫn hoàn thành tốt mục tiêu của mình. 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 (command-control).

### SpeedBoat (SpeedBoat)

SpeedBoat là kỹ thuật để tiến hành buổi cải tiến dựa trên hình ảnh ẩn dụ của một chiếc thuyền cao tốc. Mục đích của kỹ thuật này là xác định độ hài lòng của khách hàng đối với dịch vụ mà mình cung cấp. Nó được sử dụng phổ biến trong các buổi Cải tiến Sprint của Scrum.

### Sprint (Sprint)

Một phân đoạn, hoặc một chu trình lặp để sản xuất ra các 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)

Bảng công việc của Nhóm Phát triển trong một Sprint. Sprint Backlog bao gồm mục tiêu Sprint, danh sách các hạng mục Product Backlog được lựa chọn cho Sprint hiện tại và kế hoạch chi tiết các công việc để đạt được mục tiêu Sprint. 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.

### 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 hoàn thành các hạng mục Product Backlog. Sprint Backlog Task 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 (thường là một câu ngắn gọn) mô tả tổng quát và trọng tâm nhất phần tăng trưởng cần đạt được trong Sprint.

### 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, diễn ra ở cuối mỗi Sprint để Nhóm Phát triển phối hợp với Product Owner và các bên liên quan 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 (có thể dẫn đến thay đổi trên Product Backlog). Chỉ những tính năng của sản phẩm đã được hoàn thành thì mới được trình diễn hoặc dùng thử.

### Stakeholder (Bên liên quan)

Những người có quyền lợi trong kết quả của dự án hoặc sản phẩm, có thể là vì họ đã đầu tư, hoặc sẽ dùng, hoặc sẽ ảnh hưởng đến họ, chẳng hạn: Khách hàng, người dùng, nhà đầu tư, các cấp quản lý...

### Team (Nhóm)

Cách viết tắt của Development Team trong một số tài liệu. Team trong Scrum là liên chức năng.

### 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 xen vào nhau.

### Time-box (Khung thời gian)

Một khoảng thời gian cố định để thực hiện một sự kiện hoặc hoạt động nào đó. Ví dụ, một buổi Scrum Hàng ngày được đóng khung trong 15 phút và bắt buộc kết thúc sau 15 phút bắt kể kết quả như thế nào. Các sự kiện Lập kế hoạch Sprint, Sơ kết Sprint, Cải tiến Sprint đều có thể kết thúc sớm hơn khung thời gian cho phép. Tuy nhiên, các Sprint thì phải có độ dài chính xác như đã quy định.

### Transparency (Minh bạch)

Minh bạch là một trong ba trụ cột của thuyết thực nghiệm được triển khai trong Scrum, hai trụ cột khác là Thanh tra (Inspection) và Thích nghi (Adaptation). Minh bạch có nghĩa là tất cả các phương diện của quá trình sản xuất phải được dễ dàng nhìn thấy bởi tất cả

các bên có liên quan. Một trong số các yếu tố quan trọng để có được tính minh bạch đó là các bên cùng sử dụng một ngôn ngữ và các tiêu chuẩn chung được định nghĩa và đồng thuận, đảm bảo tất cả đều tin tưởng và có chung một cách hiểu về các phương diện.

### User Story (User Story)

Câu chuyện người dùng (User Story) là một tài liệu đơn 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. User Story thường có định dạng: Là <tên của vai trò>, tôi muốn <chức năng của sản phẩm> để <mục đích mong muốn đạt được>.

### User Story Point (Điểm User Story)

Điểm User Story là một đơn vị tương đối dùng để đo độ lớn của các hạng mục User Story. Điểm User Story không đại diện cho một đại lượng thời gian cụ thể (như giờ hoặc ngày) mà thể hiện mối tương quan về độ lớn giữa các hạng mục User Story. Điểm User Story được sử dụng phổ biến trong phương pháp Ước tính linh hoạt (Agile Estimation).

