BÀI 2

QUẢN LÝ YÊU CẦU

❖ Chuẩn đầu ra của học phần:

L1	Triển khai giải pháp phần mềm ở sau pha thiết kế bằng việc sử dụng công cụ và quy trình của công nghệ thông tin nhằm đáp ứng yêu cầu của một dự án.
L2	Lựa chọn công cụ và quy trình về công nghệ điện toán an toàn để phát triển hiệu quả một dự án công nghệ thông tin bằng một phương pháp quản lý dự án tiên tiến như Scrum, Kanban, XP nhằm đáp ứng mục tiêu của các bên liên quan.
L3	Sử dụng công cụ và quy trình về công nghệ điện toán an toàn để phát triển hiệu quả một dự án công nghệ thông tin bằng một phương pháp quản lý dự án tiên tiến như Scrum, Kanban, XP nhằm đáp ứng mục tiêu của các bên liên quan.

❖ Nội dung học trực tuyến:

- Yêu cầu phần mềm và phân loại yêu cầu phần mềm
- Quy trình kỹ thuật yêu cầu
- Phân tích, đặc tả, thẩm định yêu cầu phần mềm

❖ Nội dung học trực tiếp:

Quản lý và phát triển yêu cầu trong dự án Agile (Product backlog và Spint backlog)

MỤC LỤC

2.1. Yêu cầu phần mềm

2.1.1. Giới thiệu về yêu cầu phần mềm

Yêu cầu (Requirement) là đặc điểm kỹ thuật về những gì mà phần mềm cần được thực hiện. Yêu cầu là những mô tả về cách thức hoạt động, hay còn có thể coi là một thuộc tính của hệ thống. Chúng có thể là một ràng buộc nào đó đối với quá trình phát triển của hệ thống.

Các yêu cầu phần mềm (Software requirements) do khách hàng/người sử dụng phần mềm nêu ra bao gồm: Các chức năng của phần mềm, hiệu năng của phần mềm, các yêu cầu về thiết kế và giao diện, các yêu cầu đặc biệt khác, ... Kỹ sư/chuyên viên phân tích hệ thống có nhiệm vụ ghi chép lại các yêu cầu về phần mềm để phục vụ cho dự án phần mềm.

Yêu cầu phần mềm mô tả hệ thống phần mềm sẽ hoạt động như thế nào hay hệ thống có thuộc tính gì. Yêu cầu phần mềm cũng có thể chính là các ràng buộc trong quá trình phát triển hệ thống phần mềm. Ngoài ra, yêu cầu phần mềm giúp đội phát triển hiểu khách hàng muốn gì và người dùng cuối sẽ tương tác với phần mềm như thế nào.

Yêu cầu phần mềm là lý do tại sao? và là vấn đề quan trọng thế nào? mà đội phát triển phải hiểu khách hàng muốn gì trước khi bắt đầu thiết kế và xây dựng hệ thống dựa trên máy tính.

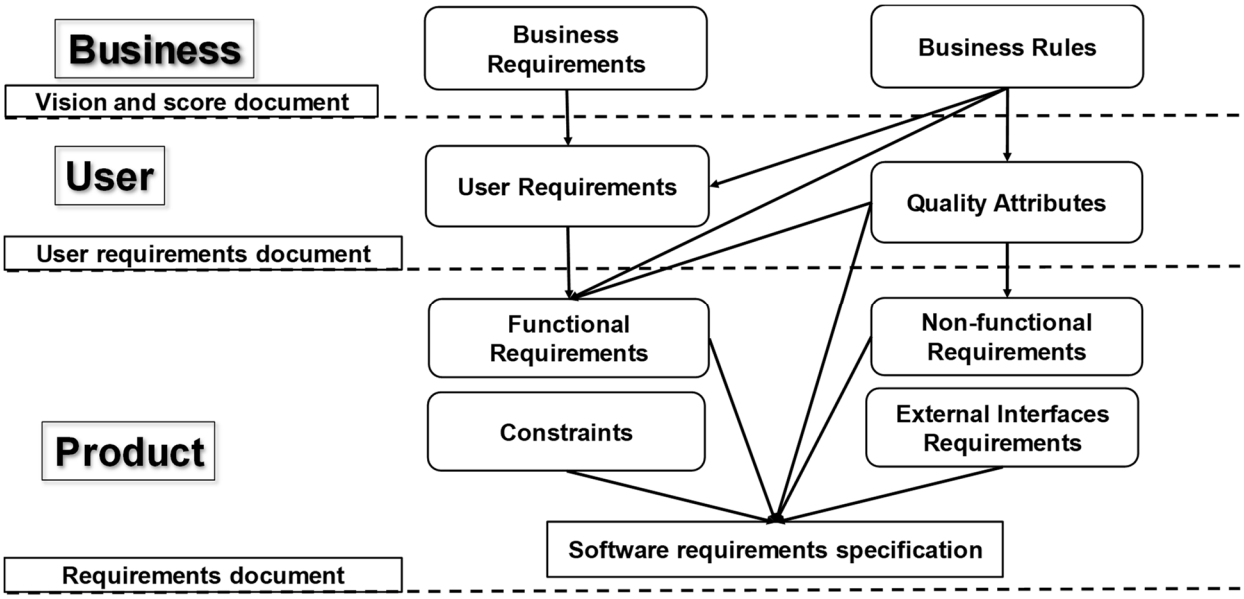
Có thể hiểu: ***Một yêu cầu là một đặc trưng của hệ thống, hay là sự mô tả những việc, mà hệ thống có khả năng thực hiện để hoàn thành mục tiêu của hệ thống.***

Hiểu được các yêu cầu của một vấn đề là một trong những nhiệm vụ khó khăn nhất mà một kỹ sư phần mềm (hay đôi khi được gọi là kỹ sư hệ thống hoặc "nhà phân tích") phải đối mặt. Hiểu được các yêu cầu phần mềm dẫn đến sự hiểu biết về sự hoạt động của phần mềm, khách hàng muốn gì và người dùng cuối sẽ tương tác với phần mềm như thế nào. Hiểu được các yêu cầu phần mềm là đóng

góp quan trọng trong việc thành bại của một dự án phát triển phần mềm. Hiểu được các yêu cầu giúp nhà phát triển hiểu được khách hàng muốn gì trước khi bắt đầu thiết kế và xây dựng hệ thống dựa trên máy tính.

2.1.2. Các loại yêu cầu

Hình 2.1. là mô hình minh họa về sự phân cấp và các loại yêu cầu.



Hình 2.1. Mô hình minh họa về các cấp và loại yêu cầu phần mềm.

Trong đó:

- **Vision and score document** (tầm nhìn và phạm vi sản phẩm). Tài liệu trình bày tổng quang về những gì sản phẩm cuối cùng có thể đạt được, xác định các lợi ích nghiệp vụ mà hệ thống sẽ cung cấp. Trình bày về phạm vi dự án, ranh giới giữa những gì vào và ra bao gồm các giới hạn và ràng buộc; tạo điều kiện thuận lợi cho việc đưa ra các cam kết và ưu tiên của dự án; điều kiện tiên quyết để quản lý phạm vi dự án.

- Business Requirements. Yêu cầu nghiệp vụ của phần mềm

- **Business Rules.** Các quy tắc nghiệp vụ

- **User Requirements** (các yêu cầu người sử dụng).

- Quality Attributes (các thuộc tính chất lượng): một số nguồn cung cấp quy tắc chất lượng phổ biến:

+ **ISO 25010 (SQuaRE)** là một tiêu chuẩn quốc tế liên quan đến phần mềm và hệ thống thông tin, nó cung cấp một cách tiếp cận hệ thống chất lượng phần mềm. Tiểu chuẩn này đề cập đến các yếu tố chất lượng như tính an toàn, tính bảo mật, hiệu suất, tương tác người dùng, và nhiều yếu tố khác.
+ **IEEE 1471** là một tiêu chuẩn quốc tế về kiến trúc phần mềm, nó cung cấp một framework cho việc quản lý kiến trúc của các hệ thống phần mềm. Trong IEEE 1471, các yếu tố chất lượng như khả năng mở rộng, dễ bảo trì, tính di động, và khả năng mở rộng thường được đề cập.
+ **Các phương pháp và mô hình phát triển phần mềm** như Agile, Waterfall, và DevOps thường đề cập đến các yếu tố chất lượng như tính linh hoạt, khả năng tích hợp, và thời gian phát triển.
+ **Các phương pháp quản lý dự án** như PMBOK (Project Management Body of Knowledge) và PRINCE2 cung cấp các hướng dẫn và tiêu chuẩn liên quan đến chất lượng trong quản lý dự án, bao gồm cả việc xác định và quản lý các yếu tố chất lượng của sản phẩm phần mềm.
+ Best practices (thực tiễn sản xuất) từ cộng đồng ngành công nghiệp phần mềm thường chia sẻ các best practices và kinh nghiệm về việc xác định và đảm bảo chất lượng phần mềm. Các tài liệu và tài nguyên từ các cộng đồng như công nghệ phần mềm **IEEE, ACM**, và các diễn đàn trực tuyến cũng là nguồn thông tin hữu ích.

- Functional Requirements: Yêu cầu chức năng

- Non-functional Requirements: Yêu cầu phi chức năng

- Constraints: Các ràng buộc

- External Interfaces Requirements: Yêu cầu về giao diện bên ngoài

- System Requirements: Yêu cầu hệ thống

- Software requirements specification: Mô tả (đặc tả) yêu cầu phần mềm

- Requirements document: Tài liệu yêu cầu

2.1.2.1. Yêu cầu người sử dụng

Yêu cầu người sử dụng (User requirements): Dành cho khách hàng, được diễn đạt bằng ngôn ngữ tự nhiên và sơ đồ về dịch vụ hệ thống cần cung cấp và các ràng buộc trong hoạt động của nó sao cho đơn giản, dễ hiểu đối với người sử dụng, hạn chế sử dụng hay áp dụng các thuật ngữ có kiến thức chi tiết về kĩ thuật/tin học. Nên mô tả các cầu người sử dụng theo:

+ Yêu cầu chức năng;

+ Yêu cầu phi chức năng.

Do yêu cầu người sử dụng được diễn đạt bằng ngôn ngữ tự nhiên nến có các hạn chế nhất định: Quá mềm dẻo dẫn đến không rõ ràng, thiếu chính xác, đôi khi xả ra tình trạng nhập nhằng.

2.1.2.2. Yêu cầu hệ thống

Yêu cầu hệ thống (System requirements) được mô tả đủ chi tiết hơn so với yêu cầu người dùng về các dịch vụ hệ thống cung cấp, các đặc trưng hệ thống cần có và nó cũng được dùng như một hợp đồng giữa khách hàng và nhà phát triển. Thường sử dụng các mô hình để mô tả (phương pháp trình bày mô hình hóa).

Lưu ý: Có nhiều quan điểm mặc định rằng yêu cầu người dùng chính là yêu hệ thống và không có sự phân biệt giữa các yêu cầu này.

2.1.2.3. Yêu cầu chức năng

Yêu cầu về chức năng (functional requirements) mô tả các chức năng hay các dịch vụ mà hệ thống phần mềm có thể thực hiện nhằm hỗ trợ cho nghiệp vụ

của khách hàng/người sử dụng. Yêu cầu chức năng là danh sách các công việc sẽ được thực hiện trên máy tính cùng với các thông tin mô tả tương ứng. Các yêu cầu chức năng bao gồm: Yêu cầu chức năng nghiệp vụ và Yêu cầu chức năng hệ thống.

*** Yêu cầu chức năng nghiệp vụ:**

Yêu cầu chức năng nghiệp vụ thuộc nhóm các yêu cầu chức năng. Yêu cầu chức năng nghiệp vụ bao gồm các chức năng của phần mềm tương ứng với công việc có thật trong thế giới thực mà khách hàng/người sử dụng cần phải thực hiện. Yêu cầu chức năng nghiệp vụ phần mềm thực hiện đảm bảo tính đúng đắn và phù hợp với pháp luật sở tại. Các yêu cầu chức năng nghiệp vụ bao gồm:

1) Chức năng lưu trữ: Tương ứng với công việc ghi chép thông tin trên sổ sách trong thế giới thực. Yêu cầu này đòi hỏi kèm theo các quy định, mẫu biểu ghi chép của khách hàng;
2) Chức năng tra cứu: Tương ứng với công việc tìm kiếm, theo dõi và xem thông tin/hoạt động về đối tượng quản lý của khách hàng.
3) Chức năng tính toán: Tương ứng với công việc tính toán trong thế giới thực theo đúng nhiệm vụ cần thực hiện của khách hàng/người sử dụng. Các công thức, phương pháp tính cần thực hiện theo theo đúng các quy định nghiệp vụ (có quy định/mẫu biểu/công thức kèm theo cho trước);
4) Chức năng kết xuất: Tương ứng với công việc lập các loại báo cáo (tổng hợp, chi tiết, …) theo các biểu mẫu cho trước trong thực tế.

