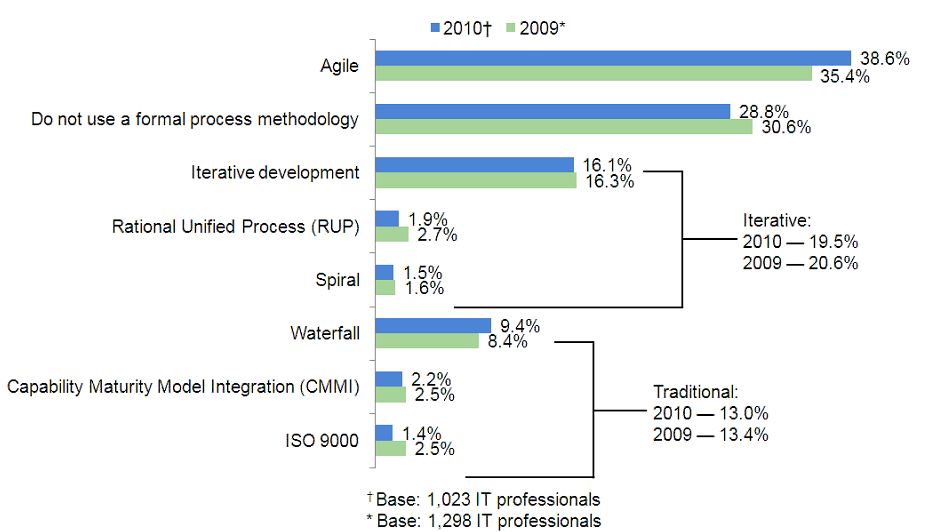
# Giới thiệu

Phát triển phần mềm linh hoạt (agile software development – gọi tắt là Agile) là một triết lí cùng với nhóm các phương pháp và phương pháp luận phát triển phần mềm dựa trên các nguyên tắc phát triển phân đoạn lặp (iterative) và tăng trưởng (incremental). Agile thường sử dụng cách lập kế hoạch thích ứng (adaptive planning), việc phát triển và chuyển giao theo hướng tiến hóa; sử dụng các khung thời gian ngắn và linh hoạt để dễ dàng phản hồi lại với các thay đổi trong quá trình phát triển. Ngày nay, triết lí Agile đã vượt xa khỏi khu vực truyền thống của mình là phát triển phần mềm mà còn đóng góp sự thay đổi trong cách thức làm việc, quản lí, sản xuất ở các ngành khác như sản xuất, dịch vụ, bán hàng, marketing, giáo dục....

Thuật ngữ "Agile" chính thức được sử dụng rộng rãi theo cách thống nhất kể từ khi "Tuyên ngôn Agile" được giới thiệu ra công chúng năm 2001. Nhờ tính linh hoạt, đa dạng và hiệu quả cao, các phương pháp Agile ngày càng trở thành sự lựa chọn hàng đầu của các khách hàng, nhà phát triển, các công ty phát triển phần mềm. Theo khảo sát của hãng nghiên cứu thị trường Forrester, mức độ phổ biến của Agile hiện đang ở mức cao nhất, và gấp nhiều lần so với các phương pháp truyền thống như thác nước hay CMMI (xem Hình 1).



*Hình 1: Mức độ phổ biến của các phương pháp (Nguồn: Forrester Research)*

## Tuyên ngôn Agile (Agile Manifesto)

Vào tháng 2 năm 2001, mười bảy nhà phát triển phần mềm đã gặp gỡ nhau ở Snowbird, Utah Resort để thảo luận về các phương pháp phát triển phần mềm gọn nhẹ và linh hoạt. Họ đã cùng nhau công bố "Tuyên ngôn Phát triển phần mềm linh hoạt" ("Manifesto for Agile Software Development" - gọi tắt là "Tuyên ngôn Agile") nhằm định nghĩa về Phát triển phần mềm linh hoạt. Đây là cột mốc quan trọng của agile cho dù trước đó, các phương pháp agile như XP, Scrum, v.v. đã được sử dụng thành công ở rất nhiều nơi, nhưng phải tới khi có sự xuất hiện của "Tuyên ngôn Agile", cùng với sự ra đời của các hiệp hội chuyên ngành agile như Agile Alliance hay Scrum Alliance, các phương pháp agile mới có một sự phát triển vượt bậc. Văn bản này rất ngắn, dễ hiểu, và rất quan trọng vì nó đưa ra các giá trị cốt lõi nhất mà toàn bộ những người dùng Agile tuân thủ; mặc dù các phương pháp họ đề xuất hoặc sử dụng trong thực tiễn có thể rất khác nhau. Toàn văn Tuyên ngôn Agile như sau:

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

* ***Cá nhân và sự tương tác****hơn là quy trình và công cụ.*
* ***Phần mềm chạy tốt****hơn là tài liệu đầy đủ.*
* ***Cộng tác với khách hàng****hơn là đàm phán hợp đồng.*
* ***Phản hồi với các thay đổi****hơn là bám sát kế hoạch.*

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

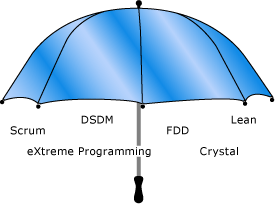
Bên cạnh đó, các nhà phát triển còn nhấn mạnh mười hai nguyên lý sau Tuyên ngôn Agile nhằm giúp các nhà phát triển vận dụng các phương pháp agile trong thực tiễn. Các nguyên lý được liệt kê sau đây:

* Ưu tiên cao nhất của chúng tôi là thỏa mãn khách hàng thông qua việc chuyển giao sớm và liên tục các phần mềm có giá trị.
* Chào đón việc thay đổi yêu cầu, thậm chí rất muộn trong quá trình phát triển. Quy trình Agile thực hiện những thay đổi phục vụ cho lợi thế cạnh tranh của khách hàng.
* Thường xuyên chuyển giao phần mềm tới khách hàng, từ vài tuần đến vài tháng, ưu tiên cho các khoảng thời gian ngắn hơn.
* Nhà kinh doanh và nhà phát triển phải làm việc cùng nhau hàng ngày trong suốt dự án.
* Xây dựng các dự án xung quanh những cá nhân tận tụy. Cung cấp cho họ môi trường và sự hỗ trợ cần thiết, và tin tưởng họ để hoàn thành công việc.
* Phương pháp hiệu quả nhất để truyền đạt thông tin tới và trong nội bộ nhóm phát triển là đối thoại trực tiếp.
* Phần mềm chạy tốt là thước đo chính của tiến độ.
* Các quy trình Agile thúc đẩy sự phát triển bền vững. Các nhà tài trợ, nhà phát triển, và người dùng có thể duy trì một nhịp độ liên tục không giới hạn.
* Liên tục quan tâm đến các kĩ thuật nổi trội và thiết kế tốt để gia tăng sự linh hoạt.
* Sự đơn giản – nghệ thuật tối đa hóa lượng công việc chưa xong – là căn bản.
* Các kiến trúc tốt nhất, yêu cầu tốt nhất, và thiết kế tốt nhất sẽ được làm ra bởi các nhóm tự tổ chức.
* Đội ngũ sẽ thường xuyên suy nghĩ về việc làm sao để trở nên hiệu quả hơn, sau đó họ sẽ điều chỉnh và thay đổi các hành vi của mình cho phù hợp.

Các nguyên lý này, cùng với năm điểm cốt lõi trong "Tuyên ngôn Agile" sẽ dẫn đường cho các nhà thực hành agile (agile practictioner) vận dụng tốt các phương pháp agile vào thực tiễn.

## Phân loại Agile

Phát triển linh hoạt bản thân nó không phải là một phương pháp. Nó là một thuật ngữ chung mô tả rất nhiều các phương pháp linh hoạt. Tại thời điểm ký kết Tuyên ngôn Agile năm 2001, những phương pháp này bao gồm: Scrum, XP, Crystal, FDD, và DSDM. Kể từ thời điểm đó, Lean cũng đã nổi lên như là một phương pháp linh hoạt có giá trị do vậy cũng được đưa vào trong chiếc ô của các phương pháp Agile trong hình minh họa dưới đây.



*Hình 1. Chiếc ô Agile*

Mỗi phương pháp phát triển linh hoạt có một cách tiếp cận hơi khác nhau để thực hiện các giá trị cốt lõi trong tuyên ngôn Agile, cũng giống như các ngôn ngữ máy tính có các biểu hiện tính năng cốt lõi trong lập trình hướng đối tượng theo những cách khác nhau. Một cuộc khảo sát gần đây cho thấy rằng, khoảng 50% học viên theo học phương pháp phát triển linh hoạt nói rằng đội của họ đang làm Scrum, 20% khác nói rằng họ đang làm Scrum với các thành phần của XP. Khoảng 12% nói rằng họ chỉ áp dụng XP. Do có hơn 80% áp dụng phát triển linh hoạt trên toàn thế giới là sử dụng Scrum và XP, nên bản MSF cho phát triển phần mềm linh hoạt phiên bản 5.0 tập trung vào quy trình cốt lõi và thực tiễn áp dụng của Scrum và XP.

## Giá trị cốt lõi của Agile

### Cá nhân và Sự tương tác

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

Tuy nhiên, chỉ tăng tần suất phản hồi và giao tiếp là không đủ để loại bỏ các vấn đề về truyền thông. Chu kỳ thanh-tra-và-thích-nghi làm việc tốt chỉ khi các thành viên trong nhóm thể hiện những hành vi quan trọng sau:

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

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

Đạt tới các hành vi đó khó khăn hơn rất nhiều so với bề ngoài của chúng. Hầu hết các nhóm tránh lé sự thật, sự minh bạch, và tin tưởng vào các chuẩn mực văn hóa hoặc có những kinh nghiệm tiêu cực từ các xung đột. Để chống lại những khuynh hướng này, ban lãnh đạo và thành viên nhóm phải tạo điều kiện cho những xung đột tích cực nhằm mục đích:

* Cải tiến quy trình phụ thuộc vào nhóm trong việc tạo ra danh sách các trở ngại hoặc vấn đề trong tổ chức, đối mặt với chúng một cách trung thực, và sau đó loại bỏ chúng một cách có hệ thống tùy theo độ ưu tiên.
* Đổi mới chỉ xảy ra với việc tự do trao đổi các ý tưởng mâu thuẫn lẫn nhau, một hiện tượng được nghiên cứu và viết thành tài liệu bởi Takeuchi và Nonaka, những người cha đỡ đầu của Scrum.
* Việc hướng các nhóm tới mục tiêu chung đòi hỏi nhóm phải đối mặt và giải quyết các vấn đề về xung đột.
* Cam kết làm việc cùng nhau sẽ xảy ra chỉ khi mọi người đồng ý với mục tiêu chung và sau đó đấu tranh cho sự tiến bộ của bản thân cũng như nhóm.

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

Để tạo ra các nhóm có hiệu suất cao, các phương pháp linh hoạt coi trọng cá nhân và sự tương tác hơn là quy trình và công cụ. Thực tế cho thấy rằng, tất cả các phương pháp linh hoạt tìm kiếm sự gia tăng trong truyền thông và cộng tác thông qua việc kiểm tra thường xuyên các chu trình thanh-tra-và-thích-nghi. Tuy nhiên, các chu trình đó chỉ thực sự làm việc tốt khi mà các nhà lãnh đạo agile khuyến khích các cuộc xung đột tích cực, thứ cần thiết để xây dựng một nền tảng vững chắc cho sự trung thực, tính minh bạch, lòng tin, sự tôn trọng, và cam kết từ các nhóm agile của họ.

### Phần mềm Chạy tốt hơn là Tài liệu Đầy đủ

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

Tất cả các nhóm agile phải xác lập những gì họ muốn nói là "phần mềm chạy tốt", cái thường được biết như là định nghĩa hoàn thành. Ở mức độ cao, một phần của chức năng hoàn thành chỉ khi các tính năng của chúng vượt qua tất cả các kiểm thử và có thể được vận hành bởi người dùng cuối. Ở mức thấp nhất, các nhóm phải vượt qua được kiểm thử đơn vị (unit test) và kiểm thử hệ thống. Các nhóm tốt nhất còn bao gồm việc kiểm thử tích hợp, kiểm thử hiệu năng, và kiểm thử chấp nhận của khách hàng trong định nghĩa hoàn thành đối với một phần chức năng. Thông qua nguồn dữ liệu phong phú từ các dự án, một công ty CMMI Cấp độ 5 cho thấy việc xác định kiểm thử chấp nhận cùng với các tính năng, triển khai một loạt các tính năng và theo độ ưu tiên, ngay lập tức chạy các kiểm thử chấp nhận với mỗi tính năng, và sửa bất cứ một lỗi nào có độ ưu tiên cao nhất sẽ tăng gấp đôi tốc độ sản xuất và giảm các sai sót đến 40%. Điều này có được từ một công ty có tỷ lệ sai sót thấp nhất thế giới.