### 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.

Đối với những nhóm mới áp dụng Scrum, các vai trò và công việc trong Scrum là khá mới. Để giúp cho các bạn dễ dàng bắt đầu và biết được công việc cụ thể của mình trong từng thời điểm, hãy sử dụng các danh mục kiểm tra được đính kèm sau đây như một gợi ý và hướng dẫn cơ bản nhất.

Trong phụ lục này:

- Danh mục kiểm tra của Product Owner
- Danh mục kiểm tra của ScrumMaster
- Danh mục kiểm tra Lập kế hoạch Sprint
- Danh mục kiểm tra Sơ kết Sprint
- Danh mục kiểm tra Cải tiến Sprint



Các danh mục kiểm tra

### 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

### 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 lý được không? Để duy trì số lượng hạng mục trong giới hạn có thể quản lý 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 cách bắn 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 tỷ 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 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 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, để 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 tồn tại 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 lý của nhóm không, các công cụ đó có tiện dụng không?
- Những công cụ quản lý 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 lý trong nhóm.
- Các thành viên nhóm có tự nguyện nhận nhiệm vụ không?
- Sự cẩn thận của việc hoàn trả nợ kỹ 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?

### 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?

# BẢNG CHỈ MỤC

## 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 phản 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 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?

5WHYs 77

A

adaptation 55

adapt. See adaptation

Agile 3, 9, 10, 16, 17, 18, 24, 32, 34, 38, 39, 40, 41, 46, 270

Agile Coach 3, 9, 10, 16, 17, 18, 24, 32, 34, 38, 39, 40, 41, 46, 270

artifact 83

ATDD 10, 229, 244, 255

B

BDD 10, 229, 244, 255

bên liên quan 81, 83, 264

bỏ phiếu chấm 10, 229, 244, 255

brain-storming 10, 229, 244, 255

Burndown 83, 165, 166, 167, 169, 170, 171, 172, 174, 263

C

Cải tiến Sprint 52, 81, 105, 129, 130, 264

chó chăn cừu 10, 229, 244, 255

chồng lắp 58, 59

CI 10, 229, 244, 255

CMMI 10, 229, 244, 255

command-and-control 10, 229, 244, 255

công cụ 10, 22, 41, 52, 80, 81, 83, 114, 137, 152, 262

cộng đồng thực hành 10, 229, 244, 255

Continuous Integration 10, 229, 244, 255

cross-functional 10, 229, 244, 255

Customer-Centric 10, 229, 244, 255

Cynefin 10, 36, 229, 244, 255

D

Daily Scrum 10, 229, 244, 255

dashboard 10, 229, 244, 255

design thinking 10, 229, 244, 255

Development Team 10, 229, 244, 255

DevOps 10, 229, 244, 255

Done 10, 229, 244, 255

dự án 10, 18, 19, 23, 32, 34, 142, 182, 229, 244, 255

dựa theo kế hoạch 10, 32, 229, 244, 255

Định nghĩa Hoàn thành 10, 229, 244, 255

E

eduScrum 10, 47, 229, 244, 255

empirical process control 10, 229, 244, 255