*** Yêu cầu chức năng hệ thống:**

Yêu cầu chức năng hệ thống thuộc nhóm các yêu cầu chức năng. Yêu cầu chức năng hệ thống bao gồm các chức năng không tương ứng với bất kỳ công việc nào trong thế giới thực, hay là các chức năng phần mềm được phát sinh thêm khi thực hiện công việc trên máy tính thay vì trong thế giới thực. Nó hỗ trợ cho quá trình thực hiện các chức năng nghiệp vụ và người dùng không nhìn thấy

được. Yêu cầu chức năng hệ thống liên quan đến tính bảo mật và tính an toàn. Các yêu cầu chức năng hệ thống bao gồm:

1) Chức năng môi trường: Định cấu hình thiết bị (máy tính, máy in, thiết bị nhập/xuất, …), ngày giờ, số người làm việc, …;
2) Chức năng mô phỏng: Mô phỏng các hoạt động của thế giới thực trên máy tính/thiết bị chuyên dụng, giúp cho khách hàng/người sử dụng có cái nhìn trực quan về lĩnh vực trong thực tế;
3) Chức năng phân quyền: Phân quyền sử dụng giữa các loại/đối tượng người dùng. Đây chính là việc chỉ định người dùng có quyền thực hiện nhiệm vụ/chức năng gì. Việc phân quyền tương ứng với nhiệm vụ của đối tượng người dùng trong thực hiện các nhiệm vụ thực tế;
4) Chức năng sao lưu: Sao lưu, khôi phục dữ liệu. Đây là một chức năng quan trọng và không thể thiếu với mỗi hệ thống phần mềm.

2.1.2.4. Yêu cầu phi chức năng

Yêu cầu phi chức năng (non-functional requirements) bao gồm các yêu cầu liên quan đến chất lượng phần mềm, là sự ràng buộc cách thức thực hiện các yêu cầu chức năng. Yêu cầu phi chức năng mô tả các yêu cầu liên quan đến các ràng buộc như: độ tin cậy, thời gian đáp ứng, độ an toàn, các tiêu chuẩn chất lượng, về môi trường, chuẩn sử dụng, qui trình phát triển, bản quyền, liên kết, … Mỗi yêu cầu phi chức năng có thể thuộc vào yêu cầu của người sử dụng hoặc của nhà phát triển (đại diện là các kỹ sư/kỹ thuật viên/chuyên viên tin học).

Trong thực tế, có những yêu cầu rất khó phân biệt được một cách rõ ràng là yêu cầu này thuộc vào yêu cầu chức năng hay yêu cầu phi chức năng.

Mỗi yêu cầu phi chức năng có thể thuộc một trong các yêu cầu sản phẩm, các yêu cầu về tổ chức hay các yêu cầu bên ngoài:

- Các yêu cầu về sản phẩm xác định ứng xử của sản phẩm về hiệu năng, khả năng sử dụng, độ tin cậy … của sản phẩm. Các yêu cầu về sản phẩm như:

Yêu cầu khả dụng (Usability requirements); Yêu cầu hiệu quả (Efficiency requirements), Yêu cầu hiệu năng (Performance requirements), Yêu cầu về không gian (Space requirements); Yêu cầu tin cậy (Reliability requirements); Yêu cầu khả chuyển (Portability requirements); Yêu cầu về tính toàn vẹn (Integrity requirements); Yêu cầu về khả năng ghi nhận (Integrity requirements); …

- Các yêu cầu về tổ chức (Organizational requirements) được lấy từ những chính sách và quy tắc của khách hàng hoặc tổ chức sử dụng hệ thống. Các yêu cầu về tổ chức như: Yêu cầu chuyển giao (Delivery requirements); Yêu cầu triển khai (Implement requirements); Yêu cầu về chuẩn (Standards requirements);

- Các yêu cầu ngoài (External requirements) hay ngoại lai được xác định từ các tác nhân ngoài của hệ thống. Các yêu cầu ngoài như: Yêu cầu hoạt động bên trong (Interoperability requirements); Yêu cầu đạo đức (Ethical requirements); Yêu cầu pháp lý (Legislative requirements).

Yêu cầu phi chức năng được cho có thể quan trọng hơn yêu cầu chức năng và phát triển theo thời gian. Nếu yêu cầu phi chức năng không được đáp ứng → hệ thống trở nên vô dụng. Đối với người sử dụng/khách hàng, các sản phẩm phần mềm thường tối thiểu phải đạt được các yêu cầu phi chức năng như sau:

- Yêu cầu khả dụng (Usability requirements): Là một trong những yêu cầu phi chức năng quan trọng bậc nhất, Usability chính là khả năng "dễ sử dụng" của hệ thống. Nó đề cập tới quy mô nguồn lực để đào tạo nhân viên mới sử dụng hệ thống. Là khả năng của phần mềm có thể hiểu được, học được, sử dụng được và hấp dẫn người sử dụng trong từng trường hợp sử dụng cụ thể.

- Yêu cầu hiệu quả (Efficiency requirements): Đề cập tới tài nguyên phần cứng cần để thực hiện các chức năng của phần mềm. Là khả năng của phần mềm có thể hoạt động một cách hợp lý, tương ứng với lượng tài nguyên nó sử dụng, trong điều kiện cụ thể.

- Yêu cầu hiệu năng (Performance requirements): Đề cập đến hiệu suất hoạt động của hệ thống, và thường được đo lường bằng những tiêu chí sau:

+ Thời gian phản hồi cho một giao dịch (transaction);

+ Số lượng giao dịch thực hiện được trong mỗi giây;

+ Công suất (capacity) được hiểu như số lượng giao dịch mà hệ thống có thể lưu trữ/ thực hiện cho mỗi đối tượng.

- Yêu cầu về không gian (Space requirements): Đề cập đến các yếu tố về tài nguyên sử dụng như: RAM, dung lượng DB, …

- Yêu cầu tin cậy (Reliability requirements): Đề cập tới lỗi khi cung cấp dịch vụ: tỉ lệ lỗi, thời gian hệ thống chết. Khả năng chịu lỗi, khả năng tự phục hồi.

- Yêu cầu khả chuyển (Portability requirements): Nếu phần mềm cài ở môi trường mới, có giữ được các tính năng như cũ không. Thích nghi với nhiều môi trường (lớp nền, services,…), cùng nhau tồn tại trong môi trường (no conflict), thay thế cho phần mềm khác có cùng chức năng.

- Yêu cầu về tính toàn vẹn (Integrity requirements): Các yêu cầu này đề cập về "độ chính trực" của dữ liệu. Tức độ chính xác, xác thực của dữ liệu. Yêu cầu dữ liệu phải được tính toán chính xác tại tất cả các thời điểm của hệ thống.

- Yêu cầu về khả năng ghi nhận (Integrity requirements): Đề cập đến khả năng ghi nhận lại các tác vụ đã thực hiện trên hệ thống, nhằm mục đích kiểm tra.

- Yêu cầu hoạt động bên trong (Interoperability requirements): Yêu cầu về khả năng tương tác;

- Yêu cầu đạo đức (Ethical requirements): phù hợp với nguyên tắc đạo đức nghề nghiệp đối với kỹ sư phần mềm;

- Yêu cầu pháp lý (Legislative requirements):

+ Yêu cầu cá nhân (Privacy requirements): Yêu cầu quyền riêng tư;

+ Yêu cầu an toàn (Safety requirements)

2.1.2.5. Đặc trưng của yêu cầu phần mềm

Yêu cầu phần mềm có các đặc trưng cơ bản sau:

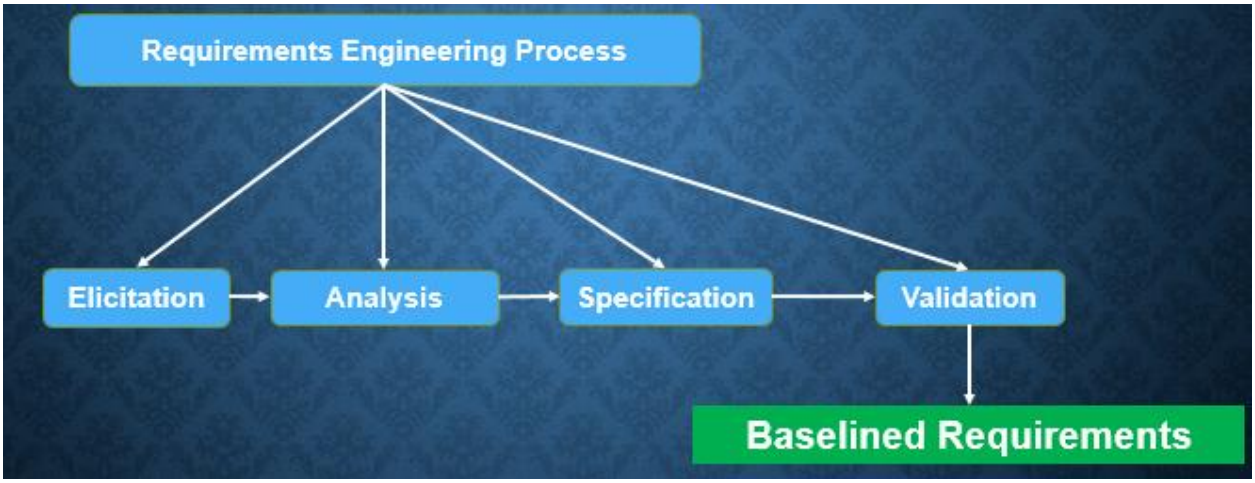
- Khả thi
- Có giá trị
- Không nhập nhằng
- Dễ kiểm chứng
- Dễ biến đổi
- Toàn vẹn
- Đầy đủ
- Lần vết được

2.1.3. Quy trình kỹ thuật yêu cầu phần mềm

Phổ rộng của các kỹ thuật xác định yêu cầu phần mềm dẫn đến sự hiểu biết tường minh về các yêu cầu được gọi là kỹ thuật yêu cầu (requirements engineering) hay kỹ thuật yêu cầu phần mềm (softwware requirements engineering). Từ quan điểm quy trình phần mềm, kỹ thuật yêu cầu là một hành động kỹ thuật phần mềm chính bắt đầu trong hoạt động giao tiếp và tiếp tục vào hoạt động mô hình hóa. Nó phải được điều chỉnh cho phù hợp với nhu cầu của quá trình, dự án, sản phẩm và những người tham gia dự án.

2.1.3.1. Các thành phần cơ bản của kỹ thuật yêu cầu phần mềm

Kỹ thuật yêu cầu là một cách tiếp cận theo hướng quy trình, theo đúng trình tự các bước được xác định trước để có tài liệu yêu cầu và duy trì các yêu cầu phần mềm trong suốt vòng đời phát triển phần mềm. Kỹ thuật yêu cầu phần mềm được tạo thành từ hai quy trình chính: phát triển yêu cầu (Requirements Development) và quản lý yêu cầu (Requirements Management) (Hình 2.2).



Hình 2.2. Các thành phần cơ bản của kỹ thuật yêu cầu phần mềm.