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

### Cộng tác với Khách hàng hơn là Thương thảo Hợp đồng

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

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

Một nhóm XP đầu tiên đã bắt đầu với một dự án CNTT nội bộ. Họ có thể có sẵn người sử dụng đầu cuối của công ty trong nhóm làm việc với họ hàng ngày. Khoảng 10% thời gian, các tư vấn viên và nhóm nội bộ có thể tìm được một người dùng cuối có thể làm việc với nhóm từng ngày. 90% thời gian còn lại, họ phải cử ra người đại diện cho khách hàng. Người này, được nhóm XP gọi là Customer (khách hàng), làm việc trực tiếp với người dùng cuối để cung cấp một danh sách các tính năng rõ ràng cùng độ ưu tiên cho phép đội phát triển có thể thực hiện.

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

### Phản hồi với Thay đổi hơn là Bám sát Kế hoạch

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

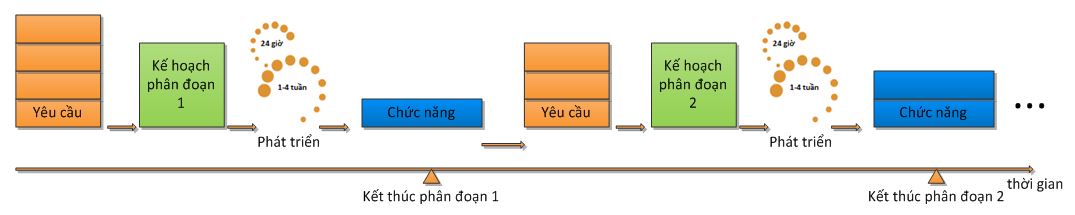
Tất cả các phương pháp phát triển linh hoạt đều được tích hợp sẵn những tiến trình thay đổi các kế hoạch trong một khoảng thời gian đều đặn dựa trên những thông tin phản hồi từ phía khách hàng cũng như bên đại diện của khách hàng. Các kế hoạch được thiết kế để sao cho luôn cung cấp giá trị kinh doanh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

## Đặc trưng Agile

Có rất nhiều phương pháp agile với các hướng tiếp cận rất khác nhau. Bên cạnh các cách thức tổ chức công việc, thiết lập quy trình, các phương pháp agile còn nghiên cứu và đưa vào sử dụng các công cụ và kĩ thuật đặc thù như công cụ tích hợp liên tục (continuous integration), kiểm thử đơn vị, mẫu thiết kế, tái cấu trúc, phát triển hướng kiểm thử (Test-Driven Development), phát triển hướng hành vi (Behaviour-Driven Development), hay lập trình theo cặp (Pair Programming)... để đảm bảo và gia tăng tính linh hoạt. Tuy vậy các phương pháp này chia sẻ nhiều đặc trưng giống nhau: cộng tác nhóm chặt chẽ, tổ chức các nhóm tự quản, liên chức năng, tính đáp ứng cao trong suốt vòng đời của dự án.

***Tính lặp (Iterative)***

Dự án sẽ được thực hiện trong các phân đoạn lặp đi lặp lại. Các phân đoạn (được gọi là Iteration hoặc Sprint) này thường có khung thời gian ngắn (từ một đến bốn tuần). Trong mỗi phân đoạn này, nhóm phát triển thực hiện đầy đủ các công việc cần thiết như lập kế hoạch, phân tích yêu cầu, thiết kế, triển khai, kiểm thử (với các mức độ khác nhau) để cho ra các phần nhỏ của sản phẩm. Các phương pháp agile thường phân rã mục tiêu thành các phần nhỏ với quá trình lập kế hoạch đơn giản và gọn nhẹ nhất có thể, và không thực hiện việc lập kế hoạch dài hạn. Có phương pháp như Scrum thậm chí sử dụng phương pháp lập kế hoạch just-in-time trong quá trình phát triển. Khi đó, thậm chí công việc lập kế hoạch, làm mịn kế hoạch được thực hiện liên tục trong suốt quá trình làm việc.



*Hình 2: Các phân đoạn lặp đi lặp lại trong agile*

***Tính tiệm tiến (Incremental) và tiến hóa (Evolutionary)***

Cuối các phân đoạn, nhóm phát triển thường cho ra các phần nhỏ của sản phẩm cuối cùng. Các phần nhỏ này thường là đầy đủ, có khả năng chạy tốt, được kiểm thử cẩn thận và có thể sử dụng ngay (potentially shippable product increment of functionality). Theo thời gian, phân đoạn này tiếp nối phân đoạn kia, các phần chạy được này sẽ được tích lũy, lớn dần lên cho tới khi toàn bộ yêu cầu của khách hàng được thỏa mãn. Khác với mô hình phát triển Thác nước – vốn chỉ cho phép nhìn thấy toàn bộ các chức năng tại thời điểm kết thúc dự án, sản phẩm trong các dự án agile lớn dần lên theo thời gian, tiến hóa cho tới khi đạt được trạng thái đủ để phát hành.

***Tính thích ứng (hay thích nghi – adaptive)***

Do các phân đoạn chỉ kéo dài trong một khoảng thời gian ngắn, và việc lập kế hoạch cũng được điều chỉnh liên tục, nên các thay đổi trong quá trình phát triển (yêu cầu thay đổi, thay đổi công nghệ, thay đổi định hướng về mục tiêu v.v.) đều có thể được đáp ứng theo cách thích hợp. Ví dụ, trong Scrum – phương pháp phổ biến nhất hiện nay – trong khi nhóm phát triển sản xuất ra các gói phần mềm, khách hàng có thể đưa thêm các yêu cầu mới, chủ sản phẩm (Product Owner) có thể đánh giá các yêu cầu này và có thể đưa vào làm việc trong phân đoạn (được gọi là Sprint trong Scrum) tiếp theo. Theo đó, các quy trình agile thường thích ứng rất tốt với các thay đổi.

**Nhóm tự tổ chức và liên chức năng**

Cấu trúc nhóm agile thường là liên chức năng(cross-functionality) và tự tổ chức(self-organizing). Theo đó, các nhóm này tự thực hiện lấy việc phân công công việc mà không dựa trên các mô tả cứng về chức danh (title) hay làm việc dựa trên một sự phân cấp rõ ràng trong tổ chức. Các nhóm này cộng tác với nhau để ra quyết định, theo dõi tiến độ, giải quyết các vấn đề mà không chờ mệnh lệnh của các cấp quản lý. Họ không làm việc theo cơ chế "mệnh lệnh và kiểm soát" (command and control). Nhóm tự tổ chức có nghĩa là nó đã đủ các kĩ năng (competency) cần thiết cho việc phát triển phần mềm, do vậy nó có thể được trao quyền để tự ra quyết định, tự quản lí và tổ chức lấy công việc của chính mình để đạt được hiệu quả cao nhất.

***Quản lý tiến trình thực nghiệm (Empirical Process Control)***

Các nhóm agile ra các quyết định dựa trên các dữ liệu thực tiễn thay vì tính toán lý thuyết hay các tiền giả định (prescription). Việc phân nhỏ dự án thành các phân đoạn ngắn góp phần gia tăng các điểm mốc để nhóm phát triển thu thập dữ kiện cho phép điều chỉnh các chiến lược phát triển của mình. Nói cách khác, Agile rút ngắn vòng đời phản hồi (short feedback life cycle) để dễ dàng thích nghi và gia tăng tính linh hoạt. Theo thời gian, các chiến lược này sẽ tiến gần đến trạng thái tối ưu, nhờ đó nhóm có thể kiểm soát được tiến trình, và nâng cao năng suất lao động.

***Đối thoại trực diện(face-to-face communication)***

Một số mô hình phát triển phần mềm dựa rất nhiều vào công việc giấy tờ, từ việc thu thập yêu cầu người dùng, viết đặc tả hệ thống, các thiết kế hệ thống v.v. Agile không phản đối công dụng của công việc tài liệu hóa, nhưng đánh giá cao hơn việc giao tiếp trực diện thay vì gián tiếp thông qua giấy tờ. Về yêu cầu của khách hàng, agile khuyến khích nhóm phát triển trực tiếp nói chuyện với khách hàng để hiểu rõ hơn về cái khách hàng thực sự cần, thay vì phụ thuộc nhiều vào các loại văn bản. Trong giao tiếp giữa nội bộ nhóm phát triển với nhau, thay vì một lập trình viên (thực hiện việc mã hóa) và một kĩ sư (thực hiện việc thiết kế) giao tiếp với nhau thông qua bản thiết kế, agile khuyến khích hai người này trực tiếp trao đổi và thống nhất với nhau về thiết kế của hệ thống và cùng nhau triển khai thành các chức năng theo yêu cầu.

Bản thân các nhóm agile thường nhỏ (ít hơn mười hai người, một nhóm lớn thường được phân nhỏ với cơ chế hợp tác đặc thù để luôn luôn có thông lượng giao tiếp tối đa) để đơn giản hóa quá trình giao tiếp, thúc đẩy việc cộng tác hiệu quả. Các dự án lớn muốn dùng agile thường phải phân thành nhóm nhỏ đồng thời làm việc với nhau hướng đến một mục tiêu chung. Việc này có thể đòi hỏi nhiều nỗ lực trong việc điều phối các mức ưu tiên giữa các nhóm.

Các nhóm phát triển thường tạo ra các thói quen và cơ chế trao đổi trực diện thường xuyên. Một trong các cơ chế thường thấy là các cuộc họp tập trung hằng ngày (daily meeting, daily Scrum, standup meeting). Tại đây, tất cả các thành viên được yêu cầu nói rõ cho nhóm của mình biết mình đã làm gì, đang làm gì, sắp làm gì và đang gặp phải khó khăn nào trong quá trình làm việc. Khi cơ chế này được thực hiện hiệu quả, nhóm luôn luôn nắm được tình hình công việc của mình, có các hành động thích hợp để vượt qua các trở lực để thực hiện thành công mục tiêu của dự án.

***Phát triển dựa trên giá trị (value-based development)***

Một trong các nguyên tắc cơ bản của agile là "phần mềm chạy tốt chính là thước đo của tiến độ". Nguyên tắc này giúp nhóm dám loại bỏ đi các công việc dư thừa không trực tiếp mang lại giá trị cho sản phẩm. Ví dụ, một nhóm thấy rằng có thể không cần phải viết ra tất cả các thiết kế của hệ thống, mà họ chỉ cần phác thảo các thiết kế này lên bảng, rồi cùng nhau triển khai các chức năng của hệ thống. Nhưng nếu như chủ sản phẩm quyết định rằng, khi chuyển giao sản phẩm, nhóm phát triển phải kèm theo thiết kế chi tiết của hệ thống vì chúng chiếm 20% giá trị của dự án theo yêu cầu của khách hàng, và vì khách hàng cần nó để chứng minh tính đúng đắn của các chức năng, và để phát triển tiếp các phân hệ tiếp theo của sản phẩm; thì nhóm phát triển sẽ phải thực hiện việc tài liệu hóa đầy đủ.

Để vận hành được cơ chế "làm việc dựa trên giá trị", nhóm agile thường làm việc trực tiếp và thường xuyên với khách hàng (hay đại diện của khách hàng), cộng tác trực tiếp với họ để biết yêu cầu nào có độ ưu tiên cao hơn, mang lại giá trị hơn sớm nhất có thể cho dự án. Nhờ đó các dự án agile thường giúp khách hàng tối ưu hóa được giá trị của dự án. Vì thế, agile gia tăng đáng kể độ hài lòng của khách hàng.

# Các phương pháp Agile

## Scrum

### Khái lược về Scrum

Scrum là một khung làm việc trong đó con người có thể xác định các vấn đề thích nghi phức tạp, trong khi vẫn giữ được năng suất và sáng tạo để chuyển giao các sản phẩm có giá trị cao nhất. Scrum có các tính chất:

* nhẹ nhàng
* dễ hiểu
* rất khó để tinh thông

Scrum là khung làm việc đã được sử dụng để quản lý quá trình phát triển các sản phẩm phức tạp từ đầu những năm 1990. Scrum không phải là một quy trình hay một kĩ thuật cụ thể để xây dựng sản phẩm; hơn thế, nó là một khung làm việc cho phép bạn sử dụng nhiều quy trình và kĩ thuật khác nhau. Scrum làm rõ độ hiệu quả tương đối của công tác quản lý và phát triển sản phẩm, từ đó cho phép bạn cải tiến nó.

#### Khung làm việc Scrum

Khung làm việc Scrum bao gồm một Nhóm Scrum với các vai trò được phân định rõ ràng, các sự kiện, các đồ nghề và các quy tắc. Mỗi thành phần trong khung làm việc phục vụ một mục đích rõ ràng và nòng cốt trong việc sử dụng và thành công của Scrum.

Các chiến lược cụ thể để sử dụng Scrum có thể rất khác nhau và được mô tả ở đâu đó.

Các quy tắc của Scrum gắn kết các yếu tố sự kiện, vai trò, đồ nghề với nhau, điều khiển các mối quan hệ và tương tác giữa chúng.