estimation 10, 229, 244, 255

eXtreme Programming 10, 229, 244, 255

F

framework 10, 229, 244, 255

Function Point 10, 229, 244, 255

G

giá trị Scrum 10, 229, 244, 255

GIÁ TRỊ SCRUM 55

Glad-Sad-Mad 10, 131, 229, 244, 255

H

Hạng mục Product Backlog 10, 229, 244, 255

hoàn thành 19, 23, 34, 45, 57, 80, 81, 115, 153, 262, 263, 264

Hofstede 10, 229, 244, 255

huấn luyện 10, 229, 244, 255

I

Impediment Backlog 78, 120

Increment 78, 120

incremental 78, 120

inspect 78, 120

inspection 55

iteration 78, 120

K

kaizen 78, 120

Kanban 78, 120, 153

khởi nghiệp công cụ 78, 120

khung thời gian 60, 78, 120, 263, 264

KLOC 20, 78, 120

L

làm mìn Product Backlog 78, 120

Lập kế hoạch Sprint 52, 78, 105, 116, 120, 260, 263

Lập trình cặp 78, 120, 245

Lean 78, 120

Lean Startup 78, 120

LeSS 78, 120

liên chức năng 78, 120

Lightweight 78, 120

Limit WIP 78, 91, 120

luồng 78, 120

M

minh bạch 78, 80, 83, 120, 262

mục tiêu Sprint 80, 262, 263

Mục tiêu Sprint 78, 120

N

Nexus 78, 120, 216, 218

nguyên nhân gốc rễ 78, 120

Nhóm 78, 120

Nhóm Phát triển 34, 52, 74, 78, 83, 89, 90, 108, 120, 150, 238, 263, 264  
Nhóm Scrum 9, 78, 114, 120, 160, 182, 264  
nợ kỹ thuật 78, 80, 81, 120, 262

**P**  
Pair Programming 78, 120  
Personal Kanban 78, 120  
phân đoạn 45, 78, 120  
phân tán 11, 78, 120, 201, 202, 203  
phân tăng trưởng 34, 155, 264  
Phân tăng trưởng 78, 120  
phát hành 34, 78, 81, 120  
Phát triển Hướng Hành vi 78, 120  
Phát triển Hướng Kiểm thử 78, 120, 239  
Phát triển Hướng Kiểm thử Chấp nhận 78, 120  
Phát triển Phần mềm Tinh gọn 78, 120  
phức tạp 78, 120, 161  
plan driven 78, 120  
plan-driven 32  
Planning Poker 78, 120, 237, 238  
PM 78, 79, 120  
point 78, 120  
Product Backlog 34, 78, 80, 81, 83, 120, 138, 139, 140, 141, 142, 144, 146, 147, 148, 152, 153, 175, 217, 262, 263  
Product Owner 52, 76, 78, 81, 82, 83, 84, 85, 108, 120, 182, 260, 263, 264

**Q**  
quản lý vật 78, 120  
quản trị dự án 78, 120  
quy trình 78, 120  
quy mô lớn 11, 78, 120, 214  
quy tắc làm việc 78, 120  
quy trình 22, 23, 37, 161, 219

**R**  
Release Burndown 78, 120, 170, 171, 172, 174  
ROI 34, 78, 83, 120  
rủi ro 57, 78, 83, 120

**S**  
SAFe 78, 120  
Scrumban 78, 120  
ScrumBut 78, 120  
Scrum Hàng ngày 52, 78, 105, 120, 121  
ScrumMaster 52, 73, 74, 75, 76, 77, 78, 79, 80, 81, 120, 260, 262  
Scrum of Scrum 78, 120  
Scrum phân tán 11  
Scrum Phân tán 78, 120  
self-organized 78, 120  
servant leader 78, 120  
SMART 78, 114, 120  
Sơ kết Sprint 52, 78, 81, 105, 120, 123, 124, 260, 264  
SpeedBoat 130, 132  
Sprint 34, 43, 52, 57, 60, 61, 80, 81, 83, 84, 105, 108, 109, 110, 111, 114, 116, 123, 124, 126, 129, 130, 138, 140, 149, 150, 152, 153, 155, 163, 165, 166, 167, 169, 175, 217, 260, 262, 263, 264  
Sprint Backlog 52, 138, 149, 150, 152, 153, 175, 263  
Sprint Burndown 78, 120  
Sprint Goal 78, 120  
Sprint Planning 78, 120  
Sprint Retrospective 78, 120  
stakeholder 78, 120  
Standup Meeting 78, 120