Phát triển yêu cầu bao gồm tất cả các hoạt động liên quan đến việc phát hiện, phân tích, xác định và xác thực các yêu cầu:

- Phát hiện các yêu cầu phần mềm (Requirements elicitation)
- Phân tích các yêu cầu phần mềm (Requirements analysis)
- Mô tả các yêu cầu phần mềm (Requirements specification)
- Thẩm định yêu cầu phần mềm (Requirements validation)

2.1.3.2. Phát hiện các yêu cầu phần mềm

Phát hiện yêu cầu liên quan đến việc các yêu cầu đến từ đâu và làm thế nào chúng có thể được thu thập. Gợi mở yêu cầu là giai đoạn đầu tiên trong việc xây dựng sự hiểu biết về vấn đề mà phần mềm cần giải quyết. Đây là một hoạt động và là nơi các bên liên quan được xác định và thiết lập các mối quan hệ giữa nhóm phát triển (thường ở dạng kỹ sư yêu cầu) và khách hàng. Các nhiệm vụ cần thực hiện:

- Đánh giá tính khả thi về kỹ thuật và nghiệp vụ của phần mềm định phát triển (dựa trên báo cáo khả thi);

- Tìm kiếm các nhân sự (chuyên gia, người sử dụng, các bên liên quan) có những hiểu biết sâu sắc nhất, chi tiết nhất về hệ thống giúp chúng ta xác định yêu cầu phần mềm;

- Xác định môi trường kỹ thuật trong đó sẽ triển khai phần mềm;

- Xác định các ràng buộc về lĩnh vực ứng dụng của phần mềm (giới hạn về chức năng/hiệu năng phần mềm);

- Xác định các phương pháp sử dụng để phát hiện các yêu cầu phần mềm: Phỏng vấn; Làm việc nhóm; Các buổi họp; Gặp gỡ đối tác, v.v.

- Thu hút sự tham gia của nhiều chuyên gia, khách hàng để chúng ta có được các quan điểm xem xét phần mềm khác nhau từ phía khách hàng;

- Xác định các yêu cầu còn nhập nhằng để làm mẫu thử;

- Thiết kế các kịch bản sử dụng của phần mềm để giúp khách hàng định rõ các yêu cầu chính.

Một số phương pháp thực hiện khảo sát phổ biến:

(i) Quan sát: theo dõi các hoạt động đang diễn ra ở thế giới thực thuộc lĩnh vực có liên quan. Có thể kết hợp với việc ghi âm, ghi hình để làm tài liệu tham khảo về sau đối với những tình huống mang tính phức tạp, quan trọng, đòi hỏi sự chính xác cao.

(ii) Phỏng vấn trực tiếp: tổ chức phỏng vấn với các nhà chuyên môn có liên quan tới nghiệp vụ công tác. Quá trình phỏng vấn được thược hiện bắt đầu từ cấp lãnh đạo cấp cao, và cuối cùng đến chuyên viên phụ trách/thực hiện trực tiếp với các vị trí công việc. Khi phỏng vấn, có thể kết hợp sử dụng các bảng câu hỏi có sẵn các câu trả lời cho đối tượng được phỏng vấn lựa chọn, …

(iii) Thu thập thông tin, tài liệu: Cần thiết phải thu thập các tài liệu liên quan đến nghiệp vụ công tác, bao gồm: quy định, mẫu biểu, các công thức tính toán, và tất cả các mẫu giấy tờ có liên quan.

Lưu ý khi tiến hành xếp loại các nghiệp vụ đã thu thập: các yêu cầu nghiệp vụ được vào 1 trong 4 loại nghiệp vụ sau:

- Nghiệp vụ lưu trữ;

- Nghiệp vụ tra cứu;

- Nghiệp vụ tính toán;

- Nghiệp vụ tổng hợp, thống kê.

2.1.4. Phân tích, đặc tả và thẩm định yêu cầu phần mềm

Phân tích các yêu cầu phần mềm bao gồm các nhiệm vụ:

- Phân loại các yêu cầu phần mềm và sắp xếp chúng theo các nhóm liên quan;

- Khảo sát tỉ mỉ từng yêu cầu phần mềm trong mối quan hệ của nó với các yêu cầu phần mềm khác;

- Kiểm tra từng yêu cầu phần mềm theo các tính chất:

+ Phù hợp;

+ Đầy đủ;

+ Rõ ràng,

+ Không trùng lặp

- Phân cấp các yêu cầu phần mềm theo dựa trên nhu cầu và đòi hỏi khách hàng/người sử dụng;

- Đánh giá từng yêu cầu phầm mềm để xác định chúng có khả năng thực hiện được trong môi trường kỹ thuật hay không, có khả năng kiểm định các yêu cầu phần mềm hay không;

- Đánh giá các rủi ro có thể xảy ra với từng yêu cầu phần mềm;

- Đánh giá thô (tương đối) về giá thành và thời gian thực hiện của từng yêu cầu phần mềm trong giá thành sản phẩm phần mềm và thời gian thực hiện phần mềm;

- Giải quyết tất cả các bất đồng về yêu cầu phần mềm với khách hàng/người sử dụng trên cơ sở thảo luận và thương lượng các yêu cầu đề ra.

Mô tả các yêu cầu phần mềm hay còn được gọi là đặc tả yêu cầu phần mềm. Khi xây dựng tài liệu mô tả yêu cầu, có thể sử dụng tới các công cụ như:

mô hình hóa, mô hình toán học hình thức (a formal mathematical model), tập hợp các kịch bản sử dụng, các nguyên mẫu hoặc bất kỳ một tổ hợp các công cụ nói trên để giúp cho tài liệu đặc tả yêu cầu được tường minh.

Các phương pháp đặc tả:

- Đặc tả phi hình thức (Informal specifications): viết bằng ngôn ngữ tự nhiên;

- Đặc tả hình thức (Formal specifications): viết bằng tập các ký pháp có các quy định về cú pháp (syntax) và ngữ nghĩa (sematic) rất chặt chẽ, thí dụ ký pháp đồ họa dùng các lưu đồ.

Các tiêu chí đánh giá chất lượng của hồ sơ đặc tả:

- Tính rõ ràng, chính xác;

- Tính phù hợp;

- Tính đầy đủ, hoàn thiện

Các thành phần của hồ sơ đặc tả:

- Đặc tả vận hành hay đặc tả chức năng (Operational specifications) mô tả các hoạt động của hệ thống phần mềm sẽ xây dựng:

+ Các dịch vụ mà hệ thống phải cung cấp,

+ Hệ thống sẽ phản ứng với đầu vào cụ thể ra sao,

+ Hành vi của hệ thống trong các tình huống đặc biệt.

- Đặc tả mô tả hay đặc tả phi chức năng (Descriptive specifications) và các ràng buộc:

+ Đặc tả các đặc tính, đặc trưng của phần mềm, các ràng buộc về các dịch vụ hay các chức năng hệ thống cung cấp như thời gian, ràng buộc về các quá trình phát triển, các chuẩn,…

Các nhiệm vụ thực hiện trong quá trình thẩm định yêu cầu phần mềm:

- Rà soát tài liệu yêu cầu để phát hiện các sai sót. Việc ra soát do đồi rà soát thực hiện.

- Tạo bản mẫu: Tạo mẫu thường được sử dụng để xác thực cách giải thích của kỹ sư yêu cầu về các yêu cầu hệ thống, cũng như để gợi ra các yêu cầu mới. Ưu điểm của nguyên mẫu là chúng có thể giúp dễ dàng hơn trong việc giải thích các giả định của kỹ sư yêu cầu và đưa ra phản hồi hữu ích;

- Xác nhận mô hình: Chất lượng của các mô hình được phát triển trong quá trình phân tích cần được xác nhận. Việc xác nhận đảm bảo trong mô hình, các đường truyền thông tin tồn tại giữa các đối tượng có liên quan đến việc trao đổi dữ liệu;

- Thực hiện kiểm thử chấp nhận đối với các yêu cầu. Điều này là có thể gặp khó khăn với các yêu cầu phi chức năng.

2.2. Quản lý và phát triển yêu cầu trong dự án Agile

2.2.1. Yêu cầu trong dự án Agile

Vận dụng các mô hình quy trình truyền thống (thác nước, xoắn ốc,…), sản phẩm của dự án luôn bắt đầu với những sản phẩm lớn và phân chia chúng xuống mức mà bạn có thể bắt đầu ước tính chính xác nhất thời gian, chi phí và nguồn lực để xây dựng phạm vi công việc. WBS (Work Breakdown Structure) là công cụ giúp nhóm dự án phân rã và tổ chức công việc thành các thành phần nhỏ hơn, dễ quản lý hơn. Mỗi thành phần trong WBS được gọi là một "work package" và bao gồm các nhiệm vụ cụ thể cần thực hiện để hoàn thành dự án. Các yêu cầu phần mềm được sắp xếp và phân rã thành các thành phần nhỏ hơn trong WBS. Điều này giúp đảm bảo rằng mọi yêu cầu đều được phân bố và quản lý một cách cụ thể và hiệu quả.

Trong Agile, không sử dụng WBS, thay vào đó, Agile sử dụng các Theme, Epic, User story, Story map để giúp xác định nội dung, cách thức hoạt động của yêu cầu và sau đó được thực hiện trong mỗi sprint để hoàn thành dự án. Các công cụ này được sử dụng để xác định yêu cầu thường được thiết kế để đảm bảo

sự linh hoạt và tương tác liên tục giữa các thành viên trong nhóm dự án và khách hàng.

Nhóm Agile lập kế hoạch và xây dựng dần dần dựa trên producs backlog. PO (Product Owner - Chủ sở hữu sản phẩm) liên lạc với nhóm để giải thích công việc và cũng để xem xét công việc sắp tới. Điều này giúp nhóm hiểu rõ hơn về công việc để cải thiện các ước tính.

Nhóm Agile liên lạc với các bên liên quan/khách hàng để xem xét và bổ sung các yêu cầu mới vào backlog.

Lưu ý: Mặc dù nhóm và các bên liên quan khác có thể đóng góp vào thông tin backlog, nhưng PO chịu trách nhiệm đảm bảo rằng nhóm và các bên liên quan khác sắp xếp và hiểu được các yêu cầu được ghi trên backlog.

2.2.2. User story

2.2.2.1. Vai trò của User story

User story (câu chuyện của người dùng) là một cách mô tả yêu cầu từ góc độ của người dùng cuối trong Agile. Mỗi user story là những mô tả ngắn gọn, đơn giản về một tính năng hoặc chức năng cụ thể mà người dùng mong đợi, được kể từ góc nhìn của người mong muốn có khả năng mới, thường là người dùng hoặc khách hàng của hệ thống.

* Ví dụ:

"Người dùng muốn có thể đăng nhập vào hệ thống bằng tên người dùng và mật khẩu của họ."

User story cho phép nhóm hiểu chính xác những gì người dùng cuối cần và muốn cũng như cách sử dụng phần mềm. Nhiều khi, phép ẩn dụ được sử dụng để giúp xác định câu chuyện của người dùng và đạt được sự hiểu biết chung.

Câu chuyện của người dùng ban đầu được sử dụng trong XP framework (Extreme Programming framework) như một công cụ để phân chia yêu cầu thành

các đơn vị nhỏ hơn và hiện được sử dụng trong hầu hết mọi khung công tác Agile.

Câu chuyện của người dùng không chỉ là một cách hiệu quả để xác định xem việc thực hiện sẽ như thế nào mà còn được sử dụng để xác định số lượng công việc có thể hoàn thành trong bất kỳ lần chạy nước rút hoặc lặp lại nhất định nào.

Câu chuyện cho phép nhóm phát triển đặt mình vào vị trí của người dùng hoặc khách hàng, đồng thời cho phép nhóm phát triển thực hành kiểm soát quy trình theo kinh nghiệm bằng cách sử dụng quan sát, trò chuyện và thử nghiệm làm cách để đưa ra quyết định.

Lưu ý: *Trong mô hình Feature-Driven Development (FDD), không có user story như trong một số phương pháp Agile. Lý do chính là FDD tập trung vào việc tổ chức và quản lý phát triển dựa trên các tính năng (features) của sản phẩm, thay vì sử dụng user stories.*

2.2.2.2. Cấu trúc user story

Cấu trúc của một user story thường tuân theo một định dạng đơn giản:

As a <**type of user**>, I want/need <**some goal**> so that I can <**reason**>.

Trong đó:

- **type of user**: Cho biết người dùng cuối sẽ là ai hay có thể hiểu tính năng này dành cho loại người dùng nào.

- **some goal**: mục tiêu của chức năng của người dùng cuối là gì? Mô tả 'cái gì' chứ không phải 'như thế nào'.

- **reason**: Lý do cần tính năng này là gì?

Cấu trúc này giúp định rõ ai là người sử dụng, tính năng cần phát triển là gì, và lý do tại sao tính năng đó lại quan trọng. Đây là cách đơn giản và hiệu quả để truyền đạt các yêu cầu từ góc nhìn của người dùng cuối.