### Lý thuyết Scrum

Scrum được xây dựng dựa trên lý thuyết quản lý tiến trình thực nghiệm (empirical process control), hay “lý thuyết thực nghiệm” (empiricism). Lý thuyết này chỉ ra rằng tri thức đến từ kinh nghiệm *và* việc ra quyết định được dựa trên những gì đã biết. Scrum sử dụng các tiếp cận lặp (iterative), tăng dần (incremental) để tối ưu hóa tính dự đoán (predictability) và kiểm soát rủi ro.

Ba yếu tố nòng cốt tạo thành một mô hình quản lý tiến trình thực nghiệm gồm: tính minh bạch (transparency), sự thanh tra (inspection) và sự thích nghi (adaptation).

#### Minh bạch

Các khía cạnh quan trọng của tiến trình phải được hiển thị rõ ràng cho những nhiệm với thành quả của tiến trình đó. Sự minh bạch yêu cầu các yếu tố này nghĩa theo một tiêu chuẩn để những người quan sát có thể hiểu những gì họ một cách. Ví dụ:

* Một ngôn ngữ chung về quy trình cần phải được chia sẻ cho tất cả các bên tham gia; và,
* Một định nghĩa chung về “Hoàn thành” phải được chia sẻ bởi những người đảm đương công việc và những người chấp nhận sản phẩm của công việc đó.

#### Thanh tra

Người sử dụng Scrum phải thường xuyên thanh tra các đồ nghề và tiến độ đến đích để phát hiện các bất thường không theo ý muốn. Tần suất thanh tra không nên quá dày để khỏi ảnh hưởng đến công việc. Công tác thanh tra có ích nhất khi được thực hiện bởi người có kĩ năng tại các điểm quan trọng của công việc.

#### Thích nghi

Nếu một người thanh tra xác định được rằng có vấn đề nào đó vượt quá giới hạn cho phép, và hậu quả của vấn đề đó đối với sản phẩm là không thể chấp nhận được, thì quy trình hoặc các vật liệu được xử lý (processed material) phải được điều chỉnh. Sự điều chỉnh phải được tiến hành càng sớm càng tốt để giảm thiểu các sai sót khác có thể xảy ra.

Scrum cung cấp bốn cơ hội chính thức cho việc thanh tra và thích nghi trong các Sự kiện Scrum, bao gồm:

* Buổi Họp Kế hoạch Sprint (Sprint Planning Meeting)
* Họp Scrum hằng ngày (Daily Scrum)
* Sơ kết Sprint (Sprint Review)
* Cải tiến Sprint (Sprint Retrospective)

### Scrum

Khung làm việc Scrum được cấu trúc để hỗ trợ việc phát triển các sản phẩm phức tạp. Scrum bao gồm Nhóm Scrum với các vai trò được phân định rõ ràng, các sự kiện, các đồ nghề và các quy tắc. Mỗi thành phần trong khung làm việc phục vụ một mục đích rõ ràng và là nòng cốt trong việc sử dụng và sự thành công của Scrum.

### Nhóm Scrum

Nhóm Scrum bao gồm Product Owner (Chủ Sản phẩm), Nhóm phát triển (Development Team), và Scrum Master. Các Nhóm Scrum là các nhóm tự quản (self-organizing) và liên chức năng (cross-functional). Các nhóm tự quản tự mình chọn cách thức tốt nhất để hoàn thành công việc của họ, chứ không bị chỉ đạo bởi ai đó bên ngoài nhóm. Các nhóm liên chức năng có đủ kĩ năng cần thiết để hoàn thành công việc mà không phụ thuộc vào bất kì người ngoài nào khác. Mô hình nhóm trong Scrum được thiết kế để tối ưu hóa sự linh hoạt, sáng tạo và năng suất.

Các Nhóm Scrum chuyển giao sản phẩm theo phân đoạn lặp đi lặp lại và tăng dần, tối đa hóa cơ hội cho các phản hồi. Việc chuyển giao tăng dần (incremental) các gói sản phẩm đạt tiêu chuẩn “Hoàn chỉnh” đảm bảo một phiên bản khả dụng của sản phẩm luôn luôn sẵn sàng để cung cấp tới chủ sản phẩm.

#### Product Owner

Product Owner (Chủ sản phẩm) chịu trách nhiệm tối đa hóa giá trị của sản phẩm và công việc của Nhóm Phát triển. Cách thức để đạt được điều đó có thể rất khác nhau giữa các tổ chức, Nhóm Scrum và các cá nhân.

Product Owner là một người chủ yếu chịu trách nhiệm về việc quản lý Product Backlog. Đây là công cụ quản lý chứa:

* Mô tả các hạng mục Product backlog;
* Trình tự của các hạng mục trong Product Backlog để đạt được mục đích và hoàn thành các nhiệm vụ;
* Sự đảm bảo giá trị của các công việc của Nhóm Phát triển;
* Sự đảm bảo cho Product Backlog là luôn luôn hiện hữu, thông suốt, và rõ ràng tới tất cả mọi người, và chỉ ra những gì mà Nhóm Scrum sẽ làm việc; và,
* Sự đảm bảo cho Nhóm Phát triển hiểu rõ các hạng mục trong Product Backlog với các mức độ cần thiết.

Product Owner có thể thực hiện công việc trên, hoặc để Nhóm Phát triển làm. Tuy nhiên, Product Owner vẫn phải chịu trách nhiệm chính. Product Owner là một người, không phải là một ủy ban. Product Owner có thể cần tới một ủy ban tham gia vào Product Backlog, nhưng những người trong ủy ban muốn thay đổi trình tự các hạng mục trong Product Backlog phải thuyết phục được Product Owner.

Để Product Owner thành công, cả tổ chức phải tôn trọng các quyết định của người này. Các quyết định đó được hiển thị trong nội dung và thứ tự trong Product Backlog. Không ai ngoài Product Owner được phép yêu cầu Nhóm Phát triển làm gì khác, và Nhóm Phát triển cũng không được phép làm gì theo lời bất cứ người nào khác.

#### Nhóm Phát triển

Nhóm Phát triển (Development Team) gồm các chuyên gia làm việc để cho ra các phần tăng trưởng có thể phát hành được (potentially releasable) cuối mỗi Sprint. Chỉ các thành viên của Nhóm Phát triển mới tạo ra các phần tăng trưởng này (Increment).

Nhóm Phát triển được cấu trúc và trao quyền để tổ chức và quản lý công việc của họ. Sự hợp lực sẽ tối ưu hóa nỗ lực và hiệu quả tổng thể của Nhóm Phát triển. Nhóm Phát triển có các đặc trưng sau:

* Đó là nhóm tự tổ chức. Không ai (kể cả Scrum Master) có quyền yêu cầu Nhóm Phát triển làm sao để chuyển Product Backlog thành các phần tăng trưởng có thể chuyển giao được;
* Đó là nhóm liên chức năng, với tất cả các kĩ năng cần thiết để tạo ra phần tăng trưởng của sản phẩm;
* Scrum không ghi nhận một chức danh nào trong Nhóm Phát triển ngoài Nhà phát triển (Developer), theo tính chất công việc của người này; không có ngoại lệ cho quy tắc này;
* Các thành viên Nhóm phát triển có thể có các kĩ năng chuyên biệt và các chuyên môn đặc thù, nhưng họ phải chịu trách nhiệm dưới một thể thống nhất là Nhóm Phát triển; và,
* Nhóm Phát triển không chứa các nhóm con nào khác với các chức năng đặc thù như ‘nhóm kiểm thử’ hay ‘phân tích nghiệp vụ’.

**Độ lớn của Nhóm Phát triển**

Độ lớn tối ưu của Nhóm Phát triển là đủ nhỏ để giữ được sự linh hoạt và đủ lớn để hoàn thành công việc. Ít hơn ba người có thể làm giảm sự tương tác và dẫn đến năng suất thấp. Các Nhóm Phát triển nhỏ hơn có thể phải đối mặt với các ràng buộc kĩ năng trong suốt Sprint, dẫn đến Nhóm Phát triển khó có thể chuyển giao gói tăng trưởng phát hành được. Một nhóm nhiều hơn chín người cần nhiều sự điều phối hơn. Các Nhóm Phát triển lớn phát sinh quá nhiều phức tạp để thực hiện việc kiểm soát tiến trình thực nghiệm. Các vai trò Product Owner và Scrum Master không được tính vào kích thước của Nhóm Phát triển, trừ khi họ cũng kiêm luôn vai trò là thành viên của Nhóm Phát triển.

#### Scrum Master

Scrum Master chịu trách nhiệm đảm bảo mọi người hiểu và dùng được Scrum. Scrum Master thực hiện việc này bằng cách đảm bảo Nhóm Scrum tuân thủ lý thuyết, thực tiễn và các quy tắc của Scrum. Scrum Master là một lãnh đạo, nhưng cũng là đầy tớ của Nhóm Scrum.

Scrum Master giúp đỡ những người ngoài Nhóm Scrum hiểu cách phải tương tác với Nhóm sao cho hiệu quả nhất. Scrum Master giúp đỡ tất cả mọi người thay đổi các mối tương tác này để tối đa hóa giá trị mà Nhóm Scrum tạo ra.

**Scrum Master phục vụ Product Owner điều gì?**

Scrum Master phục vụ Product Ower theo nhiều cách, bao gồm:

* Tìm kiếm các kĩ thuật để quản lý hiệu quả Product Backlog;
* Giao tiếp tích cực với Nhóm Phát triển về tầm nhìn, mục đích, và các hạng mục của Product Backlog;
* Dạy cho Nhóm Phát triển biết cách tạo ra các hạng mục Product Backlog thật rõ ràng và đơn giản;
* Hiểu rõ việc lập kế hoạch dài hạn sản phẩm trong một môi trường thực nghiệm;
* Hiểu rõ và thực hành sự linh hoạt (agility); và,
* Thúc đẩy các sự kiện Scrum theo yêu cầu hoặc khi cần thiết.

**Scrum Master phục vụ gì cho Nhóm Phát triển?**

Scrum Master phục vụ Nhóm Phát triển theo nhiều cách, bao gồm:

* Huấn luyện Nhóm Phát triển cách tự tổ chức và làm việc liên chức năng;
* Dạy và lãnh đạo Nhóm Phát triển cách tạo ra các sản phẩm có giá trị cao;
* Loại bỏ các trở lực trong quá trình tác nghiệp của Nhóm Phát triển;
* Thúc đẩy các sự kiện Scrum theo yêu cầu hoặc khi cần thiết; và,
* Huấn luyện Nhóm Phát triển trong trường hợp tổ chức chưa có hiểu biết về Scrum.

**Scrum Master phục vụ gì cho Tổ chức?**

Scrum Master phục vụ Tổ chức theo nhiều cách, bao gồm:

* Lãnh đạo và huấn luyện tổ chức trong việc áp dụng Scrum;
* Lập kế hoạch triển khai Scrum trong phạm vi tổ chức;
* Giúp đỡ nhân viên và các bên hữu quan hiểu và sử dụng được Scrum cũng như quá trình phát triển sản phẩm thực nghiệm (emprical product development);
* Tạo ra sự thay đổi làm tăng năng suất của Nhóm Scrum; và,
* Làm việc với các Scrum Master khác để gia tăng hiệu quả của việc áp dụng Scrum trong tổ chức của mình.