**T**  
tâm nhìn 188  
tảng băng trôi 78, 120, 191  
tăng trưởng 34, 43, 52, 62, 78, 120, 138, 155,

156, 175, 264  
tạo tác 10, 78, 83, 120, 137, 263  
TDD 239, 240  
technical debt 78, 120  
thanh tra 52, 55, 77, 78, 120  
thay đổi 22, 23, 29, 34, 51, 78, 81, 108, 188  
thích nghi 78, 120  
thí điểm 78, 120  
thuyết quản lý thực nghiệm 78, 120  
Tích hợp Liên tục 78, 120  
tiến độ 23, 83, 140, 173  
tiếp cận lặp 78, 120  
time-box 78, 120  
tinh gọn 78, 120  
tổ chức học tập 126  
transparent 78, 120  
Trao quyền 188  
Triển khai Liên tục 78, 120  
Triết lý Agile 78, 120  
trụ cột 52, 55  
trực quan 50  
tuần tự 57, 58  
Tuckman 78, 120  
tự tổ chức 23, 52, 78, 120  
Tuyên ngôn Agile 17, 24

**U**  
ước tính 45, 140, 152, 237, 238  
User Story 144, 162, 233, 234

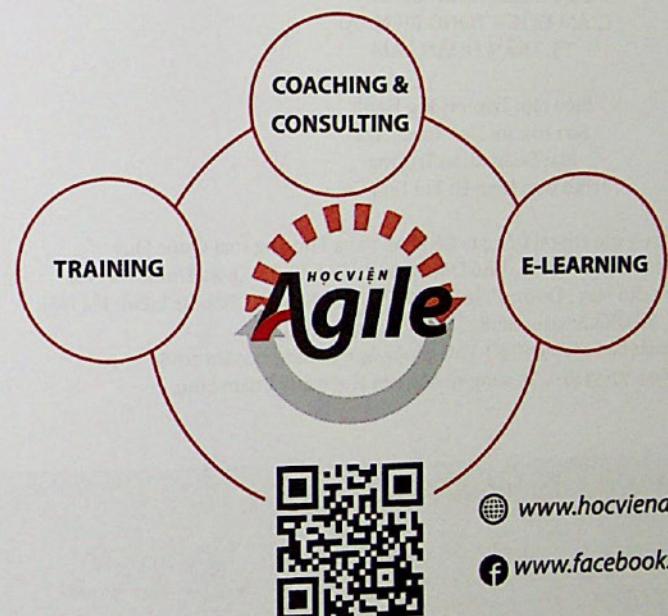
**W**  
waterfall 20, 57

**X**  
XP 78, 120

## 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 lý 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, tinh bắng hữu bền chặt, và không ngừng đổi mới sáng tạo.



CÔNG TY TNHH MTV  
NHÀ XUẤT BẢN THẾ GIỚI

Trụ sở chính:

Số 46, Trần Hưng Đạo, Hoàn Kiếm, Hà Nội  
Tel: 0084.24.38253841 - Fax: 0084.24.38269578

Chi nhánh:

Số 7, Nguyễn Thị Minh Khai, Quận 1, TP. Hồ Chí Minh  
Tel: 0084.28.38220102  
Email: marketing@thegioipublishers.vn  
Website: www.thegioipublishers.vn

## CẨM NANG SCRUM

(Làm chủ phương pháp năng suất và sáng tạo gấp đôi)

Chủ trách nhiệm xuất bản:  
GIÁM ĐỐC - TỔNG BIÊN TẬP  
TS. TRẦN ĐOÀN LÂM

Biên tập: Trịnh Hồng Hạnh  
Sửa bản in: Ngô Tuyết Nga  
Bìa: Đoàn Xuân Trường  
Trình bày: Nguyễn Thị Kim Cúc

In 1.000 bản, khổ 23.5 x 19 cm tại Công ty Cổ phần In và Thương mại Quốc Duy  
Địa chỉ: Số 9, ngách 130/1, ngõ 130 phố Đốc Ngữ, P. Vĩnh Phúc, Q. Ba Đình, Hà Nội  
Xưởng sản xuất: Trụ cầu N25, Đường Tân Xuân, P. Đông Ngạc, Q. Bắc Từ Liêm, Hà Nội.

Số ĐKQB: 4024-2018/CXBIPH/03-283/ThG  
Quyết định xuất bản số: 1293/QĐ-ThG cấp ngày 05 tháng 12 năm 2018  
ISBN: 978-604-77-5348-2. In xong và nộp lưu chiểu quý I năm 2019.