* Ví dụ:

- *Là một khách hàng, tôi muốn có khả năng đặt món ăn từ nhà hàng yêu thích của mình thông qua ứng dụng di động, để tiết kiệm thời gian và thuận tiện khi đặt hàng.*
- *Là khách hàng và bên liên quan chính, tôi cần có thể đăng nhập từ mọi nơi trên thế giới để cải thiện quan hệ và dịch vụ khách hàng tốt hơn.*
- *Là người dùng cuối, tôi cần một ứng dụng để theo dõi con mình để có thể nhận được thông tin cập nhật về nơi ở của con và có thể giữ an toàn cho con".*
- *Là người ra quyết định trong tổ chức của mình, tôi cần một ứng dụng có thể truy cập vào kết quả hàng ngày của nhân viên bán hàng của tôi, bất kể họ ở đâu trên thế giới, để tôi nhận được thông tin cập nhật về doanh số một cách kịp thời."*

Mặc dù các ví dụ trên có thể được coi là mơ hồ và không cụ thể, nhưng chúng có thể được xem xét kỹ lưỡng hơn khi quá trình tạo câu chuyện của người dùng tiến triển. Hãy nhớ ngay rằng, chúng ta đang cố gắng thu hút các bên liên quan, cố gắng hiểu nhu cầu của họ và lý do họ muốn sản phẩm, dịch vụ hoặc kết quả đó.

Mục tiêu của bất kỳ câu chuyện nào của người dùng là đảm bảo rằng nhóm có thể chia thành một khối lượng công việc cụ thể (khoảng 1 đến 3 ngày làm việc) và kiểm tra mức độ hoàn thành của nó.

Câu chuyện của người dùng cung cấp cho nhóm(nhóm phát triển, chủ sở hữu sản phẩm, người quản lý dự án Agile) các thông tin chi tiết và cách để xem các bên liên quan và người dùng cuối là những người cần gì. Điều này sẽ tạo ra sự đồng tình từ phía nhóm và khách hàng sẽ cảm thấy được lắng nghe và thấu hiểu.

Câu chuyện của người dùng là một định nghĩa cấp cao, chứa vừa đủ thông tin để nhóm phát triển có thể đưa ra ước tính hợp lý về nỗ lực thực hiện công việc thực tế. Câu chuyện của người dùng thường được viết trên thẻ mục lục hoặc ghi chú dán, được lưu trữ và sắp xếp trên bảng hoặc bất cứ nơi nào có không gian để tạo điều kiện lập kế hoạch và thảo luận thêm. Câu chuyện của người dùng chuyển trọng tâm từ việc viết các bài viết về các tính năng sang thảo luận về chúng.

Lưu ý: *Câu chuyện của người dùng không phải là một yêu cầu. Nó được định nghĩa như một lời nhắc nhở để có một cuộc trò chuyện và những cuộc thảo luận này thường quan trọng hơn bất cứ điều gì được viết ra. Chính những cuộc đối thoại này đã khơi dậy những điểm suy nghĩ quan trọng nhất về yêu cầu.*

Nhóm phát triển viết User story trên một tấm thiệp (Card). Điều này làm cho câu chuyện của người dùng trở nên hữu hình và độc đáo, đồng thời nó cũng giới hạn nội dung trên đó. Trong một lần lặp/dự án có thể có nhiều câu chuyện của người dùng. Chủ sở hữu sản phẩm có quyền quyết định và phân loại thẻ hoặc câu chuyện nào của người dùng có giá trị cao hơn so với câu chuyện khác. Giống như cách tiếp cận thẻ của Kanban hoặc bảng ghi chú Post-it của Scrum để mô tả các câu chuyện hoặc nhiệm vụ của người dùng.

2.2.2.3. Xác định tiêu chí chấp nhận

Làm thế nào để nhóm phát triển biết khi nào nhóm đã hoàn thành? Đây là một câu hỏi quan trọng và là câu hỏi đầu tiên nên được hỏi khi bắt đầu cuộc trò chuyện do câu chuyện của người dùng gợi ý. Để làm việc hiệu quả, điều quan trọng là phải biết:

- Khi nào nên dừng lại?
- Ranh giới của công việc là gì?
- Nhóm phát triển phải làm bao nhiêu để nó được chấp nhận?

- Làm thế nào để nhóm có thể tránh được việc quá mức?

Tiêu chí chấp nhận (Acceptance Criteria) là một phần quan trọng của user story trong Agile. Nó định rõ các điều kiện mà một tính năng hoặc chức năng phải đáp ứng để được coi là hoàn thành và chấp nhận được. Các tiêu chí này:

- Định nghĩa rõ ràng: Giúp làm rõ các yêu cầu và mong đợi từ phía người dùng hoặc khách hàng.
- Giảm thiểu nhầm lẫn: Đảm bảo rằng tất cả các bên liên quan (bao gồm cả nhóm phát triển và người dùng) đều có cùng một hiểu biết về yêu cầu.
- Cơ sở kiểm thử: Cung cấp các điều kiện cụ thể để nhóm kiểm thử có thể dựa vào đó để xác nhận tính năng hoạt động đúng như yêu cầu.
- Định hướng phát triển: Hướng dẫn các nhà phát triển trong việc xây dựng tính năng, đảm bảo rằng họ biết rõ những gì cần đạt được.
- Xác nhận hoàn thành: Giúp xác định khi nào một user story có thể được coi là hoàn thành và sẵn sàng để giao cho người dùng hoặc khách hàng.

Để một câu chuyện của người dùng thực sự được thực hiện, nó cần có các tiêu chí chấp nhận – những điều kiện tiên quyết phải được đáp ứng để một câu chuyện được đánh giá là hoàn chỉnh.

Tiêu chí chấp nhận có thể có nhiều dạng, từ những điều kiện hài lòng đơn giản cho đến những kiểm tra nghiêm ngặt và rất chính xác. Với mục đích trở nên linh hoạt, các câu lệnh đơn giản dạng nhị phân được viết bằng ngôn ngữ đơn giản là tốt nhất để bắt đầu.

Ví dụ. Xác định tiêu chí chấp nhận về ngày giao hàng của đơn hàng online:

- Ngày giao hàng luôn là ngày làm việc tiếp theo của ngày đặt hàng.
- Ngày giao hàng sẽ là thứ Hai nếu đơn hàng được đặt vào thứ Bảy.
- Thời gian chốt ngày giao hàng cho các đơn hàng sẽ là 3 giờ chiều.
- Sẽ có email xác nhận ngày giao hàng.

- Tất cả các dòng sản phẩm đều có quy định về ngày giao hàng giống nhau.

- Người dùng có thể để lại lời nhắn cho nhân viên giao hàng.

- Ngày giao hàng dự kiến sẽ được hiển thị trên màn hình khi đặt hàng.

Tiêu chí chấp nhận được giao cho nhóm cộng tác viết là cách khả thi nhất để bao quát mọi góc độ. Được dẫn dắt bởi chủ sở hữu sản phẩm (Product Owner) hoặc đại diện doanh nghiệp, nhóm có thể thảo luận về các câu chuyện, loại bỏ sự mơ hồ và xác định kết cục của trò chơi. Điều này là cách tốt nhất để mang lại sự gắn kết trong nhóm và chúng không cần kéo dài hay tốn nhiều công sức. Tiêu chí chấp nhận cung cấp đơn vị đo hữu ích để báo cáo tiến độ.

2.2.2.4. Phân rã user story

Đôi khi, quá nhiều cuộc trò chuyện hay sẽ tạo ra nhiều tài liệu, gây khó khăn cho nhóm phát triển. Tuy nhiên, đây là một điều tốt, giúp nhóm phát triển nắm bắt tất cả.

Giữ câu chuyện của người dùng ở có độ lớn vừa phải là điều quan trọng. Câu chuyện càng phức tạp thì càng có nhiều nguy cơ xảy ra sai sót. Với một câu chuyện người dùng có quá nhiều tiêu chí chấp nhận, khi này, nhóm phát triển cần chia nhỏ câu chuyện người dùng với ít tiêu chí chấp nhận hơn. Việc chia nhỏ hay phân rã có thể cần được thực hiện nhiều lần có được gói công việc hợp lý hơn.

Khi phân rã User story, Agile sử dụng nguyên lý MVP. MVP (Minimum Viable Product - Sản phẩm tối thiểu). MPV là phiên bản đầu tiên mà nhóm phát triển tạo ra để thu thập phản hồi từ người dùng và xác định hướng phát triển tiếp theo. Điều này giúp đảm bảo rằng sản phẩm cuối cùng được phát triển phù hợp nhất với nhu cầu và mong đợi của người dùng (thỏa mãn nhu cầu cơ bản của người dùng). MVP và một số câu chuyện người dùng được viết hay hoàn chỉnh

với các tiêu chí chấp nhận. Theo phương pháp ước lượng truyền thống, các nhà quản lý dự án ước tính thực hiện nhiệm vụ này và sau đó đưa ra dự đoán của họ. Nhưng trong một dự án linh hoạt, nhóm, những người thực sự thực hiện công việc sẽ đưa ra các ước tính. Điều này giúp dự đoán đáng tin cậy hơn, còn có lợi thế là thu hút được sự tham gia của nhóm.

2.2.2.5. Story point

Story Points là đơn vị đo lường được sử dụng để ước tính độ phức tạp và công sức cần thiết để hoàn thành một user story. Chúng không phải là đơn vị thời gian mà thay vào đó là sự kết hợp của nhiều yếu tố, bao gồm:

- Độ phức tạp: Mức độ khó khăn của User Story.
- Khối lượng công việc: Tổng lượng công việc cần thiết.
- Rủi ro: Các yếu tố không chắc chắn và rủi ro liên quan.

Story points có vai trò quan trọng trong việc lập kế hoạch sprint, giúp đội phát triển xác định lượng công việc phù hợp cho mỗi sprint dựa trên khả năng (velocity) của đội. Story points còn cho phép so sánh độ phức tạp giữa các user story khác nhau.

Một số kỹ thuật ước tính điểm (Story point) phù hợp với các dự án Agile:

- Sử dụng phương pháp ước lượng size: Gán các thẻ S, M, L, XL và XXL cho mọi thứ và đánh giá đại khái mức độ công việc mà mỗi kích thước liên quan. Phương pháp này dễ sử dụng và là một khởi đầu tốt, tuy nhiên nó có thể thiếu độ chính xác.

- Phương pháp Planning Poker: Planning Poker là một kỹ thuật phổ biến trong Agile để ước lượng điểm cho các user story (Story Points). Đây là một cách thức hợp tác, giúp đội phát triển đạt được sự đồng thuận về độ phức tạp và khối lượng công việc của mỗi user story. Lợi ích của Planning Poker: (1) Khuyến khích thảo luận và đồng thuận: Planning Poker giúp các thành viên trong đội chia sẻ quan điểm và đạt được sự đồng thuận về ước lượng. (2) Giảm

bớt ảnh hưởng của cá nhân: Bằng cách tiết lộ đồng thời, không ai bị ảnh hưởng bởi ước lượng của người khác. (3) Tăng độ chính xác: Qua thảo luận và đồng thuận, ước lượng sẽ chính xác hơn so với việc ước lượng cá nhân. Dưới đây là các bước chi tiết để sử dụng Planning Poker:

+ Chuẩn bị: chọn bộ thẻ Planning Poker: Mỗi thành viên trong đội cần có một bộ thẻ Planning Poker. Mỗi bộ thẻ thường có các giá trị 0, 1, 2, 3, 5, 8, 13, 20, 40 và 100. Đây là chuỗi số Fibonacci, được chọn vì sự tăng dần giúp phân biệt rõ ràng giữa các mức độ phức tạp khác nhau.
+ Chuẩn bị User Stories: Đảm bảo rằng tất cả các user stories cần ước lượng đã được viết rõ ràng và có đầy đủ thông tin.
+ Quy trình Planning Poker:
    - ✓ Giải thích User Story: Product Owner (PO) hoặc người dẫn dắt cuộc họp sẽ đọc và giải thích từng user story, cung cấp bất kỳ chi tiết hoặc ngữ cảnh bổ sung nào cần thiết. Đội phát triển có thể đặt câu hỏi để làm rõ mọi khía cạnh của user story.
    - ✓ Thảo luận ngắn: Đội phát triển thảo luận ngắn gọn về user story, đề cập đến các yếu tố như độ phức tạp, rủi ro, khối lượng công việc và các phụ thuộc liên quan.
    - ✓ Chọn thẻ: Mỗi thành viên trong đội chọn một thẻ từ bộ thẻ Planning Poker của mình, thể hiện ước lượng điểm cho user story dựa trên hiểu biết cá nhân của họ về yêu cầu.
    - ✓ Tiết lộ thẻ: Tất cả các thành viên đồng thời tiết lộ thẻ của mình để không bị ảnh hưởng bởi ước lượng của người khác.
    - ✓ Thảo luận về ước lượng: Nếu tất cả các ước lượng giống nhau, giá trị đó được chấp nhận. Nếu có sự khác biệt, các thành viên với ước lượng cao nhất và thấp nhất giải thích lý do của mình. Điều này giúp đội hiểu rõ hơn về các quan điểm khác nhau.