### Các sự kiện Scrum

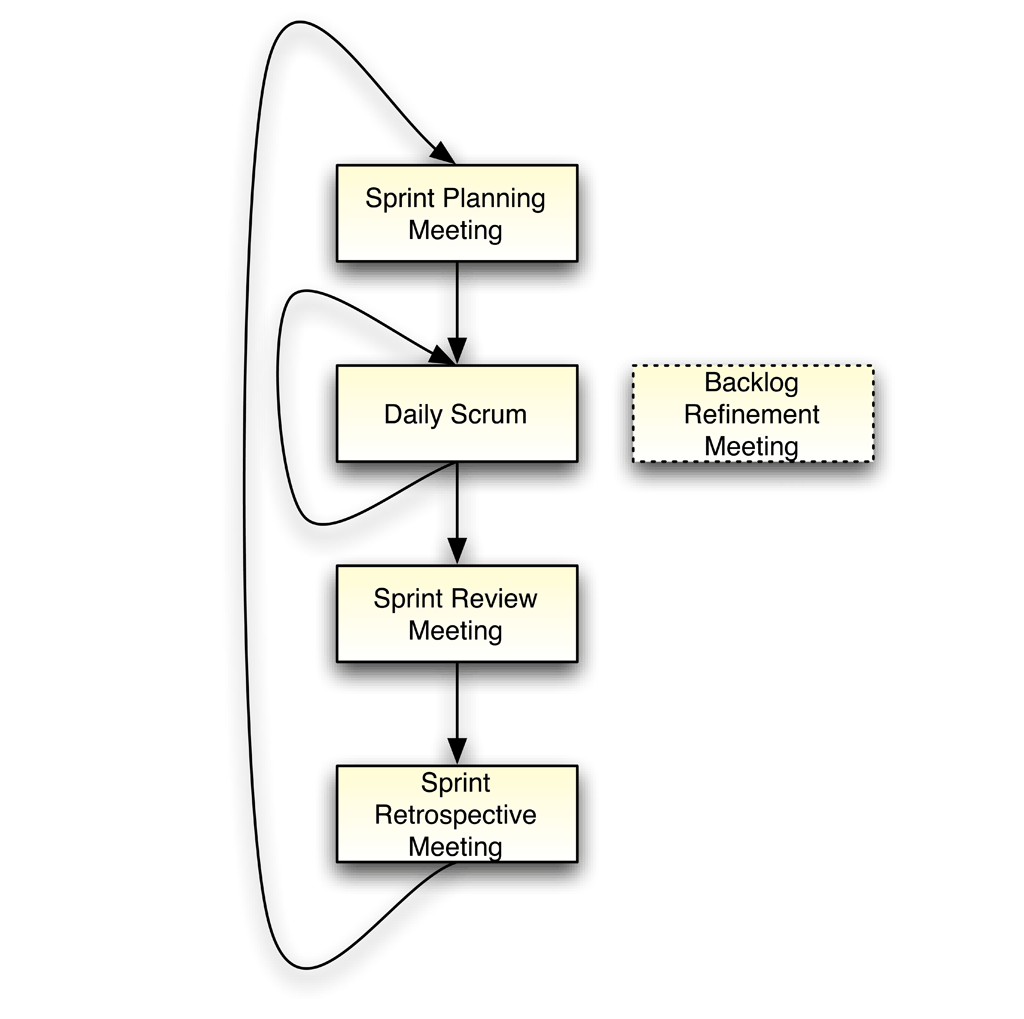
Các sự kiện được mô tả trong Scrum nhằm tạo ra thói quen và để giảm thiểu những buổi họp hành vốn không được định nghĩa trong Scrum. Scrum dùng các sự kiện được đóng khung thời gian (time-boxed), nghĩa là mỗi sự kiện có giới hạn thời gian tối đa. Điều này đảm bảo thời lượng vừa đủ để tránh lãng phí thời gian không cần thiết cho sự kiện.

Bao gồm cả bản thân Sprint vốn chứa tất cả các sự kiện khác, mỗi sự kiện trong Scrum là một cơ hội chính thức để thực hiện cơ chế thanh tra và thích nghi. Các sự kiện này được thiết kế đặc biệt để đảm bảo sự minh bạch và thanh tra. Nếu không thực hiện được các điều này có thể dẫn đến giảm thiểu tính minh bạch và đánh mất cơ hội để thanh tra và thích nghi.

#### Sprint

Trái tim của Scrum chính là Sprint, một khung-thời-gian (time-box) có thời gian một tháng hoặc ngắn hơn để tạo ra các phần tăng trưởng của sản phẩm có thể phát hành được. Sprint có khoảng thời gian nhất quán trong suốt quá trình phát triển. Một Sprint mới bắt đầu ngay khi Sprint trước khép lại.

Sprint chứa và bao gồm một cuộc Họp Kế hoạch Sprint (Sprint Planning Meeting), các cuộc Họp Scrum hằng ngày (Daily Scrum), một buổi Sơ kết Sprint (Sprint Review), và một buổi họp Cải tiến Sprint (Sprint Retrospective).



*Hỉnh 3. Các sự kiện Sprint*

Trong suốt Sprint:

* Không cho phép bất kì sự thay đổi nào ảnh hưởng đến Mục tiêu Sprint (Sprint Goal);
* Thành phần Nhóm Phát triển được giữ nguyên;
* Mục tiêu chất lượng không được cắt giảm; và,
* Phạm vi có thể được làm rõ và tái thương lượng giữa Product Owner và Nhóm Phát triển.

Mỗi Sprint có thể được coi như một tiểu dự án với độ dài một tháng. Giống như dự án, Sprint được dùng để hoàn tất điều gì đó. Mỗi Sprint có một định nghĩa về việc phải xây dựng cái gì, một bản thiết kế và bản kế hoạch linh hoạt sẽ hướng dẫn quá trình xây dựng đó, các công việc cần làm, và sản phẩm của quá trình đó.

Sprint được giới hạn trong vòng một tháng lịch (calendar month). Khi Sprint bị kéo dài quá thì định nghĩa về việc phải xây dựng cái gì có thể bị thay đổi, sự phức tạp sẽ gia tăng và rủi ro sẽ tăng theo. Sprint đảm bảo tính dự đoán bằng sự thanh tra và thích nghi trong tiến trình tiến tới mục tiêu của mỗi tháng đó. Sprint cũng sẽ giới hạn rủi ro trong phạm vi chi phí của một tháng lịch.

**Hủy Sprint**

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

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

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

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

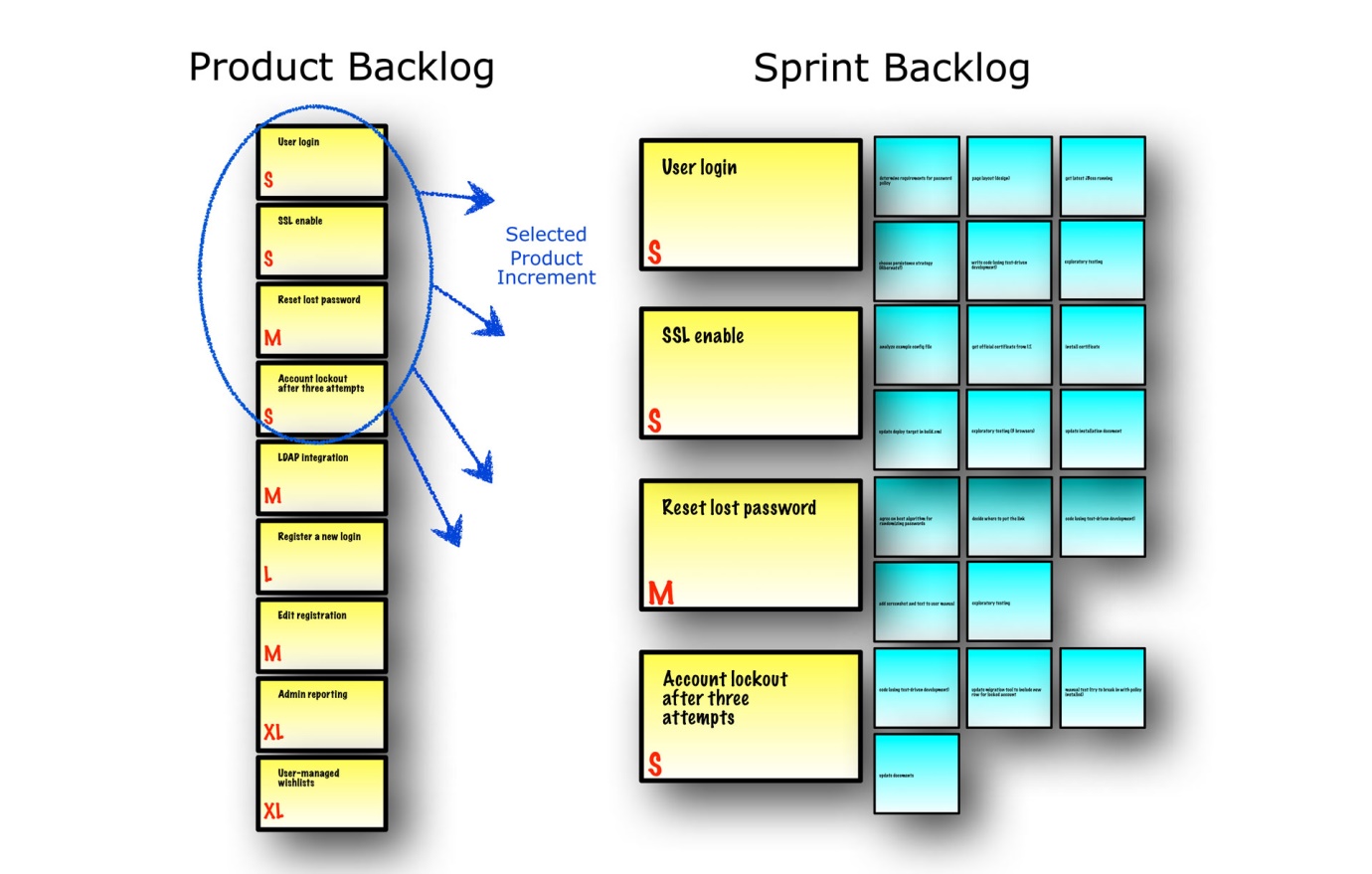
#### Họp Kế hoạch Sprint

Công việc trong Sprint được lên kế hoạch trong buổi Họp Kế hoạch Sprint (Sprint Planning Meeting). Kế hoạch cho Sprint được tạo ra nhờ nỗ lực cộng tác của toàn bộ Nhóm Scrum.

Buổi Họp Kế hoạch Sprint được đóng khung trong tám tiếng cho mỗi Sprint một tháng. Với các Sprint ngắn hơn thì thời gian cho buổi họp được rút ngắn lại. Ví dụ như một Sprint hai tuần có thể chỉ cần họp kế hoạch tới bốn tiếng là đủ.

Buổi Họp Kế hoạch Sprint có hai phần, mỗi phần chiếm một nửa khung thời gian. Hai phần của buổi Họp Kế hoạch Sprint lần lượt trả lời hai câu hỏi sau đây:

* Sprint này phải chuyển giao cái gì?
* Làm sao để đạt được điều đó?



*Hình 4. Đưa project backlog vào sprint backlog*

**Phần Một: Phải làm gì trong Sprint này?**

Trong phần này, Nhóm Phát triển làm việc để dự báo chức năng sẽ được phát triển trong Sprint. Product Owner trình bày các hạng mục được xếp thứ tự Product Backlog cho Nhóm Phát triển và toàn bộ Nhóm Scrum sẽ hợp tác để tìm hiểu công việc phải làm trong Sprint.

Đầu vào của buổi họp này là Product Backlog, phần tăng trưởng của sản phẩm gần đây nhất, năng lực hiện có của Nhóm Phát triển trong Sprint tới, và hiệu suất trong quá khứ của Nhóm Phát triển. Số lượng hạng mục được chọn từ Product Backlog cho Sprint này sẽ hoàn toàn phụ thuộc vào Nhóm Phát triển. Chỉ Nhóm Phát triển có thể đánh giá họ có thể hoàn thành những gì trong Sprint tới đây.

Sau khi Nhóm Phát triển dự báo các hạng mục Product Backlog sẽ được chuyển giao trong Sprint, Nhóm Scrum xác lập Mục tiêu Sprint. Mục tiêu Sprint là đích đến của Sprint thông qua việc triển khai các hạng mục Product Backlog, và nó cung cấp những chỉ dẫn cho Nhóm Phát triển về việc tại sao phải xây dựng phần tăng trưởng đó.

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

Sau khi đã chọn công việc cho Sprint, Nhóm Phát triển quyết định cách thức để xây dựng các chức năng sẽ có trong phần tăng trưởng “hoàn chỉnh” trong suốt Sprint. Các hạng mục Product Backlog được lựa chọn cho Sprint cộng với kế hoạch để chuyển giao chúng được gọi là Sprint Backlog.

Nhóm Phát triển thường bắt đầu công việc bằng cách thiết kế hệ thống và các công việc cần thiết để chuyển Product Backlog thành gói sản phẩm chạy được. Công việc có thể lớn nhỏ khác nhau. Tuy vậy, một lượng công việc vừa đủ được lên kế hoạch trong suốt buổi Họp Kế hoạch Sprint cho Nhóm Phát triển sẽ dự báo những thứ có thể làm trong Sprint sắp tới. Các công việc được lên kế hoạch trong những ngày đầu tiên của Sprint bởi Nhóm Phát triển sẽ được phân tách thành các đơn vị nhỏ hơn trong phạm vi một ngày hoặc nhỏ hơn nữa ở cuối buổi họp. Nhóm Phát triển sẽ tự tổ chức để làm việc trên Sprint Backlog, cả khi lập kế hoạch lẫn thực thi kế hoạch trong suốt Sprint.

Product Owner có thể có mặt trong phần hai của buổi Họp Kế hoạch Sprint để làm sáng tỏ các khúc mắc về các hạng mục được lựa chọn trong Product Backlog , cũng như giúp Nhóm Phát triển đưa ra quyết định trong một số tình huống đặc biệt. Nếu Nhóm Phát triển thấy có quá nhiều hoặc quá ít việc, họ có thể thương lượng thêm với Product Owner về việc này. Nhóm Phát triển cũng có thể mời một số người khác tham dự để hỗ trợ một số vấn đề kĩ thuật hoặc chuyên môn.