### CÔNG TY CP VĂN HÓA TRUYỀN THÔNG SỐNG

Trụ sở chính: 176 Thái Hà, P. Trung Liệt, Q. Đống Đa, TP. Hà Nội | Tel: (024) 3 722 62 34 | Fax: (84-24) 3 722 62 37  
VPGD: Phòng 308, tầng 3, Tòa nhà số 14 Pháo Đài Láng, P. Láng Thượng, Q. Đống Đa, TP. Hà Nội | Tel: (024) 6 656 60 58  
Phòng kinh doanh: Tel: (84-24) 3 773 8857 | Email: sale@alphabooks.vn  
VPĐD phía Nam: 138C Nguyễn Đình Chiểu, P.6, Q.3, TP. Hồ Chí Minh | Tel: (028) 3 822 03 34

### DƯƠNG TRỌNG TẤN



là một trong những người tiên phong truyền bá tri thức Agile tại Việt Nam, sáng lập viên của AgileVietnam, nhà sáng lập Học viện Agile và CodeGym. Trước đó, ông Tấn tham gia giảng dạy và giữ nhiều chức vụ quản lý tại khối giáo dục FPT.

### NGUYỄN VIỆT KHOA



có hơn 14 năm kinh nghiệm phát triển phần mềm trên nhiều nền tảng công nghệ và quy trình khác nhau. Ông là sáng lập viên HanoiScrum, TapChiLapTrinh.vn, miệt mài truyền bá các tri thức về lập trình linh hoạt. Ông từng nhiều năm giảng dạy và quản lý trong khối giáo dục FPT, hiện đang là Giám đốc Vận hành của Agilead Global.

### PHẠM ANH ĐỐI



là một trong những chuyên gia Scrum được chứng nhận đầu tiên tại Việt Nam, là Agile Coach được chứng nhận bởi ICAgile. Hiện ông là Giám đốc Học viện Agile, ngoài việc chính là tư vấn, đào tạo và huấn luyện cho các doanh nghiệp vừa và nhỏ tại Hà Nội, ông Đội còn là người tham gia tổ chức các hội thảo Agile có uy tín như: Agile Tour, Scrum Gathering, Agile Vietnam Conference, v.v..

### NGUYỄN KHẮC NHẬT



là một chuyên gia lập trình đa nền tảng. Ông Nhật từng là giảng viên IT tại Đại học FPT, lập trình viên và quản trị dự án tại FPT Software, đồng sáng lập TapChiLapTrinh.vn. Tham gia sáng lập Học viện Agile, ông Nhật mong muốn mang tri thức Agile lan tỏa sâu rộng hơn trong cộng đồng. Hiện ông Nhật là Phó Tổng giám đốc CodeGym Việt Nam - hệ thống đào tạo lập trình kiểu mới.

Hành trình vạn dặm bắt đầu với một bước chân nhỏ - Lão Tử -

HỌC VIỆN  
**Agile**

SỐNG | QUẢN TRỊ

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



CÔNG TY CP VĂN HÓA TRUYỀN THÔNG SỐNG

Trụ sở chính: 176 Thái Hà, Đống Đa, Hà Nội | Tel: (024) 3 722 62 34 | Fax: (84-24) 3 722 62 37  
VPGD: Phòng 308, tầng 3, Tòa nhà số 14 Pháo Đài Láng, P. Láng Thượng, Q. Đống Đa, TP. Hà Nội | Tel: (024) 6 656 60 58  
Phòng kinh doanh: Tel: (84-24) 3 773 8857 | Email: sale@alphabooks.vn  
VPĐD phía Nam: 138C Nguyễn Đình Chiểu, P.6, Q.3, TP. Hồ Chí Minh | Tel: (028) 3 822 03 34



Tìm mua ebook của Sống tại: waka.vn, mikiapp.com, Alezoo.com

Giá: 199.000đ