- ✓ Lặp lại nếu cần Thiết: Sau thảo luận, đội lại chọn thẻ và tiết lộ cùng một lúc. Quá trình này lặp lại cho đến khi đạt được sự đồng thuận hoặc ít nhất là sự đồng thuận tương đối về điểm số.
- ✓ Ghi lại điểm: Điểm số được chốt và ghi lại cho user story đó.

Thang điểm story point có thể được tính như sau:

- ➢ 1 điểm: Rất đơn giản, ít rủi ro, có thể hoàn thành nhanh chóng.
- ➢ 2-3 điểm: Trung bình, cần nỗ lực nhưng không quá phức tạp.
- ➢ 5-8 điểm: Phức tạp, nhiều bước thực hiện, có thể gặp rủi ro.
- ➢ 13 điểm trở lên: Rất phức tạp, nhiều rủi ro, có thể cần chia nhỏ.

2.2.2.6. Estimated Effort

Effort là số giờ làm việc cần thiết để hoàn thành một user story trong các dự án phát triển phần mềm theo Agile. Effort giúp đội phát triển dự đoán và phân bổ thời gian, nguồn lực hiệu quả cho từng phần của dự án. Đơn vị thời gian của Effort là giờ (Hour).

Mục tiêu của Effort:

- Lập kế hoạch sprint: Giúp đội phát triển xác định số lượng User Story có thể hoàn thành trong một Sprint.
- Theo dõi tiến độ: Giúp quản lý và các bên liên quan theo dõi tiến độ của dự án và điều chỉnh kế hoạch nếu cần thiết.
- Quản lý kỳ vọng: Giúp thiết lập kỳ vọng rõ ràng về thời gian hoàn thành và độ phức tạp của từng User Story.
- Phân bổ nguồn lực: Giúp đội phát triển phân bổ nguồn lực một cách hợp lý, tránh tình trạng quá tải hoặc thiếu việc.

Cách kỹ thuật ước tính Effort:

- Planning Poker: Sử dụng bộ bài có giá trị số giờ để đánh giá Effort cho mỗi user story thông qua thảo luận và đồng thuận.

- T-Shirt Sizes: Nhóm các User Story theo kích thước (XS, S, M, L, XL) và gán số giờ tương ứng với từng kích thước.
- Historical Data: Sử dụng dữ liệu từ các dự án trước để ước tính effort cho các User Story tương tự.
- Expert Judgment: Dựa vào kinh nghiệm và kiến thức của các chuyên gia trong đội để ước tính effort.
- Three-Point Estimation: Sử dụng ba điểm dữ liệu (ước tính lạc quan, bi quan và trung bình) để tính toán effort.

$$E = \frac{\mathrm{O} + 4\mathrm{M} + \mathrm{P}}{6}$$

với: O (Optimistic): Thời gian ngắn nhất có thể hoàn thành công việc, giả định rằng mọi thứ đều thuận lợi; M (Most Likely): Thời gian trung bình có thể hoàn thành công việc, giả định rằng mọi thứ diễn ra bình thường; P (Pessimistic): Thời gian dài nhất có thể hoàn thành công việc, giả định rằng có nhiều trở ngại và khó khăn.

Với Three-Point Estimation, sử dụng công thức sau để tính độ biến động (σ - Standard Deviation):

$$\sigma = \frac{P - O}{6}$$

2.2.2.7. Xác định mức độ ưu tiên của user story theo MoSCoW

MoSCoW là một kỹ thuật phổ biến được sử dụng trong quản lý dự án và phát triển phần mềm để ưu tiên các yêu cầu, bao gồm cả user story trong các dự án Agile. MoSCoW là viết tắt của bốn loại mức độ ưu tiên: Must have, Should have, Could have, và Won't have. Đây là cách tiếp cận giúp các nhóm phát triển và các bên liên quan xác định những yêu cầu nào là thiết yếu và những yêu cầu nào có thể được thực hiện sau hoặc thậm chí không cần thực hiện.

- **Must have**. Đây là các user story/yêu cầu bắt buộc phải có cho sản phẩm hoặc hệ thống. Nếu thiếu bất kỳ yêu cầu nào trong nhóm này, dự án có thể bị coi là thất bại.
- **Should have**. Đây là các user story/yêu cầu quan trọng nhưng không bắt buộc. Những yêu cầu này có ảnh hưởng lớn đến thành công của dự án nhưng hệ thống vẫn có thể hoạt động mà không có chúng.
- **Could have.** Đây là các user story/yêu cầu không quá quan trọng và chỉ nên được thực hiện nếu còn đủ thời gian và nguồn lực sau khi hoàn thành các yêu cầu "must have" và "should have".
- **Won't have.** Đây là các yêu cầu sẽ không được thực hiện trong khoảng thời gian hiện tại nhưng có thể được xem xét cho tương lai. Các yêu cầu này không ảnh hưởng đến thành công của dự án trong giai đoạn hiện tại.

Các lợi ích của phương pháp MoSCoW

- Rõ ràng trong ưu tiên: Phương pháp này giúp các nhóm dự án và các bên liên quan hiểu rõ yêu cầu nào là quan trọng nhất và cần phải được thực hiện trước.
- Quản lý thời gian và nguồn lực hiệu quả: Bằng cách tập trung vào các yêu cầu "must have" và "should have" trước, nhóm dự án có thể đảm bảo rằng các nguồn lực được sử dụng một cách hiệu quả nhất.
- Dễ dàng thỏa thuận: Phương pháp MoSCoW cung cấp một ngôn ngữ chung cho các nhóm để thảo luận về mức độ ưu tiên của các yêu cầu, giúp dễ dàng đạt được sự đồng thuận.
- Giảm rủi ro: Bằng cách xác định rõ các yêu cầu thiết yếu, phương pháp này giúp giảm rủi ro liên quan đến việc thiếu các chức năng quan trọng khi dự án hoàn thành.

Cách áp dụng MoSCoW:

- Xác định yêu cầu: Bắt đầu bằng việc thu thập tất cả các yêu cầu hoặc user story từ các bên liên quan.
- Phân loại yêu cầu: Sử dụng phương pháp MoSCoW để phân loại mỗi yêu cầu vào một trong bốn nhóm: Must have, Should have, Could have, Won't have.
- Xem xét và đồng thuận: Thảo luận với các bên liên quan để đảm bảo rằng tất cả mọi người đều đồng ý với việc phân loại và mức độ ưu tiên đã được xác định.
- Lập kế hoạch và triển khai: Sử dụng danh sách ưu tiên này để lập kế hoạch các sprint hoặc các đợt phát triển, bắt đầu với các yêu cầu "must have".

Sử dụng phương pháp Planning Poker để ước tính giá trị cho MoSCoW.

2.2.2.8. Nguyên tắc INVEST trong phát triển User story

INVEST là một công cụ giúp định hình và phát triển User story. INVEST được hình thành các thuộc tính sau:

- **I**ndependent (Độc lập): Khi viết câu chuyện của người dùng, cần thiết giữ chúng độc lập với nhau. Điều này có nghĩa là mỗi câu chuyện của người dùng không được tạo theo thứ tự xuất hiện.

- **N**egotiable (Có thể thương lượng): Mọi câu chuyện của người dùng cho dù đã viết trên thẻ không có nghĩa là nó chắc chắn sẽ được thực hiện. Nó có thể được thay đổi sau khi có những cuộc thương lượng.

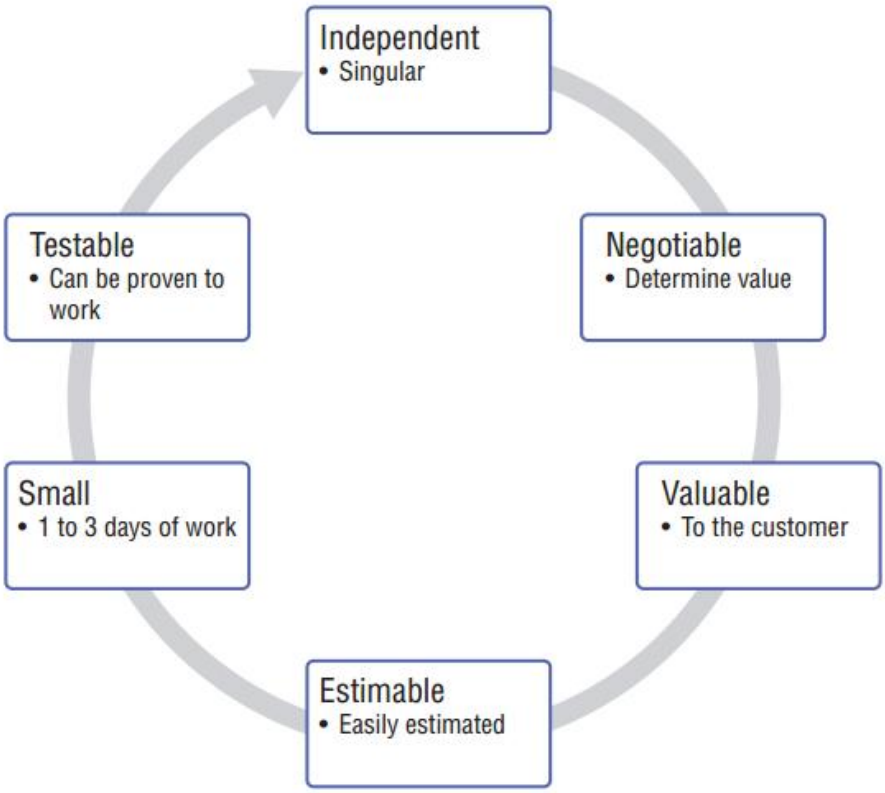
- **V**aluable (Có giá trị): Câu chuyện của người dùng có thể có giá trị đối với người dùng nhưng không có giá trị đối với khách hàng và ngược lại. Mục tiêu là cung cấp đủ chi tiết trong user story sao cho có thể mô tả tính năng cũng như lợi ích phù hợp và có giá trị đối với khách hàng, người dùng hoặc cả hai.

- **E**stimable (Có thể ước tính/ước lượng): cần xác định được số lượng công việc có thể hoàn thành dựa trên câu chuyện của người dùng. Hay nói cách khác, câu chuyện của người dùng phải thể hiện được công việc cần phải thực hiện.

- **S**mall (Nhỏ): Tất cả các câu chuyện của người dùng cần phải đủ nhỏ để có thể hoàn thành tối đa trong khoảng tối đa 3 ngày làm việc. Điều quan trọng là phải giữ các câu chuyện của người dùng đủ nhỏ để có thể dễ dàng trả lời tất cả các câu hỏi về câu chuyện và hiểu được những gì cần hoàn thành. Nếu có quá nhiều câu hỏi được đặt ra và không thể trả lời nếu không có sự rõ ràng hơn, đó có thể là do câu chuyện quá lớn hoặc bao gồm nhiều đặc điểm liên quan khác nhau. Cần thiết giữ câu chuyện người dùng đủ nhỏ để nhóm có thể xác định quy mô của công việc, định nghĩa của công việc đã hoàn thành là gì và có thể tạo ra mức tăng trưởng khả thi ở mức tối thiểu trong quá trình lặp lại. Giữ những câu chuyện nhỏ và đơn giản giúp duy trì sự tập trung vào mục tiêu và cho phép nhóm thực hiện công việc một cách thích hợp.

- **T**estable (Có thể kiểm tra được): Kiểm tra là phương pháp và là bằng chứng cho thấy điều gì đó đúng, nó hoạt động và đáp ứng yêu cầu. Khi bài kiểm thử được thông qua, có nghĩa là nó đáp ứng yêu cầu.

Hình 2.4. minh họa về các hoạt động trong việc viết user story với INVEST.



Hình 2.4. Tiến trình phát triển user story

2.2.2.9. Viết kịch bản cho User story

Trong Agile, một "ngôn ngữ kịch bản" (scripting language) để phát triển user story thường liên quan đến việc tạo các kịch bản chi tiết và có cấu trúc để mô tả cách hệ thống sẽ được sử dụng. Một ví dụ điển hình là Gherkin, được sử dụng trong kỹ thuật BDD (Behavior Driven Development).