Kết thúc buổi Họp Kế hoạch Sprint, Nhóm Phát triển phải biết cách giải thích cho Product Owner và Scrum Master biết họ sẽ dự định làm việc như thế nào với tư cách một nhóm tự tổ chức để hoàn thành Mục tiêu Sprint và tạo ra phần tăng trưởng theo yêu cầu.

**Mục tiêu Sprint**

Mục tiêu Sprint cho phép Nhóm Phát triển có một số sự linh hoạt nhất định về việc phải triển khai các chức năng như thế nào trong suốt Sprint.

Khi Nhóm Phát triển làm việc, họ sẽ ghi nhớ Mục tiêu này trong đầu. Để thỏa mãn Mục tiêu Sprint, họ sẽ triển khai các chức năng cũng như các kĩ thuật cần thiết. Nếu công việc phức tạp hơn mong đợi thì họ có thể cộng tác với Product Owner để thương lượng về phạm vi của Sprint Backlog trong Sprint.

Mục tiêu Sprint có thể đóng vai trò là một cột mốc (milestone) trong một mục đích lớn hơn trong chặng đường phát triển sản phẩm.

#### Họp Scrum Hằng ngày

Cuộc họp Scrum Hằng (Daily Scrum) ngày được đóng khung trong 15 phút để Nhóm Phát triển đồng bộ hóa các hoạt động của thành viên và tạo lập kế hoạch cho 24 giờ tiếp theo. Điều này có được nhờ việc thanh tra các công việc kể từ cuộc họp Scrum Hằng ngày trước, và dự báo những công việc sẽ được hoàn thành trước buổi họp lần sau.

Cuộc họp Scrum Hằng ngày được tổ chức tại cùng một địa điểm để giảm thiểu sự phức tạp không cần thiết. Trong suốt cuộc họp, mỗi thành viên Nhóm Phát triển giải thích rõ:

* Việc gì đã được thực hiện kể từ lần họp trước?
* Việc gì sẽ được hoàn thành trước buổi họp lần sau?
* Có vấn đề gì nảy sinh trong quá trình làm việc?

Nhóm Phát triển sử dụng cuộc họp Scrum Hằng ngày để đánh giá tiến độ công việc hướng tới Mục tiêu Sprint và đánh giá xu hướng tiến triển của công việc trong Sprint Backlog. Cuộc họp Scrum Hằng ngày tối ưu hóa khả năng để Nhóm Phát triển có thể đạt được Mục tiêu Sprint. Nhóm Phát triển thường họp mặt ngay sau khi họp xong Scrum Hằng ngày để tái lập kế hoạch cho các công việc còn lại trong Sprint. Hằng ngày, Nhóm Phát triển có thể giải thích cho Product Owner và Scrum Master biết họ định làm gì với tư cách là một nhóm tự quản để hoàn thành mục tiêu và tạo ra các phần tăng trưởng cần thiết trong Sprint.

Scrum Master đảm bảo cho Nhóm Phát triển tham gia họp, nhưng chính Nhóm Phát triển mới có trách nhiệm chính trong cuộc họp Scrum Hằng ngày. Scrum Master phải dạy cho Nhóm Phát triển biết cách giữ cuộc họp ngắn gọn trong phạm vi khung thời gian 15 phút.

Scrum Master phải áp đặt quy tắc về việc chỉ có Nhóm Phát triển mới được tham gia cuộc họp Scrum Hằng ngày. Cuộc họp Scrum Hằng ngày không phải là cuộc họp báo cáo tình hình, mà là để mọi người biến đổi các hạng mục Product Backlog thành ra phần tăng trưởng cần thiết.

Họp Scrum Hằng ngày sẽ cải tiến quá trình giao tiếp, lược bỏ các buổi họp hành không cần thiết, nhận biết và loại bỏ các trở lực trong quá trình phát triển, nhấn mạnh và phát huy các quyết định nhanh chóng, và nâng cao mức độ hiểu biết của Nhóm Phát triển về dự án. Cuộc họp này là chìa khóa của cơ chế thanh tra và thích nghi trong Scrum.

#### Sơ kết Sprint

Buổi Sơ kết Sprint (Sprint Review) được tổ chức khi Sprint kết thúc để rà soát lại phần tăng trưởng vừa làm ra trong Sprint đó, và để thực hiện các biện pháp thích nghi nếu cần. Trong cuộc họp này, Nhóm Scrum và các bên hữu quan sẽ trao đổi với nhau về những gì vừa hoàn thành trong Sprint vừa rồi. Trên cơ sở đó và những sự thay đổi trong Product Backlog trong suốt Sprint, người tham dự cuộc họp sẽ hợp tác để thảo luận về những công việc sắp triển khai. Đây là cuộc họp không trang trọng, và việc trình bày về gói tăng trưởng chủ yếu nhằm mục đích cung cấp các phản hồi hữu ích và khuyến khích sự cộng tác giữa các bên.

Cuộc họp này được đóng khung trong bốn giờ cho các Sprint có độ dài một tháng. Sprint ngắn hơn thì thời gian họp rút bớt cho phù hợp. Ví dụ, một Sprint hai tuần chỉ cần đến hai giờ cho buổi Sơ kết Sprint.

Buổi Sơ kết Sprint có một số đặc điểm sau:

* Product Owner nhận biết phần nào là “Hoàn thành” và phần nào chưa “Hoàn thành”;
* Nhóm Phát triển thảo luận những điều thuận lợi trong Sprint vừa qua, những khó khăn mà nhóm đã trải qua, và cách thức giải quyết các vấn đề đó;
* Nhóm Phát triển trình diễn các phần việc đã “Hoàn thành” và trả lời các câu hỏi của cử tọa về gói tăng trưởng;
* Product Owner trao đổi về Product Backlog. Dựa trên tiến độ hiện thời, Product Owner đưa ra dự đoán ngày hoàn thành dự án; và ,
* Toàn bộ nhóm thảo luận về những gì sẽ làm, nhờ đó buổi Sơ kết Sprint cung cấp các giá trị đầu vào cho buổi Họp Kế hoạch Sprint tiếp theo.

Kết quả của cuộc họp Sơ kết Sprint là một bản Product Backlog đã được cập nhật, với các hạng mục dự định sẽ được triển khai trong Sprint tới. Product Backlog có thể được điều chỉnh toàn diện để thích ứng với các cơ hội mới.

#### Cải tiến Sprint

Buổi họp Cải tiến Sprint (Sprint Retrospective) là cơ hội để Nhóm Scrum tự thanh tra và đưa ra kế hoạch cho các cải tiến trong Sprint tiếp theo. Buổi họp Cải tiến Sprint được tổ chức ngay sau Sơ kết Sprint và trước khi cuộc Họp Kế hoạch Sprint tiếp theo diễn ra. Cuộc họp này được đóng khung trong phạm vi ba giờ cho các Sprint một tháng. Sprint ngắn hơn thì cuộc họp sẽ được rút ngắn lại cho phù hợp.

Mục đích của cuộc họp Cải tiến Sprint là để:

* Thanh tra lại tất cả các yếu tố trong Sprint vừa diễn ra, từ con người, các mối quan hệ, quy trình, và công cụ;
* Nhận biết và xếp đặt lại các hạng mục chủ chốt đã được thực hiện tốt, và các cải tiến dự định; và,
* Tạo ra một kế hoạch để triển khai các cải tiến cách thức làm việc của Nhóm Scrum.

Scrum Master động viên Nhóm Scrum để cải tiến, trong phạm vi khung làm việc Scrum, quy trình phát triển và các biện pháp thực hành để nâng cao hiệu quả và thú vị cho Sprint tiếp theo. Trong cuộc họp Cải tiến Sprint, Nhóm Scrum sẽ lập kế hoạch để gia tăng chất lượng sản phẩm bằng việc định nghĩa lại Định nghĩa “Hoàn thành” khi cần thiết.

Kết thúc cuộc họp Cải tiến Sprint, Nhóm Scrum phải nhận biết ra các cải tiến sẽ được triển khai trong Sprint tới. Việc triển khai các cải tiến này chính là sự thích nghi của bản thân Nhóm Scrum. Mặc dù các cải tiến có thể được triển khai tại bất kì thời điểm nào đó, cuộc họp Cải tiến Sprint cung cấp một phiên làm việc chính thức để tập trung vào việc thanh tra và thích nghi.

### Đồ nghề Scrum

Các đồ nghề Scrum (artifact) biểu đạt các công việc hoặc các giá trị bằng nhiều cách hữu ích để cung cấp tính minh bạch và các cơ hội cho việc thanh tra và thích nghi. Các đồ nghề Scrum được thiết kế để tối đa hóa tính minh bạch và thông suốt của các thông tin chính cần thiết để đảm bảo Nhóm Scrum thành công trong việc chuyển giao các gói tăng trưởng đạt tiêu chuẩn “Hoàn thành”.

#### Product Backlog

Product Backlog là một danh sách sắp thứ tự tất cả những gì cần thiết của sản phẩm. Nó là nguồn yêu cầu duy nhất thể hiện các thay đổi trong sản phẩm. Product Owner là người chịu trách nhiệm về Product Backlog, nội dung của nó, sự hiện diện, và thứ tự các hạng mục trong đó.

Product Backlog có thể không bao giờ hoàn chỉnh. Phiên bản sớm nhất của Product Backlog chỉ cho thấy các yêu cầu được tìm hiểu rõ ràng từ lúc đầu tiên. Product Backlog sẽ tiến hóa cùng với với sản phẩm và môi trường mà nó sẽ được sử dụng. Product Backlog là động, nó thay đổi thường xuyên để nhận biết những gì mà sản phẩm cần phải có để có tính cạnh tranh và hữu ích. Chừng nào sản phẩm còn đó, thì Product Backlog cũng hiện diện.

Product Backlog liệt kê tất cả các tính năng (feature), chức năng, yêu cầu, cải thiện, vá lỗi cần thiết để làm nên sản phẩm trong tương lai. Các hạng mục trong Product Backlog được mô tả với các thuộc tính như: mô tả, thứ tự, và ước lượng.

Product Backlog thường được sắp xếp theo các giá trị, độ rủi ro, độ ưu tiên, và sự cần thiết. Các hạng mục đứng đầu danh sách sẽ trực tiếp điều khiển các hoạt động phát triển. Càng ở thứ tự cao hơn, các hạng mục càng được quan tâm nhiều hơn, và được tập trung nỗ lực nhiều hơn vì chính giá trị của chúng.

Các hạng mục có thứ tự cao hơn rõ ràng và chi tiết hơn những mục ở vị trí thấp hơn trong Product Backlog. Ước lượng sẽ chính xác hơn nếu như hạng mục rõ ràng và chi tiết hơn; vị trí càng thấp, hạng mục càng ít chi tiết. Hạng mục Product Backlog sắp tham gia vào Sprint tới thuộc loại “mịn” (fine-grained), được phân tách sao cho các hạng mục đó có thể được hoàn thành trong khung thời gian của Sprint. Hạng mục Product Backlog có thể hoàn tất trong một Sprint được coi là “sẵn sàng” hay “có thể hành động”, có thể được chọn ra trong buổi Họp Kế hoạch Sprint.

Khi sản phẩm được đưa vào sử dụng và mang lại giá trị, thị trường sẽ cung cấp các phản hồi, Product Backlog sẽ trở thành một danh sách lớn hơn và toàn diện hơn. Nhu cầu thì không ngừng thay đổi, vì thế một Product Backlog là một thực thể sống động. Sự thay đổi trong các yêu cầu nghiệp vụ, điều kiện thị trường, hay công nghệ có thể dẫn đến các thay đổi trong Product Backlog.

Trong trường hợp nhiều Nhóm Scrum làm việc với nhau trên cùng một sản phẩm. Một Product Backlog được dùng để mô tả những công việc tới đây của Sản phẩm. Khi đó, các hạng mục có thể được nhóm lại theo một tính chất nào đó.

Việc “làm mịn” (grooming) Product Backlog là hoạt động thêm vào các chi tiết, ước lượng, và trình tự của các hạng mục trong Product Backlog. Đây là quá trình liên tục, theo đó Product Owner và Nhóm Phát triển thảo luận về các chi tiết của từng hạng mục. Trong suốt quá trình làm mịn này, các hạng mục liên tục được xem xét và rà soát cẩn thận. Tuy nhiên, chúng có thể được cập nhật tại bất kì thời điểm nào bởi Product Owner.

Việc làm mịn là một hoạt động bán thời gian giữa Product Owner và Nhóm Phát triển. Thông thường, Nhóm Phát triển sẽ có hiểu biết để tự mình thực hiện việc làm mịn này. Thời điểm và cách thức thực hiện việc làm mịn sẽ được quyết định bởi Nhóm Scrum. Quá trình làm mịn thường không tiêu tốn nhiều hơn 10% lượng thời gian của Nhóm Phát triển. Nhóm Phát triển chịu trách nhiệm việc ước lượng. Product Owner có thể gây ảnh hưởng lên Nhóm băng cách giúp họ hiểu và lựa chọn trong các tình huống khó, nhưng người trực tiếp làm việc sẽ đưa ra con số ước lượng cuối cùng.