BDD sử dụng các từ khóa như Feature, Scenario, Given, When, Then để mô tả các yêu cầu và kịch bản sử dụng. Đây là một ngôn ngữ cụ thể dùng để viết các test case có cấu trúc và dễ hiểu, cả cho người không có kỹ thuật lẫn các nhà phát triển.

Cấu trúc BDD:

```
Feature: [Tính năng]
Scenario: [Kịch bản]
   Given [Bối cảnh]
```

When [Hành động]

Then [Kết quả]

Scenario: [Kịch bản khác]

Given [Bối cảnh khác]

When [Hành động khác]

Then [Kết quả khác]

2.2.3. Theme

Theme là một khái niệm trừu tượng mô tả một tập hợp các mục tiêu, ý định, hoặc giá trị cốt lõi mà dự án hoặc sản phẩm cần tập trung. Nó không chỉ đơn thuần là một mục tiêu cụ thể, mà thường là một xu hướng hoặc một khía cạnh rộng lớn của dự án. Một theme có thể giúp định hình và cung cấp hướng cho dự án, làm cho mọi người trong nhóm hiểu rõ mục tiêu chung của họ và làm việc hướng tới cùng một mục tiêu.

*** Ví dụ:** Trong một ứng dụng giao diện người dùng (UI) và trải nghiệm người dùng (UX), có thể xác định một Theme như sau:

"Tối ưu hóa trải nghiệm người dùng"

Mục tiêu của theme này là cải thiện trải nghiệm của người dùng khi sử dụng ứng dụng.

Viết mục tiêu cụ thể từ Theme:

Có thể xây dựng một số mục tiêu cụ thể có thể nằm trong phạm vi của theme "Tối ưu hóa trải nghiệm người dùng":

(1) Tăng tính tương tác: Tạo ra một giao diện người dùng thân thiện và dễ sử dụng để tăng cường tương tác của người dùng với ứng dụng.

(2) Tối ưu hóa hiệu suất: Giảm thiểu thời gian tải và tăng tốc độ phản hồi của ứng dụng để cải thiện trải nghiệm người dùng.

(3) Tăng cường khả năng đáp ứng: Đảm bảo ứng dụng có thể hiển thị đúng trên mọi loại thiết bị và mọi kích thước màn hình, từ điện thoại di động đến máy tính bảng và máy tính để bàn.

Theme "Tối ưu hóa trải nghiệm người dùng" giúp định hình và định hướng cho dự án UI/UX, cho phép nhóm tập trung vào cải thiện và tối ưu hóa trải nghiệm người dùng một cách toàn diện.

Sau khi xác định được theme, tiếp theo chia theme thành các epics và cuối cùng là chuyển user stories cụ thể:

(1) Thêm điều kiện chấp nhận

(2) Ước lượng và ưu tiên user stories

(3) Tạo bản đồ câu chuyện

2.2.4. Epic

Có thể coi Epic là một user story lớn, thời gian hoàn thành thường vượt qua khía cạnh của một Sprint. Nhóm dự án có thể phân rã Epic thành các user story.

- Ví dụ về Epic:

Với tư cách là ***người quản lý tiếp thị****, tôi cần* ***phát triển một chiến dịch web tương tác*** *để* ***thúc đẩy doanh số bán hàng để khách hàng biết đến các sản phẩm mới.***

Cả Epic và User Story đều dùng để mô tả yêu cầu và tính năng của sản phẩm. Epic là một tập hợp lớn các yêu cầu hoặc tính năng có phạm vi rộng lớn và chưa được phân chia hoàn toàn thành các phần nhỏ hơn. Epic thường đại diện cho các mục tiêu lớn và dài hạn của dự án, không được phân rã chi tiết và cụ thể.

- Ví dụ so sánh sự khác biệt giữa Epic và User story:

- Epic: *Tính năng thanh toán trực tuyến cho một ứng dụng thương mại điện tử. Nó bao gồm nhiều yêu cầu như quản lý giỏ hàng, xử lý thanh toán, và cung cấp hóa đơn.*
- User Story: *Người dùng muốn thêm sản phẩm vào giỏ hàng và thanh toán nhanh chóng mà không cần tài khoản.*

Lưu ý: *Epic khác với Theme, vì hầu hết các user story trong Epic đều liên quan đến nghiệp vụ cụ thể và chúng phải được chia nhỏ và thực hiện theo thứ tự. Những user story thuộc Theme có thể được thực hiện độc lập mà không theo bất kỳ thứ tự nào. Có thể phải lặp đi lặp lại nhiều lần để hoàn thành các user story trong Epic, nhưng cho đến khi tất cả các user story được hoàn thành thì giá trị nghiệp vụ mới có thể đạt được.*

Mỗi Theme trong Agile về cơ bản chỉ là một tập hợp các câu chuyện hoặc yêu cầu của người dùng. Có thể hiểu, chủ đề là nhóm các "yêu cầu" có thể có cùng chức năng với các phần tử khác nhau. Cũng cần lưu ý, chủ đề thường có thể quá lớn để tạo ra các tính năng cụ thể trong một Sprint.

2.2.5. Story map

Story mapping là một kỹ thuật giúp tổ chức và hình dung các user story trong quá trình phát triển sản phẩm. Nó giúp nhóm phát triển hiểu rõ hơn về thứ tự thực hiện và mức độ ưu tiên của các công việc.

Việc tạo Story Map thường được thực hiện trước khi viết User Story. Điều này giúp tạo ra một cái nhìn tổng quan về sản phẩm và các hoạt động chính mà người dùng sẽ thực hiện. Sau khi có Story Map, có thể chi tiết hóa và viết các User Story dựa trên những gì đã được xác định trong Story Map.

Các bước xây dựng một story map:

(1) Xác định mục tiêu sản phẩm: Đầu tiên, cần hiểu rõ mục tiêu chính của sản phẩm hoặc tính năng mà bạn đang phát triển.

(2) Xác định các hoạt động chính (Activities): Xác định các hoạt động chính mà người dùng sẽ thực hiện để đạt được mục tiêu của sản phẩm.

(3) Phân chia hoạt động thành các bước cụ thể (Steps): Mỗi hoạt động chính được phân chia thành các bước cụ thể mà người dùng sẽ thực hiện.

(4) Tạo user story từ các bước cụ thể: Viết các user stories cho từng bước cụ thể.

(5) Sắp xếp các user stories theo thứ tự thực hiện (Backbone): Sắp xếp các user stories theo thứ tự thực hiện, từ trái sang phải, tạo thành một cột sống (backbone) của story map.

2.2.6. Phát triển user story

2.2.6.1. User story workshop

User Story Workshop (USW) là một hoạt động trong phương pháp Agile được thực hiện để thu thập yêu cầu từ khách hàng hoặc người dùng cuối và chuyển chúng thành các câu chuyện của người dùng có ý nghĩa và thực tế để sử dụng trong quá trình phát triển phần mềm. Vai trò của USW:

- **Thu thập yêu cầu từ người dùng:** Trong USW, các thành viên của nhóm phát triển, bao gồm các nhà quản lý dự án, người sử dụng, và các chuyên gia liên quan, hợp tác để thu thập yêu cầu từ người dùng cuối.
- **Xác định các tình huống sử dụng:** Workshop giúp xác định và mô tả các tình huống sử dụng cụ thể mà người dùng mong đợi sản phẩm hoặc dịch vụ sẽ đáp ứng. Điều này giúp nhóm hiểu rõ hơn về ngữ cảnh và yêu cầu của người dùng.
- **Tạo ra user stories**: Dựa trên các yêu cầu và tình huống sử dụng, Workshop giúp nhóm tạo ra các user stories cụ thể và có ý nghĩa. Mỗi

user story mô tả một tính năng hoặc một khía cạnh của sản phẩm từ góc nhìn của người dùng.

- **Phân tích và phân nhóm ưu tiên**: Workshop cũng có thể bao gồm việc phân tích và ưu tiên hóa các user story để xác định những tính năng quan trọng nhất và cần thiết nhất để triển khai trong các chu kỳ phát triển sau này.
- **Tạo ra sự đồng thuận**: Bằng cách tham gia vào Workshop, các thành viên của nhóm đồng ý về các yêu cầu và mục tiêu của dự án. Điều này giúp tạo ra sự đồng thuận và sự hiểu biết chung về hướng phát triển của sản phẩm.

User story workshop là cơ hội để mọi người cùng làm việc và giao tiếp với nhau. Đây là một cách hợp tác để đạt được cái nhìn sâu sắc có giá trị về nhu cầu của người dùng cuối và các yêu cầu là gì, đồng thời nó cũng thu hút các bên liên quan trong quá trình thiết kế. Và cũng là chìa khóa cho hầu hết việc lập kế hoạch trong môi trường Agile. Thời gian cho mỗi user story workshop có thể kéo dài từ một đến hai giờ.

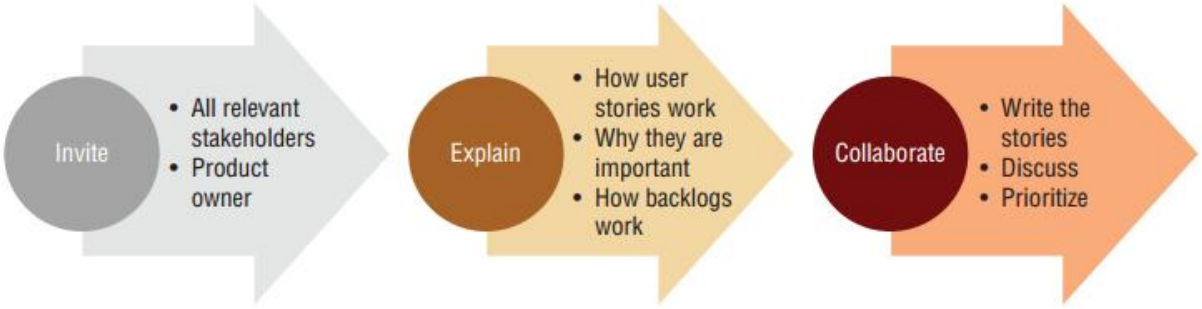
Các nội dung hoạt động trong USW:

- **Hội thoại (conversation)**. Các cuộc hội thoại với khách hàng/người dùng cuối giúp xác định được các câu chuyện người dùng. Giao tiếp, tương tác và trò chuyện là điểm mấu chốt của Agile. Không có nó, sẽ không bao giờ xây dựng được thứ đúng đắn.

- Sự xác nhận (confimation). Chúng ta đã làm đúng chưa? Chúng ta đã xây dựng đúng thứ chưa? Nó có dùng được không? Chúng ta đã thử nghiệm nó để đảm bảo chưa? Chúng ta có thể kiểm tra nó để đảm bảo không? Khách hàng cần phê duyệt kết quả và xác thực việc hoàn thành chính xác của nó. Đánh giá của khách hàng cũng sẽ tạo cơ hội để giới thiệu phần tăng trưởng khi kết thúc

một lần chạy nước rút hoặc lặp lại. Điều này cho phép khách hàng xác nhận rằng nó đúng hay không bằng cách sử dụng thử nghiệm chấp nhận.

Hình 2.3 cho thấy các bước trong tiến trình hoạt động của US workshop.



Hình 2.3. Tiến trình user story workshop

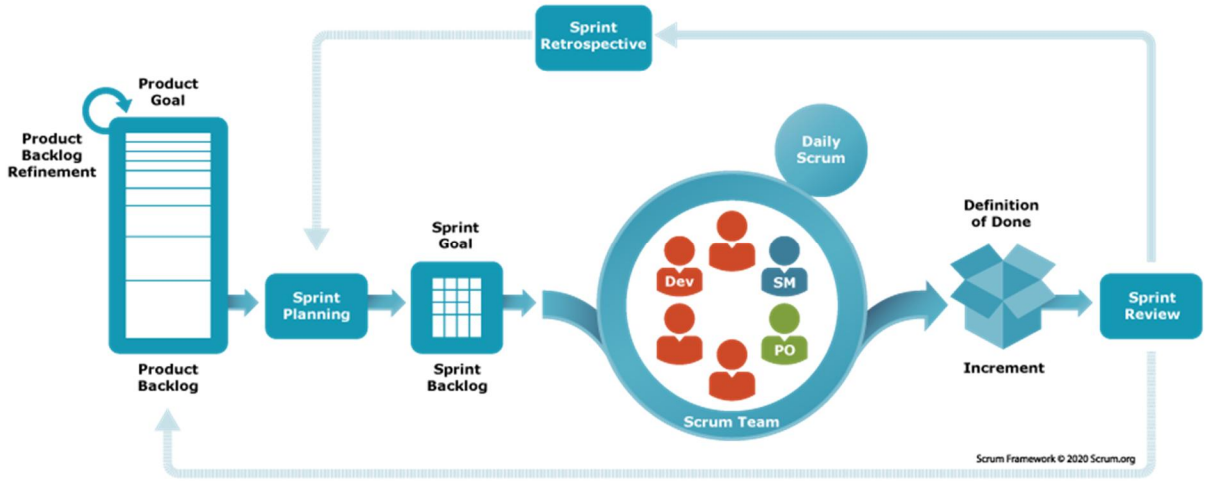
2.2.6.2. Vai trò của Project Manager, Development Team và Product Owner

PO (Product Owner - Chủ sở hữu sản phẩm) thường xuyên liên lạc với các bên liên quan và nhóm phát triển để thúc đẩy tầm nhìn của dự án, thu thập thông tin và làm việc để tạo ra những câu chuyện cần thiết của người dùng nhằm giải quyết vấn đề và tạo ra giá trị của sản phẩm dự án. Các bên liên quan có thể có nhiều ý tưởng về những thứ họ muốn trong quá trình tăng trưởng và chủ sở hữu sản phẩm sẽ cần biến những yêu cầu đó thành câu chuyện của người dùng. Sau đó, PO sẽ đưa ra các ước tính và ưu tiên các câu chuyện dựa trên giá trị và có thể mô tả kết quả cho phép nhóm hoàn thành chúng. PO cũng cần phải xem xét với các ý tưởng và danh sách mong muốn trở nên quá dài hoặc đi theo hướng không hỗ trợ cho tầm nhìn tổng thể của dự án.

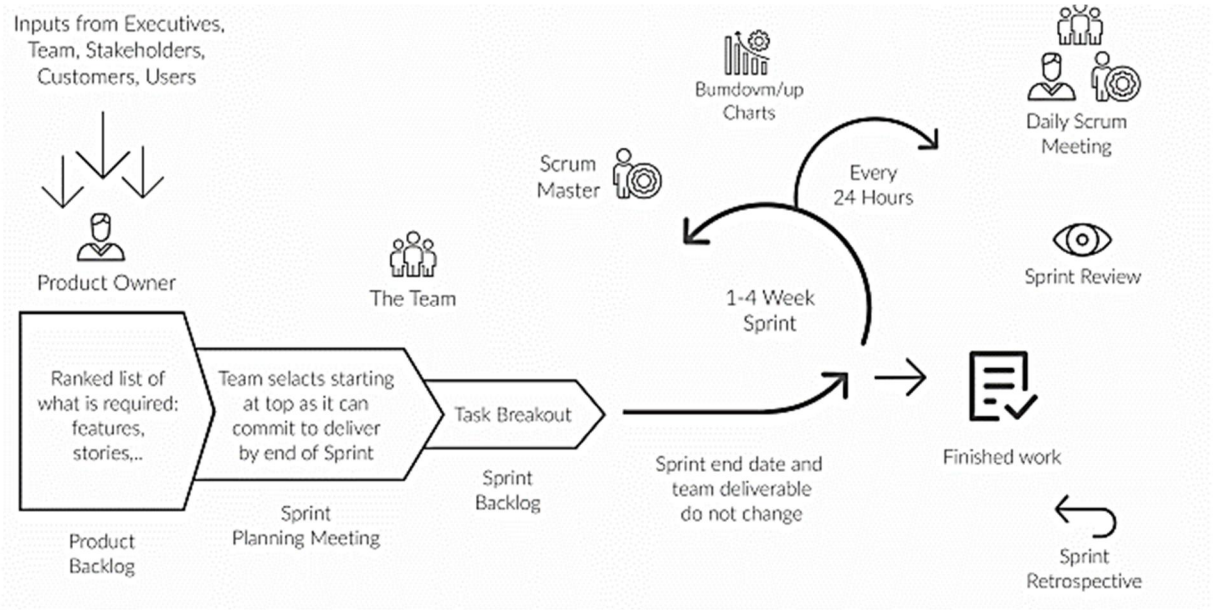
Kinh nghiệm của Agile cho thấy, mỗi **DT (Development Team – Nhóm phát triển)** lý tưởng thường có số thành viên không quá chín người. Chín người đó sẽ làm việc theo mức tăng trưởng mà PO chỉ định. DT sẽ xem xét và xác định xem họ có thể hoàn thành các câu chuyện của người dùng như thế nào trong mỗi lần lặp lại. Điều này hoàn toàn phụ thuộc vào nhóm, dựa trên những gì họ nghĩ họ có thể hoàn thành và những gì họ xác định là năng lực của mình. Lưu ý rằng chính nhóm là người quyết định điều này chứ không phải người quản lý dự án

(PM - Project Manager) Agile, PO hoặc các bên liên quan khác. Điều đó có nghĩa là DT đã chọn số lượng công việc mà họ có thể hoàn thành chỉ trong lần lặp giúp PO đánh giá và ưu tiên cho các user story cho mỗi lần lặp. Tất cả các ước tính đều được xem xét khi DT xác định số lượng câu chuyện hoặc công việc họ có thể hoàn thành trong một lần lặp. DT cũng muốn giới hạn công việc đang tiến hành (WIP- Work In Progress) hoặc đặt giới hạn WIP để không đưa quá nhiều công việc vào cùng một lúc, điều này có thể tạo ra tình trạng tràn công việc ở lần lặp tiếp theo.

Công việc của người quản lý dự án Agile có nhiệm vụ giúp PO sắp xếp các tính năng và chức năng chính ở đầu danh sách, giúp DT xác định công việc nào họ có thể hoàn thành và duy trì tầm nhìn của dự án. PM giúp tạo điều kiện thuận lợi cho các sự kiện trong Sprint góp phần mang lại kết quả hiệu quả cũng như cung cấp thông tin liên lạc minh bạch cho các bên liên quan về các quy trình đang được sử dụng trong dự án. PM duy trì tầm nhìn về dự án cho nhóm phát triển dự án. Hình 2.5(a)(b) minh họa về các nhóm trong phương pháp Agile.



(a)



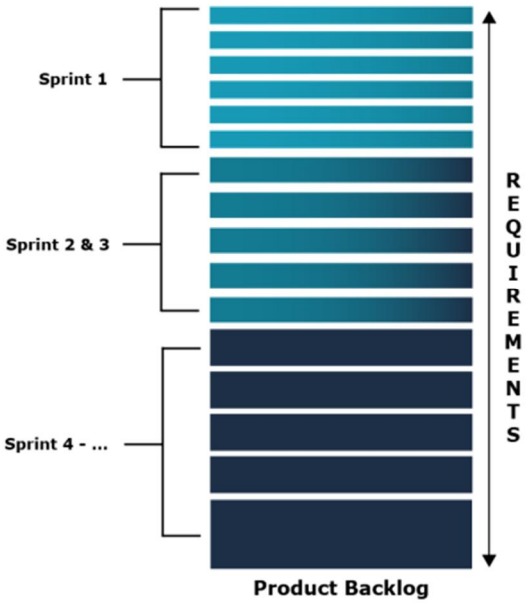
(b)

Hình 2.5. Minh họa về nhóm trong phương pháp Agile

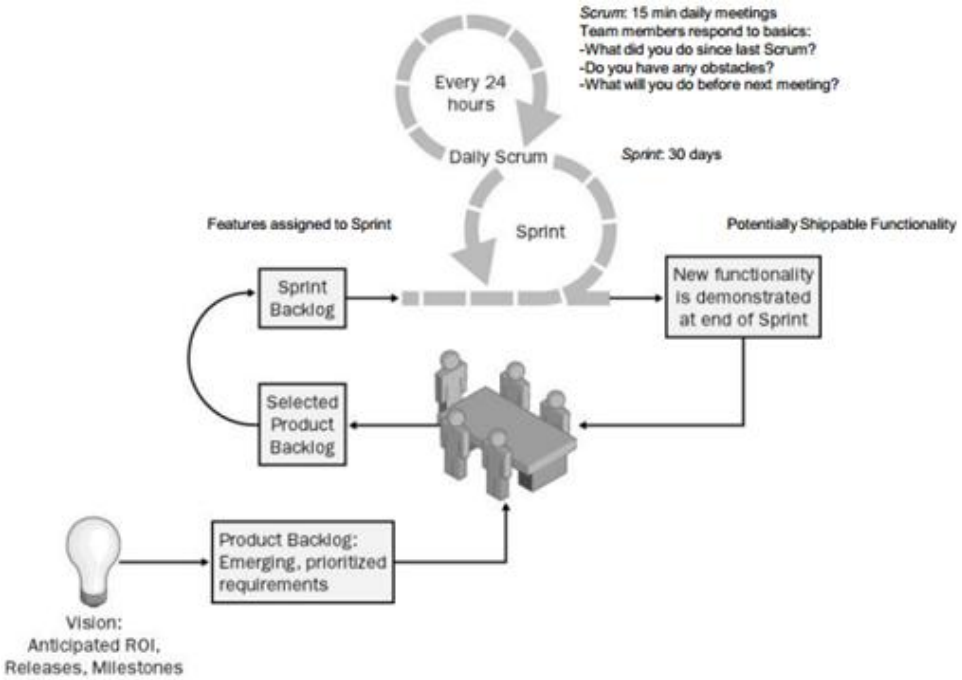
2.2.7. Quản lý yêu cầu trong Agile

2.2.7.1. Products backlog

Product Backlog là một danh sách các yêu cầu hoặc tính năng cần phát triển trong sản phẩm hoặc dịch vụ. Product Backlog là một phần quan trọng của quy trình Agile, được sử dụng để theo dõi và quản lý các yêu cầu của sản phẩm trong suốt quá trình phát triển. Product Backlog là một công cụ quan trọng trong Agile giúp quản lý và ưu tiên yêu cầu của sản phẩm, đồng thời cung cấp thông tin đầu vào cho quyết định về việc phát triển sản phẩm trong các Sprint và theo dõi tiến độ của dự án. Hình 2.6 và Hình 2.7 minh họa về sự tương quan giữa Product Backlog, Srprint backlog và Yêu cầu



Hình 2.6. Minh họa Product Backlog, Srprint backlog



Hình 2.7 Backlog và yêu cầu của dự án trong Scrum

Product Backlog được ưu tiên hóa bởi Product Owner dựa trên nghiệp vụ và yêu cầu của người dùng, và là nguồn tham khảo chính cho việc lập kế hoạch và triển khai công việc trong Sprint. Product Backlog tập trung vào việc quản lý

và ưu tiên các yêu cầu và tính năng cần phát triển trong tương lai của sản phẩm. Bảng 2.1(a)(b) là các mẫu products backlog được sử dụng trong dự án Agile.

Vai trò của Product Backlog trong Agile bao gồm:

- Tập hợp yêu cầu của sản phẩm: Product Backlog chứa tất cả các yêu cầu từ khách hàng, người dùng và các bên liên quan khác về tính năng và cải tiến của sản phẩm. Nó là nơi tập hợp và quản lý tất cả các yêu cầu cho sản phẩm.
- Ưu tiên hóa yêu cầu: Product Owner (chủ sở hữu sản phẩm) là người chịu trách nhiệm ưu tiên hóa các yêu cầu trong Product Backlog dựa trên giá trị kinh doanh, ưu tiên của khách hàng và các yếu tố khác. Các yêu cầu quan trọng hơn được đặt ở phía trên và được ưu tiên triển khai trước.
- Phân chia yêu cầu thành User Stories: Product Backlog thường chứa các yêu cầu được phân chia thành các User Stories (câu chuyện của người dùng) để mô tả yêu cầu từ góc nhìn của người dùng cuối. Điều này giúp tạo ra sự rõ ràng và dễ hiểu về những gì cần được phát triển.
- Cung cấp thông tin đầu vào cho Sprint Planning: Trong quy trình Agile, Product Backlog cung cấp thông tin đầu vào quan trọng cho cuộc họp Sprint Planning. Sprint Planning là nơi nhóm phát triển quyết định những yêu cầu nào trong Product Backlog sẽ được triển khai trong Sprint tiếp theo.
- Theo dõi tiến độ và tiến triển: Product Backlog được cập nhật thường xuyên để phản ánh tiến độ và tiến triển của dự án. Các yêu cầu được di chuyển qua lại giữa các trạng thái như "To-Do" (Đã sắp xếp), "In Progress" (Đang tiến hành), và "Done" (Hoàn thành) khi chúng được phát triển và hoàn thành.