**Kiểm soát tiến độ hướng đến Mục tiêu**

Tại bất kì thời điểm nào, tổng khối lượng công việc còn lại để đạt mục tiêu phải được nhóm tổng kết. Prododuct Owner theo dõi lượng công việc còn lại ít nhất một lần trong buổi họp Sơ kết Sprint. Product Owner so sánh giá trị này với lượng thời gian còn lại tính từ các Sprint trước để đánh giá tiến độ hướng đến thời điểm dự tính hoàn thành mục tiêu của dự án. Thông tin này được minh bạch trước tất cả các bên hữu quan.

Rất nhiều phương pháp có thể được sử dụng để biểu thị tiến độ công việc, như các biểu đồ burndown, biểu đồ burnup hay các phương pháp dự báo khác. Các biện pháp này đã được chứng minh là rất hữu ích. Tuy nhiên, những thứ này không thay thế được tầm quan trọng của tính thực nghiệm trong quá trình phát triển. Trong các môi trường phức tạp, ta không thể biết được những điều sắp xảy ra. Chỉ có thể dựa trên những điều đã xảy ra để đưa ra các quyết định cho tương lai.

#### Sprint Backlog

Sprint Backlog là tập hợp các hạng mục Product Backlog được lựa chọn để phát triển trong Sprint, kèm theo một kế hoạch để chuyển giao phần tăng trưởng của sản phẩm và hiện thực hóa Mục tiêu Sprint. Sprint Backlog là một bản dự báo của Nhóm Phát triển về những chức năng sẽ có trong phần tăng trưởng sẽ được chuyển giao.

Sprint Backlog xác định công việc mà Nhóm Phát triển phải làm để biến các hạng mục Product Backlog thành phần tăng trưởng đạt tiêu chuẩn “Hoàn thành”. Sprint Backlog cho thấy tất cả những việc Nhóm Phát triển cần phải làm để tiến tới Mục tiêu Sprint.

Sprint Backlog là một kế hoạch với chi tiết vừa đủ để những thay đổi về tiến độ công việc có thể nhìn thấy được trong các cuộc họp Scrum Hằng ngày. Nhóm Phát triển chỉnh sửa Sprint Backlog trong suốt Sprint, và Sprint Backlog sẽ được cập nhật trong suốt thời gian đó. Sự cập nhật này xảy ra khi Nhóm Phát triển làm việc theo kế hoạch của họ, và hiểu rõ hơn về các công việc cần thiết để đạt Mục tiêu Sprint.

Mỗi khi có thêm việc mới, Nhóm Phát triển đưa vào Sprint Backlog. Khi công việc bắt đầu hay kết thúc, giá trị ước lượng về thời gian còn lại để hoàn tất công việc được cập nhật. Khi có phần nào đó của kế hoạch là không cần thiết, chúng sẽ bị bỏ đi. Chỉ có Nhóm Phát triển mới có thể thay đổi Sprint Backlog trong Sprint. Sprint Backlog là một bức tranh thời gian thực về công việc mà Nhóm Phát triển lên kế hoạch để hoàn thành trong Sprint, và nó cơ bản thuộc về Nhóm Phát triển.

**Kiểm soát tiến độ Sprint**

Tại bất kì thời điểm nào trong Sprint, tổng lượng thời gian còn lại để hoàn thành công việc có thể tính toán được. Nhóm Phát triển sẽ theo dõi con số này thường xuyên, ít nhất là vào các cuộc họp Scrum Hằng ngày. Dựa vào việc theo dõi này, họ có thể dự báo về các tiến độ đạt đến Mục tiêu Sprint. Theo đó, họ sẽ quản lý được tiến độ công việc. Scrum không quan tâm đến lượng thời gian bỏ ra trên các hạng mục Sprint Backlog. Scrum chỉ quan tâm đến lượng thời gian còn lại và thời điểm hoàn thành công việc.

#### So sánh Product backlog và Sprint Backlog

* Product backlog và Sprint backlog có nhiều điểm chung nhưng khác mục đích sử dụng. Product backlog được quản lí bởi product owner và cung cấp một cái nhìn ở mức cao về công việc mà team phát triển phải hoàn thành để tạo ra sản phẩm. Product owner đánh giá user story trong product backlog và cung cấp các thông tin chi tiết phù hợp trong Sprint planning meeting để team phát triển có thể đánh giá và hiện thực từng story.
* Ngược lại, team phát triển của bạn tạo Sprint backlog, chứa danh sách chi tiết tất cả các task mà team cần phải hoàn thành để kết thúc các user story trong Sprint. Trong product backlog, team của bạn đánh giá user story bằng story point còn trong Sprint backlog, team đánh giá task bằng số giờ thực hiện.
* Product owner cập nhật product backlog hành tuần, nhưng team phải cập nhật Sprint backlog hàng ngày.
* Product owner duy trình product backlog suốt dự án, nhưng team phát triển tạo Sprint backlog mới vào mỗi Sprint.
* Bảng sau tóm tắt những điểm khác nhau giữa product backlog và Sprint backlog.

|  |  |  |
| --- | --- | --- |
| **Mục** | **Product Backlog** | **Sprint Backlog** |
| Mức độ chi tiết | Ít chi tiết hơn | Rất chi tiết |
| Đơn vị ước lượng | Story points | Số giờ |
| Chủ sở hữu | Product owner | Team phát triển |
| Duyệt lại | Hàng tuần | Hàng ngày |
| Kéo dài | Dự án | Sprint |

#### Gói tăng trưởng

Gói tăng trưởng (Increment) là tập hợp tất cả các hạng mục Product Backlog đã được hoàn thành trong suốt Sprint hiện tại và những Sprint trước đó. Cuối một Sprint, một gói tăng trưởng phải thỏa mãn điều kiện “Hoàn thành”, có nghĩa là nó phải ở trạng thái sử dụng được và thỏa mãn định nghĩa của Nhóm Scrum về “Hoàn thành”. Gói tăng trưởng phải ở trạng thái dùng được để Product Owner có thể quyết định phát hành (release) nó.

### Định nghĩa “Hoàn thành”

Khi một hạng mục Product Backlog hoặc một Gói tăng trưởng cho là “Hoàn thành”, mọi người phải hiểu rõ “Hoàn thành” như thế nghĩa là thế nào. Mặc dù việc xác định rõ định nghĩa này hoàn toàn phụ thuộc vào từng Nhóm Scrum, nhưng mọi thành viên phải chia sẻ chung một cách hiểu về việc hoàn thành một công việc, để đảm bảo tính minh bạch và thông suốt. Đây chính là “Định nghĩa Hoàn thành” (Definition of Done) cho Nhóm Scrum; nó được dùng để đánh giá khi nào công việc thực sự hoàn thành trên mỗi gói tăng trưởng của sản phẩm.

Định nghĩa giống nhau sẽ chỉ dẫn cho Nhóm Phát triển nắm được số lượng hạng mục Product Backlog có thể được lựa chọn cho một Sprint. Mục đích của mỗi Sprint là để chuyển giao Gói tăng trưởng của các chức năng có tiềm năng chuyển giao được tuân thủ “Định nghĩa Hoàn thành: của Nhóm Scrum.

Mỗi Sprint, Nhóm Phát triển chuyển giao một Gói tăng trưởng. Phần tăng trưởng này phải là khả dụng, để Product Owner có thể lựa chọn và phát hành ngay lập tức. Mỗi gói tăng trưởng được cộng dồn vào các gói tăng trưởng trước đó và được kiểm thử toàn bộ để đảm bảo chúng làm việc tốt với nhau.

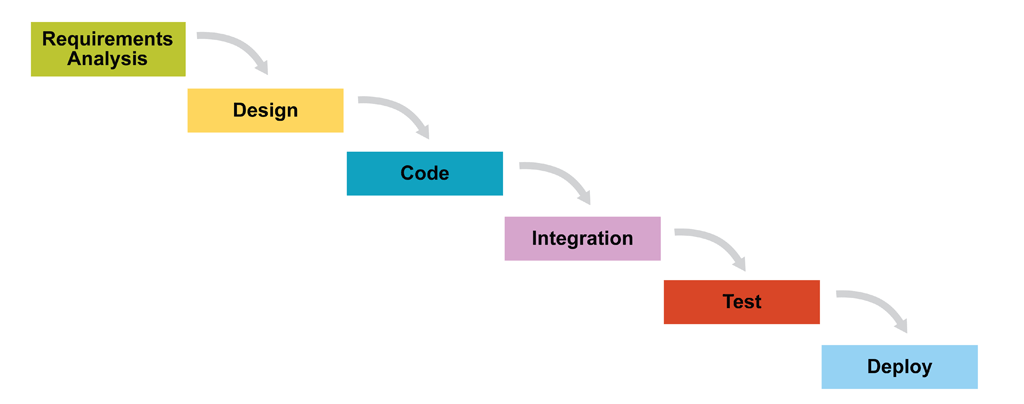
Khi Nhóm Scrum ngày càng trưởng thành thì “Định nghĩa Hoàn thành” càng được mở rộng với các chỉ tiêu khắt khe hơn để đạt chất lượng cao hơn.

### So sánh scrum và các quy trình phần mềm truyền thống

Với các phương pháp truyền thống, việc lập kế hoạch dự án (xác định những việc cần làm và thời gian kế thúc) dựa trên kinh nghiệm chứ không phải là môi trường làm trực tiếp. Và so với kế hoạch đó khi bắt tay vào xây dựng thì thường có độ trễ nhất định.

Scrum và quy trình thác nước (waterfall), xoắn ốc (spiral)

Mô hình thác nước chia dự án phần mềm gồm các giai đoạn: đặc tả yêu cầu, thiết kế hệ thống, cài đặt (lập trình), kiểm thử và bảo trì. Quy trình này dễ quản lý nhưng lại kém linh hoạt và không hiệu quả bởi nếu có sự thay đổi ở các giai đoạn sau sẽ ảnh hưởng rất lớn đến các giai đoạn trước.



*Hình 5. Chu trình phát triển phần mềm theo mô hình thác nước*

Quy trình xoắn ốc chia dự án thành các giai đoạn: lập kế hoạch, phân tích rủi ro, giao tiếp khách hàng, đánh giá lại, sản xuất và phân phối. Nó vẫn chưa được sử dụng rộng rãi.

Các quy trình phần mềm truyền thống thường có quá khá nhiều giai đoạn, nhiều thành phần, yếu tố trong suốt thời gian phát triển sản phẩm. Phương pháp scrum tránh điều này. (Xem bảng)

|  |  |  |  |
| --- | --- | --- | --- |
| **Đặc điểm** | **Waterfall** | **Spiral** | **Scrum** |
| Xác định các giai đoạn phát triển | Bắt buộc | Bắt buộc | Chỉ có giai đoạn lập kế hoạch và kết thúc |
| Sản phẩm cuối cùng | Được xác định trong quá trình lập kế hoạch | Được xác định trong quá trình lập kế hoạch | Xác định trong quá trình xây dựng dự án |
| Chi phí sản phẩm | Được xác định trong quá trình lập kế hoạch | Thay đổi cục bộ | Xác định trong quá trình xây dựng dự án |
| Ngày hoàn thành sản phẩm | Được xác định trong quá trình lập kế hoạch | Thay đổi cục bộ | Xác định trong quá trình xây dựng dự án |
| Đáp ứng với môi trường sử dụng | Trong kế hoạch ban đầu | Trong kế hoạch ban đầu | Xuyên suốt từ kế hoạch đến xây dựng và kết thúc |
| Kinh nghiệm trao đổi | Đào tạo trước cho đến khi bắt tay làm dự án | Đào tạo trước cho đến khi bắt tay làm dự án | Thực hiện trong quá trình làm dự án |
| Khả năng thành công | Thấp | Trung bình thấp | Cao |

### Quy trình triển khai Scrum

Có nhiều cách để triển khai, có thể sử dụng 10 bước sau:

- Bước 1: Thu nhập các đặc điểm của sản phẩm (Backlog) trong đơn đặt hàng. Đây là bước quan trọng nhất. Lập nên các đội làm việc, có thể tách thành các đội nếu cần thiết và thảo luận với nhau về nghiệp vụ cần làm. Sau đó bổ nhiệm một người vào vị trí Product owner, người này có khả năng trao đổi, bao quát công việc tốt, biết sắp xếp ưu tiên đúng thứ tự các nhiệm vụ. Sau đó tự tổ chức lại đội làm việc, đề xuất ra vị trí Scrum Master và thảo luận chi tiết các yêu cầu, sắp xếp chúng theo thứ tự ưu tiên.

• Bước 2: Ước lượng đầy các yêu cầu về sản phẩm đầu ra. Có ước lượng ở mức độ cao, chia sản phẩm thành số lượng các danh mục backlog. Tuy nhiên số lượng sẽ không chính xác được, về sau chúng sẽ được bổ sung. Tiếp đến là ước lượng chi tiết từng backlog, ước lượng số lượng các đội làm việc.