Bảng 2.1. Mẫu Products backlog

	A	B	C	E	G	I	K	M	O	Q	S
2	Actor		General activity	Find job			Manage vacancy			Recruit candidate	
4			General user task	Browse jobs	Port resume	Receive job alert	Post vacancy	Amend vacancy	Cancel vacancy	Search candidates	Contact candidates
5			V 1.0 (mm/dd/yyyy)								
6			As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....
8			As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....		As a ...(user/role)...I want to ...(action)...so that ...(objective)....		As a ...(user/role)...I want to ...(action)...so that ...(objective)....		As a ...(user/role)...I want to ...(action)...so that ...(objective)....
10				As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....		As a ...(user/role)...I want to ...(action)...so that ...(objective)....				As a ...(user/role)...I want to ...(action)...so that ...(objective)....
12			Log system events	Perform general search	Upload resume	Subscribe 'Job Alert'	Submit job vacancy	Edit vacancy	Remove a vacancy	Review application list	Send PM to job seeker
14				Perform advanced search			Register as Employer		Confirm vacancy removal		
15			V 2.0(mm/dd/yyyy)								
16			As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....	As a ...(user/role)...I want to ...(action)...so that ...(objective)....
18			Archive system data per hour		Build resume online	Receive SMS notification for Job Alert	Approve vacancy amendment	Receive notification for expired post	Perform advanced candidate	Invite for interview	

(a)

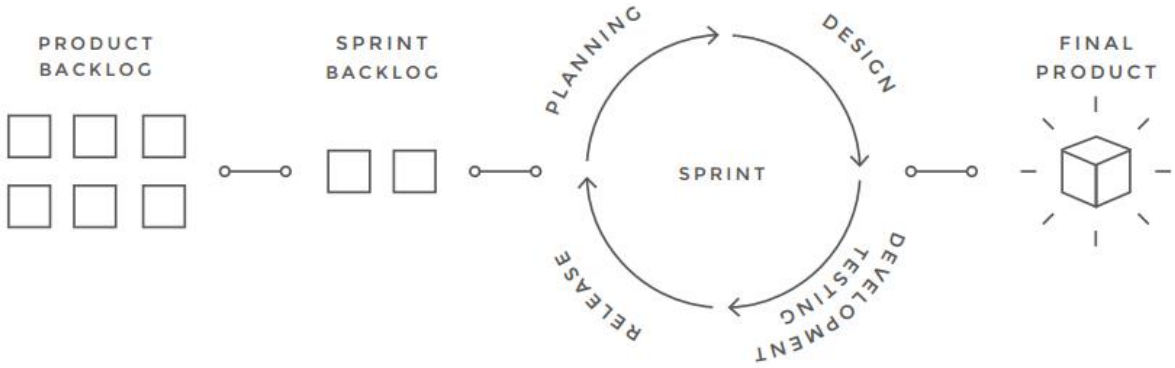
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Actor				User story ID		User story		Work pakage		Estimate (size: Small/Medium/Large)		Priority [1,n]		Sprint [1,n]		Task owner		Estimated effort (hour)
2	V 1.0 (dd/mm/yyyy)																		
3	Bảo mẫu		Tìm việc		US002		Là bảo mẫu, tôi muốn xem được danh sách các thông tin tuyển bảo mẫu để tìm kiếm công việc.		Xem danh sách tin tuyển dụng		Small		1		1		J Smith		16
5			Đăng tin		US003		Là bảo mẫu, tôi muốn đọc được chi tiết các thông tin tuyển người từ phụ huynh		Xem chi tiết tin tuyển dụng										
7	Phụ huynh				US001		Là người tìm kiếm bảo mẫu tôi muốn đăng một thông tin tìm kiếm bảo mẫu để trông đứa trẻ nhà tôi.		Đăng thông tin tuyển bảo mẫu										
9					US004		Là phụ huynh, tôi muốn gửi tin nhắn tới bảo mẫu		Gửi tin nhắn cho bảo mẫu										
11	Quản trị viên																		
13																			
15					US002		As a ...(user/role)...I want to ...(action)...so that ...(objective)....		...(action)...		Medium		3		2				
16																			
17	V 2.0 (dd/mm/yyyy)																		

(b)

2.2.7.2. Sprint backlog

Sprint backlog là một danh sách các công việc cụ thể mà nhóm phát triển cam kết hoàn thành trong một Sprint (chu kỳ phát triển) cụ thể trong Agile. Nó được tạo ra từ việc chọn các user stories từ product backlog và chuyển chúng thành các công việc cụ thể và có thể thực hiện trong khoảng thời gian của một Sprint. Sprint Backlog là công cụ quan trọng trong Agile giúp nhóm phát triển quản lý và theo dõi công việc cụ thể được cam kết để hoàn thành trong một

Sprint cụ thể, đảm bảo rằng nhóm có thể phát triển và giao sản phẩm có giá trị trong một khoảng thời gian ngắn. Hình 2.8 minh họa về Sprint trong Agile.



Hình 2.8. Sprint trong Agile

Vai trò của Sprint Backlog trong Agile bao gồm:

- Xác định và chuyển đổi yêu cầu thành công việc cụ thể: Sprint Backlog giúp nhóm phát triển chuyển đổi các User Stories từ Product Backlog thành các công việc cụ thể, đảm bảo rằng mỗi công việc có thể được hoàn thành trong suốt thời gian của Sprint.
- Phân công công việc và gán trách nhiệm: Sprint Backlog là nơi nhóm phát triển phân công và gán trách nhiệm cho mỗi công việc cho các thành viên của nhóm. Điều này giúp đảm bảo rằng mỗi thành viên có hiểu biết rõ về nhiệm vụ của họ trong suốt Sprint.
- Theo dõi tiến độ và tiến triển: Sprint Backlog là công cụ để theo dõi tiến độ của nhóm phát triển trong suốt Sprint. Các công việc được di chuyển qua lại giữa các trạng thái như "To-Do" (Đã sắp xếp), "In Progress" (Đang tiến hành), và "Done" (Hoàn thành) khi chúng được thực hiện.
- Điều chỉnh và thích nghi: Trong quá trình Sprint, Sprint Backlog có thể được điều chỉnh và thích nghi tùy thuộc vào tiến trình phát triển và những thay đổi xuất hiện. Các công việc mới có thể được thêm vào

hoặc các công việc không cần thiết có thể được loại bỏ để đảm bảo rằng Sprint có thể hoàn thành đúng hạn và đạt được mục tiêu của nó.

2.2.7.3. WIP

WIP (Work In Progress) là số lượng công việc hoặc nhiệm vụ đang được thực hiện đồng thời trong một giai đoạn hoặc quy trình cụ thể. WIP thường ám chỉ số lượng các User Stories, công việc, hay tính năng đang ở trạng thái đang thực hiện, từ khi bắt đầu đến khi hoàn thành. WIP đóng vai trò quan trọng trong Agile bằng cách giúp kiểm soát và quản lý dòng công việc, tăng hiệu suất làm việc và khả năng dự đoán, cũng như tối ưu hóa quá trình làm việc của nhóm.

Vai trò của WIP trong Agile bao gồm:

- Quản lý dòng công việc: WIP giúp nhóm giới hạn số lượng công việc mà họ đang làm đồng thời. Điều này giúp tránh tình trạng quá tải công việc, tăng hiệu suất làm việc và giảm thời gian hoàn thành các công việc.
- Tăng sự tập trung: Bằng cách giảm thiểu số lượng công việc đang thực hiện đồng thời, WIP giúp tăng sự tập trung của nhóm vào các công việc quan trọng nhất và ưu tiên nhất.
- Phát hiện và giải quyết vấn đề nhanh chóng: Khi WIP được kiểm soát, bất kỳ vấn đề nào phát sinh cũng được nhận biết sớm và giải quyết nhanh chóng hơn, vì nhóm tập trung vào một số lượng nhỏ hơn các công việc cùng một lúc.
- Tăng khả năng dự đoán: Bằng cách kiểm soát WIP, nhóm có thể dễ dàng dự đoán thời gian hoàn thành các công việc và dự án hơn, vì họ biết chính xác số lượng công việc đang được thực hiện đồng thời và tốc độ hoàn thành của họ.

- Tối ưu hóa quá trình làm việc: WIP cung cấp thông tin giúp nhóm phát triển hiểu rõ hơn về hiệu suất làm việc của họ và cải thiện quy trình làm việc theo thời gian.

2.2.7.4. Phân bổ user story vào các sprint

Phân bổ các user story trên product backlog items vào các sprint là một bước quan trọng trong việc lập kế hoạch sprint và quản lý dự án. Quá trình này thường dựa trên một số yếu tố chính sau:

- Khả năng làm việc của đội (Team Capacity): Đánh giá tổng số giờ mà đội có thể làm việc trong một sprint, bao gồm cả thời gian làm việc của từng thành viên. Điều này bao gồm việc xem xét các yếu tố như ngày nghỉ, cuộc họp, và bất kỳ công việc không liên quan nào khác.

- Ưu tiên của Product Owner: Product Owner sẽ xác định các user story quan trọng nhất dựa trên giá trị kinh doanh, phản hồi của khách hàng, và các yếu tố chiến lược khác. Các user story có độ ưu tiên cao sẽ được chọn để thực hiện trước

- Điểm ước lượng (Story Points) và nỗ lực ước tính (Estimated Effort): Mỗi user story có điểm ước lượng (story points) và nỗ lực ước tính (estimated effort) giúp xác định độ khó và thời gian cần thiết để hoàn thành. Dựa trên khả năng làm việc của đội, bạn có thể nhóm các user story sao cho tổng số điểm ước lượng hoặc nỗ lực ước tính không vượt quá khả năng của đội trong một sprint.

- Phụ thuộc giữa các user story: Xem xét các phụ thuộc giữa các user story. Một số user story cần phải hoàn thành trước khi các user story khác có thể bắt đầu. Việc xác định và giải quyết các phụ thuộc này sẽ giúp sắp xếp các user story hợp lý hơn trong các sprint.

- Mục tiêu Sprint: Mỗi sprint nên có một hoặc nhiều mục tiêu rõ ràng. Các user story được chọn cho sprint nên liên quan đến việc đạt được các mục tiêu này.

2.2.7.5. Mục tiêu sprint

- Xác định phạm vi sprint: phạm vi của sprint được viết dựa trên Product backlog, bao gồm các user stories và tasks/subtasks cụ thể mà đội phát triển sẽ hoàn thành trong sprint. Điều này giúp xác định rõ ràng các mục tiêu cần đạt được.
- Đặt mục tiêu cụ thể, rõ ràng: mục tiêu của sprint nên cụ thể và tập trung vào một hoặc một vài tính năng chính. Mục tiêu nên rõ ràng để cả đội phát triển và các bên liên quan đều hiểu được.
- Đảm bảo mục tiêu có thể đo lường: mục tiêu của sprint cần phải có thể đo lường được để xác định khi nào mục tiêu đã hoàn thành. Điều này giúp đánh giá được kết quả của sprint.
- Phù hợp với mục tiêu nghiệp vụ: mục tiêu của sprint nên phù hợp với các mục tiêu nghiệp vụ tổng thể của dự án hoặc sản phẩm. Điều này đảm bảo rằng sprint đang đóng góp vào giá trị cốt lõi của dự án.
- Đặt mục tiêu thực tế: Mục tiêu của sprint cần phải thực tế và có thể hoàn thành trong thời gian quy định của sprint (thường là 2 đến 4 tuần). Điều này giúp đảm bảo rằng đội phát triển có thể đạt được mục tiêu mà không bị quá tải hãy trễ thời gian phát hành sản phẩm.
- Đưa vào các yêu cầu quan trọng: Chọn các yêu cầu hoặc user stories quan trọng nhất từ Product backlog để đưa vào Sprint backlog. Điều này đảm bảo rằng những tính năng có giá trị cao nhất được phát triển trước.
- Xác định tiêu chí hoàn thành: định nghĩa rõ ràng các tiêu chí để một công việc được coi là hoàn thành. Điều này bao gồm cả kiểm thử và tài liệu liên quan.