• Bước 3: Lên kế hoạch phát triển các vòng lặp Sprint. Sử dụng các cuộc trao đổi kế hoạch phát triển Sprint với tất cả các thành viên. Xác định khoảng thời gian sẽ phát triển một Sprint (thường là 30 ngày), mục tiêu của Sprint là gì, sẽ đạt được gì, phân tích các yêu cầu của Sprint một cách rõ ràng.

• Bước 4: Lên kế hoạch phát triển các nhiệm vụ của Sprint. Tất cả mọi người sẽ xác định ngân sách của Sprint đó, chia các đặc điểm thành các tác vụ nhỏ hơn, ước lượng số thời gian sẽ làm từng task (giờ), hoàn tất các yêu cầu và nhận dạng task quan trọng.

• Bước 5: Tạo ra không gian làm việc cộng tác cho tất cả mọi người. Thường sử dụng bảng trắng để vẽ nên những vấn đề cần thiết cho tất cả mọi người cùng đánh giá.

• Bước 6: Các thành viên bắt tay xây dựng từng Sprint. Lập trình, kiểm thử và điều chỉnh thời gian để có hiệu quả tốt nhất. Đôi khi có thể hủy bỏ một Sprint và quay lại với việc lập kế hoạch khác.

• Bước 7: Mọi người báo cáo kết quả để tiếp tục làm việc. Các báo cáo tập trung vào các vấn đề: đạt được những gì so với lần trao đổi trước; sẽ hoàn thành những gì trong lần trao đổi tiếp theo; có những trở ngại gì trong quá trình làm việc v.v.

• Bước 8: Tổng hợp kết quả trên biểu đồ. Đây là bức tranh tổng quát về những việc đã làm được, những việc chưa làm được, thời gian ước lượng còn lại và có thể điều chỉnh lại.

• Bước 9: Xem xét để hoàn tất. Khi các thành viên nói công việc đã hoàn thành có nghĩa là mọi thay đổi sẽ bị từ chối, đẩy lại cho vòng lặp sau.

• Bước 10: Đánh giá, phản ánh và lặp lại. Có các cuộc họp đánh giá lại Sprint của các thành viên. Sẽ trình bày những gì đạt được, phản hồi của khách hàng, xét thời hạn của Sprint. Nhìn lại biểu đồ ở bước 8 để xác định lại toàn bộ hệ thống và tiếp nhận những đóng góp, bổ sung để đưa tiếp vào các vòng lặp Sprint tiếp theo.

### Điểm mạnh của Scrum

Điểm mạnh nhất đó là việc linh hoạt, dự án không được cố định từ đầu về thời gian hoàn thành hay những yêu cầu mà nó sẽ được xác định khi phát triển thực tế.

Phân phối sản phẩm mềm dẻo: nội dung sản phẩm chuyển giao được xác định linh hoạt theo môi trường sử dụng thực tế. Thời gian biểu linh hoạt: có thể muộn hoặc sớm hơn so với kế hoạch ban đầu.

Chất lượng sản phẩm tốt và giảm rủi ro sản xuất, chi phí thấp. Khả năng trao đổi giữa khách hàng và nhà phát triển, giữa những thành viên trong đội được đặt lên mức cao.

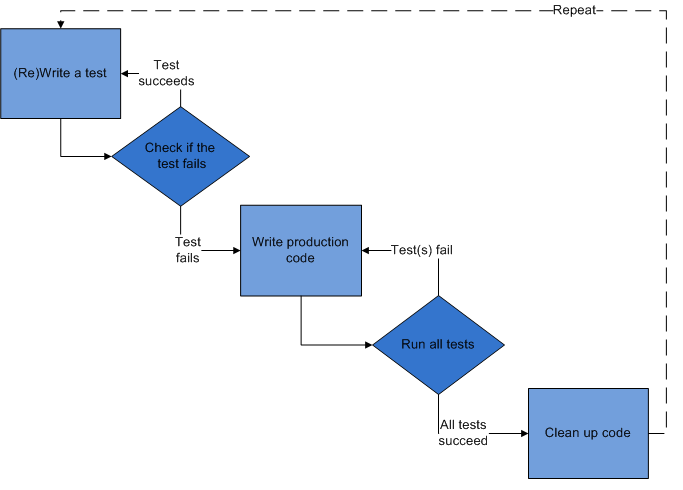
Tốc độ phát triển nhanh, tiết kiệm thời gian. Việc chuẩn bị hành động cho những thay đổi trong quá trình phát triển tốt hơn vì hầu như hàng ngày luôn có những buổi họp đánh giá lại ở những vòng lặp phát triển.

Các bugs và các vấn đề được phát hiện sớm hơn rất nhiều so với các phương pháp truyền thống bởi vì khách hàng được tham gia đánh giá rất nhiều và đầu ra của sản phẩm rất nhanh. Và khi đi sai hướng, có thể hủy ngay Sprint đó để quay lại với bản kế hoạch.

## Test-driven development (TDD)

Phát triển phần mềm theo hướng kiểm thử (TDD) là một quy trình phát triển phần mềm dựa trên sự lặp lại của một chu kỳ phát triển rất ngắn: đầu tiên các nhà phát triển viết một trường hợp kiểm thử tự động cho một cải tiến mong muốn hoặc chức năng mới, sau đó tạo ra đoạn mã tối thiểu để vượt qua kiểm thử đó, và cuối cùng điều chỉnh mã mới với các tiêu chuẩn chấp nhận được. Kent Beck, người được cho là người đã phát triển kỹ thuật, đã tuyên bố vào năm 2003 rằng TDD khuyến khích thiết kế đơn giản.

### Quy trình phát triển



*Hình 6. Quy trình phát triển của TDD*

#### Thêm một kiểm thử

Trong phát triển phát triển phần mèm hướng kiểm thử , mỗi tính năng mới bắt đầu bằng việc viết một kiểm thử. Thử nghiệm này chắc chắn phải thất bại vì nó được viết trước khi tính năng được code . ( Nếu nó không thất bại , hoặc là đề xuất tính năng " mới " đã tồn tại hoặc kiểm thử có vấn đề). Để viết một kiểm thử, các nhà phát triển phải hiểu rõ đặc điểm kỹ thuật của tính năng và yêu cầu. Các nhà phát triển có thể thực hiện điều này thông qua các user case và các user story để nắm bắt các yêu cầu và điều kiện ngoại lệ, và có thể viết các kiểm thử bằng bất cứ framework kiểm thử nào miễn là phù hợp với môi trường phát triển , hoặc cũng có thể sửa lại một kiểm thử có sẵn. Đây là một điểm khác biệt của phát triển phát triển phần mèm hướng kiểm thử so với viết kiểm thử đơn vị sau khi các mã được viết: nó khiến các nhà phát triển tập trung vào các yêu cầu trước khi viết mã, một sự khác biệt nhỏ nhưng quan trọng.

#### Chạy tất cả các kiểm thử và kiểm tra xem kiểm thử mới tạo có lỗi không?

Điều này đảm bảo rằng các kiểm thử hiện tại đều hoạt động chính xác, và các kiểm thử mới không thể nào chạy đúng nếu không có thêm mã nguồn mới. Bước này cũng kiểm tra chính kiểm thử : nó bác bỏ trường hợp rằng các kiểm thử mới luôn thành công và do đó không cần thiết. Các kiểm thử mới nên thất bại theo những lý do đã dự đoán trước.

#### Viết một số mã nguồn

Bước tiếp theo là viết một số mã nguồn để có thể vượt qua kiểm thử. Mã mới được viết ở giai đoạn này không cần phải hoàn hảo. Điều này có thể chấp nhận được vì các bước sau sẽ cải tiến mã nguồn.

Tại thời điểm này , mục đích duy nhất của mã được viết là phải vượt qua các kiểm thử .

#### Chạy thử nghiệm

Nếu tất cả các kiểm thử đều được vượt qua, lập trình viên có thể tự tin rằng mã đáp ứng tất cả các yêu cầu kiểm tra. Đây là một dấu hiệu tốt để bắt đầu bước cuối cùng của chu kỳ.

#### Cấu trúc lại code

Bây giờ mã phải được dọn dẹp nếu cần thiết. Di chuyển mã từ nơi nó được đặt thuận tiện để vượt qua các kiểm thử đến nơi mà nó nên thuộc về. Loại bỏ bất kỳ sự trùng lặp bạn có thể tìm thấy trong mã nguồn. Hãy chắc chắn rằng biến và tên hàm được sử dụng đại diện cho chức năng của chúng. Làm rõ bất kỳ cấu trúc có thể bị hiểu sai. Sử dụng bốn quy tắc của Kent Beck về thiết kế đơn giản làm tài liệu hướng dẫn, cũng như bất cứ điều gì khác mà bạn biết về cách viết mã sạch. Bằng cách chạy lại các kiểm thử , các nhà phát triển có thể tự tin rằng cấu trúc lại code không làm tổn hại đến bất kỳ chức năng hiện có .

#### Lặp lại

Bắt đầu với một thử nghiệm mới, lặp đi lặp lại chu kỳ để hoàn thành các chức năng.

### Một số lưu ý

#### Cấu trúc test

Test case với cấu trúc hợp lí đảm bảo tất cả các hành động cần thiết được hoàn thành, khiến các trường hợp kiểm thử dễ đọc hơn, và luồng thực hiện của chương trình trơn chu hơn. Với một cấu trúc thống nhất, các test case tự nó đã mang đầy đủ thông tin cần thiết. Một cấu trúc thường được áp dụng cho các test case có: (1) thiết lập, (2) thực hiện, (3) xác nhận, và (4) dọn dẹp.

* Thiết lập: Đặt các test unit cần kiểm thử (Unit Under Test - UUT) hoặc toàn hệ thống kiểm thử vào trạng thái cần kiểm thử.
* Thực hiện: Kích hoạt / hướng các UUT để thực hiện các hành vi mục tiêu và lấy tất cả kết quả đầu ra, chẳng hạn như giá trị các trả về và các thông số đầu ra. Bước này thường đơn giản.
* Xác nhận: Đảm bảo các kết quả của các kiểm thử là chính xác. Những kết quả này có thể bao gồm kết quả đầu ra rõ ràng lấy được qua bước Thực hiện hoặc các trạng thái thay đổi trong UUT.
* Dọn dẹp: Khôi phục các UUT hoặc toàn bộ hệ thống kiểm thử về trạng thái trước khi thực hiện kiểm thử. Việc phục hồi này cho phép một kiểm thử thực hiện nối tiếp ngay lập tức.

### Lợi ích

Một nghiên cứu thực hiện năm 2005 cho thấy rằng việc sử dụng TDD có nghĩa là viết nhiều kiểm thử và các lập trình viên viết nhiều kiểm thử hơn có xu hướng làm việc hiệu quả hơn.

Lập trình viên chỉ thuận sử dụng TDD dự án mới cho biết họ hiếm khi cảm thấy cần phải sử dụng một trình gỡ lỗi. Sử dụng kết hợp với một hệ thống kiểm tra phiên bản, khi thử nghiệm thất bại bất ngờ, quay trở lại mã cho phiên bản cuối cùng đã vượt qua tất cả các kiểm thử thường có thể được hiệu quả hơn việc debug.

TDD không chỉ xác nhận tính đúng đắn, nhưng cũng có thể điều khiển việc thiết kế của một chương trình. Bằng cách tập trung vào các test case đầu tiên, người ta phải tưởng tượng như thế nào các chức năng được sử dụng bởi khách hàng. Vì vậy, các lập trình viên thường lo lắng về giao diện trước khi thực hiện.

TDD cung cấp khả năng thực hiện các bước nhỏ khi cần thiết. Nó cho phép lập trình viên tập trung vào các nhiệm vụ chính là vượt qua được các kiểm thử. Trường hợp đặc biệt và xử lý lỗi không được coi là bước đầu, kiểm thử để tạo ra những trường hợp này không liên quan và được thực hiện một cách riêng biệt. TDD đảm bảo theo cách này là tất cả các mã được viết được kiểm thử ít nhất 1 lần. Điều này giúp đội ngũ lập trình, và người sử dụng tự tin hơn vào mã nguồn.

Trong khi TDD cần nhiều mã nguồn hơn là không sử dụng TDD vì mã nguồn của unit test, tổng thời gian thực hiện code có thể ngắn hơn. Một số lượng lớn các kiểm thử giúp hạn chế số lượng các lỗi trong các mã. Việc kiểm tra sớm và thường xuyên của các thử nghiệm giúp xác định dược sớm hơn các lỗi trong chu kỳ phát triển, ngăn ngừa chúng trở thành vấn đề nghiêm trọng. Loại bỏ các lỗi sớm trong quá trình phát triển thường giúp tránh được quá trình debug dài dòng và tẻ nhạt sau này trong dự án.

TDD có thể giúp tạo ra các mã nguồn có độ module hóa cao, linh hoạt, và có khả năng mở rộng cao hơn. Hiệu ứng này thường đi kèm về vì phương pháp đòi hỏi các nhà phát triển nghĩ về phần mềm về ở các đơn vị nhỏ có thể được viết và kiểm tra độc lập và tích hợp với nhau sau đó. Điều này dẫn đến các class nhỏ hơn, tập trung hơn, liên kết lỏng lẻo, và các interface rõ ràng hơn. Việc sử dụng các mẫu thiết kế hướng đối tượng cũng góp phần vào việc mô-đun hóa tổng thể của mã bởi vì mô hình này đòi hỏi các mã được viết sao cho các mô-đun có thể được chuyển đổi dễ dàng giữa các phiên bản giả cho các kiểm thử đơn vị và các phiên bản "thực sự" để triển khai sử dụng.

Bởi vì không có nhiều mã được viết hơn mức cần thiết để vượt qua một kiểm thử thất bại, kiểm thử tự động có xu hướng bao gồm tất cả đường đi trong mã nguồn. Ví dụ, một nhà phát triển TDD thêm một nhánh khác vào 1 phát biểu IF, đầu tiên nhà phát triển sẽ phải viết một trường hợp thử nghiệm không đúng cho nhánh mới thêm vào. Kết quả là, các bài kiểm tra tự động từ TDD có xu hướng rất kỹ lưỡng: họ phát hiện bất kỳ thay đổi bất ngờ trong hành vi của mã nguồn. Các vấn đề này có thể phát sinh khi sự thay đổi sau này trong chu kỳ phát triển vô tình thay đổi chức năng khác.

## Extreme programming

Extreme programming (XP) là một phương pháp phát triển phần mềm với mục đích nâng cao chất lượng phần mềm và đáp ứng với yêu cầu thay đổi của khách hàng. Như là một phương pháp của Agile, nó ủng hộ thường xuyên "phát hành" trong chu kỳ phát triển ngắn, với mục đích nâng cao năng suất và giới thiệu các điểm kiểm soát mà tại đó yêu cầu mới của khách hàng có thể được thông qua.

Các yếu tố khác XP bao gồm: lập trình theo cặp, thực hiện unit test của tất cả các mã, tránh lập trình các tính năng cho đến khi chúng thực sự cần thiết, cơ cấu quản lý phẳng, đơn giản và rõ ràng trong mã nguồn và liên lạc thường xuyên giữa khách hàng với các lập trình viên. Phương pháp lấy tên từ ý tưởng cho rằng các yếu tố có lợi của phương pháp phát triển phần mềm truyền thống được đưa đến mức "cực đoan (extreme).". Ví dụ, review code được coi là một thói quen có lợi; thực hiện với mức độ cao, mã nguồn có thể được xem xét liên tục, tiêu biểu là lập trình theo cặp.

Các nhà phê bình đã lưu ý một số nhược điểm tiềm năng, bao gồm cả vấn đề với các yêu cầu không ổn định, và thiếu tài liệu thiết kế kĩ thuật tổng thể.

### Khái niệm

#### Mục tiêu

XP cố gắng để giảm chi phí của việc thay đổi yêu cầu nhờ có nhiều chu kỳ phát triển ngắn, chứ không phải là một con đường dài. Trong học thuyết này, sự thay đổi là một phần tự nhiên, không thể tránh được và mong muốn của các dự án phát triển phần mềm, và cần được lên kế hoạch từ trước, thay vì cố gắng để xác định một tập hợp các yêu cầu ổn định. Extreme Programming cũng giới thiệu một số giá trị cơ bản, nguyên tắc và thực hành trên các khuôn khổ của Agile.

#### Các hoạt động

XP mô tả 4 hoạt động cơ bản được thực hiện trong quá trình phát triển phần mềm: code mã nguồn, kiểm thử, lắng nghe, và thiết kế. Mỗi hoạt động được mô tả dưới đây.

* Coding: Những người ủng hộ của XP cho rằng phần thực sự quan trọng của quá trình phát triển hệ thống là mã nguồn. Không có mã nguồn, không có sản phẩm. Mã nguồn cũng có thể được sử dụng để tìm ra các giải pháp phù hợp nhất. Mã hóa cũng có thể giúp truyền đạt suy nghĩ về các vấn đề lập trình. Một lập trình viên đối phó với một vấn đề lập trình phức tạp, hoặc thấy khó để giải thích các giải pháp cho các lập trình viên khác, có thể dùng một đoạn code để mô tả suy nghĩ.
* Kiểm thử: Phương pháp tiếp của XP là nếu ít kiểm thử có thể loại bỏ một vài sai sót, nhiều thử nghiệm có thể loại bỏ nhiều sai sót hơn.
  + Unit test xác định xem một tính năng hoạt động đúng như dự định. Một lập trình viên viết nhiều kiểm thử tự động mà họ có thể nghĩ rằng có thể "phá vỡ" mã nguồn; nếu tất cả các kiểm thử chạy thành công thì mã nguồn được xem là hoàn thiện. Mỗi đoạn mã được viết được kiểm tra trước khi chuyển sang viết mã cho các tính năng tiếp theo.
  + Kiểm thử chấp nhận xác minh rằng các yêu cầu được hiểu bởi các lập trình viên và đáp ứng yêu cầu thực tế của khách hàng.
  + Ban đầu, kiểm thử tích hợp toàn hệ thống được khuyến khích như là một hoạt động kết thúc mỗi ngày, nhăm phát hiện sớm các giao diện không tương thích, kết chúng lại. Tuy nhiên, kiểm thử tích hợp toàn hệ thống đã được giảm xuống còn hàng tuần, hoặc có thể ít hơn, tùy thuộc vào sự ổn định của giao diện tổng thể trong hệ thống.
* Lắng nghe: Lập trình viên phải lắng nghe những gì khách hàng cần hệ thống thực hiện, những gì "business logic" cần. Họ phải hiểu những nhu cầu này đủ để cung cấp cho khách hàng các thông tin phản hồi về các khía cạnh kỹ thuật, như thế nào vấn đề có thể được giải quyết, hoặc không thể được giải quyết.
* Thiết kế: Từ quan điểm đơn giản, ta có thể nói rằng việc phát triển hệ thống không cần nhiều hơn việc viết mã, kiểm thử và lắng nghe. Nếu những hoạt động này được thực hiện tốt, kết quả sẽ luôn luôn là một hệ thống hoạt động tốt. Trong thực tế, thường thì điều này không đạt được. Ta có thể đi một chặng đường dài mà không cần thiết kế nhưng ở một thời gian nhất định người sẽ gặp rắc rối. Hệ thống trở nên quá phức tạp và phụ thuộc trong hệ thống khiến chúng không còn được rõ ràng. Người ta có thể tránh điều này bằng cách tạo ra một cấu trúc thiết kế tổ chức logic trong hệ thống. Thiết kế tốt sẽ tránh được rất nhiều phụ thuộc trong một hệ thống; này có nghĩa là thay đổi một phần của hệ thống sẽ không ảnh hưởng đến các bộ phận khác của hệ thống

### Các giá trị

* Truyền đạt thông tin: Xây dựng hệ thống phần mềm đòi hỏi việc truyền đat các yêu cầu hệ thống tới các nhà phát triển của hệ thống. Với các phương pháp phát triển phần mềm chính thức, nhiệm vụ này được thực hiện thông qua tài liệu hướng dẫn. Kỹ thuật lập trình XP có thể được xem như là phương pháp để xây dựng nhanh chóng và phổ biến kiến thức giữa các thành viên của một nhóm phát triển. Mục đích là để cung cấp cho tất cả các nhà phát triển một cái nhìn chia sẻ về hệ thống phù hợp với quan điểm của người sử dụng của hệ thống. Để đạt được này, lập trình XP ủng hộ thiết kế đơn giản, ẩn dụ phổ biến, sự hợp tác giữa người dùng và lập trình viên, giao tiếp bằng lời nói thường xuyên, và phản hồi thường xuyên.
* Sự đơn giản: Lập trình XP khuyến khích bắt đầu với các giải pháp đơn giản nhất. Chức năng bổ sung có thể được thêm vào sau đó. Sự khác biệt giữa phương pháp này và các phương pháp phát triển hệ thống thông thường là tập trung vào thiết kế và code cho ngày hôm nay hơn là thay vì cho ngày mai, tuần tới, hoặc tháng tới. Điều này đôi khi tóm tắt thành phương pháp tiếp cận "Bạn không sẽ không cần nó" (YAGNI - You aren't gonna need it). Những người ủng hộ XP cho răng điều này đôi khi có thể dẫn đến những nỗ lực nhiều hơn vào ngày mai để thay đổi hệ thống. Code và thiết kế cho các yêu cầu không chắc chắn trong tương lai có nghĩa nguy cơ sử dụng tài nguyên vào những thứ có thể không cần thiết, trong khi đó trì hoãn tính năng rất quan trọng. Một thiết kế đơn giản với mã đơn giản có thể dễ dàng hiểu bởi hầu hết các lập trình viên trong đội ngũ.
* Phản hồi: Trong lập trình XP, thông tin phản hồi liên quan đến nhiều khía cạnh khác nhau trong quá trình phát triển hệ thống:
  + Thông tin phản hồi từ hệ thống: bằng cách viết các kiểm thử đơn vị, hoặc chạy các kiểm thử tích hợp định kỳ, các lập trình viên có thông tin phản hồi trực tiếp từ trạng thái của hệ thống sau khi thực hiện thay đổi.
  + Thông tin phản hồi từ khách hàng: Các kiểm thử chức năng (hay còn gọi là kiểm thử nghiệm thu) được viết bởi khách hàng và các tester. Họ sẽ nhận được thông tin phản hồi cụ thể về trạng thái hiện tại về hệ thống của họ. Đánh giá này được lên kế hoạch một lần trong mỗi hai hoặc ba tuần để khách hàng có thể dễ dàng chỉ đạo sự phát triển.
  + Thông tin phản hồi từ đội: Khi khách hàng đưa ra yêu cầu mới, đội trực tiếp đưa ra một ước tính thời gian cần để thực hiện.
  + Thông tin phản hồi liên quan chặt chẽ đến việc truyền thông và sự đơn giản. Sai sót trong hệ thống có thể dễ dàng truyền đạt bằng cách viết một kiểm thử đơn vị chứng minh rằng một phần nhất định của mã này sẽ lỗi. Các lập trình viên dựa vào thông tin phản hồi từ hệ thống rồi code lại phần bị lỗi. Một khách hàng có thể kiểm tra hệ thống định kỳ theo yêu cầu chức năng.
* Sự cam đảm: Can đảm cho phép các nhà phát triển cảm thấy thoải mái với refactoring mã của họ khi cần thiết. Điều này có nghĩa rà soát hệ thống hiện có và sửa đổi nó để thay đổi tương lai có thể được thực hiện dễ dàng hơn. Một ví dụ khác về sự can đảm là biết khi nào nên bỏ mã nguồn đi: can đảm để loại bỏ mã nguồn lỗi thời, không quan tâm là bao nhiêu nỗ lực đã được sử dụng để tạo ra mã nguồn.
* Sự tôn trọng: Tôn trọng bao gồm tôn trọng người khác cũng như lòng tự trọng. Lập trình viên không bao giờ thực hiện các thay đổi mà phá vỡ biên hệ thốn, hay làm cho kiểm thử đơn vị đã có thất bại, hoặc trì hoãn công việc của đồng nghiệp của họ. Các thành viên tôn trọng công việc của mình bằng cách luôn luôn phấn đấu cho chất lượng cao và tìm kiếm các thiết kế tốt nhất cho các giải pháp trong tầm tay thông qua tái cấu trúc. Áp dụng bốn giá trị trước đó dẫn đến sự tôn trọng thu được từ những người khác trong nhóm. Không ai trong nhóm cảm thấy không được đánh giá cao hay bỏ qua.

# Các công cụ hiện có

|  |  |  |
| --- | --- | --- |
| Tên công cụ | Điểm hạn chế | Điểm cần khắc phục ở tool |
| www.versionone.com | * Là một tool rất mạnh hỗ trợ agile, Version one có tất cả những gì cần có cho một team scrum. Điểm yêu duy nhất của nó là quá phức tạp, quá nhiều tình năng không cần thiết và giao diện nhìn tương đối rắc rối. * Không có tính tương tác cao do vẫn đơn thuần là 1 website, người dùng muốn cập nhật thông tin phải Refresh lại trang. | * Giao diện nhìn trực quan hơn, đơn giản hơn. * Chỉ cần cung cấp các chức năng cần thiết với 1 tool Scrum. * Sử dụng những công nghệ mới cho web app, tăng cường khả năng tương tác giữa các thành viên trong team nhờ vào cơ chế tự động cập nhật web app khi có bất cứ thay đổi nào tới project. |
| *www.scrumwise.com* | * Quá đơn giản, không có các tính năng quản lí bug, issue, user. * Tính năng Report còn sơ xài, thiếu xuất file báo cáo | * Cung cấp thêm các tính năng quản lí bug, issue, user. * Tính năng report với nhiều loại biểu đồ, định dạng. |