BÀI 2
XÁC ĐỊNH QUY TRÌNH PHÁT TRIỂN PHẦN MỀM

1. Quy trình và mô hình quy trình

1.1. Quy trình

Vòng đời phát triển phần mềm (SDLC-Software Development Life Cycle) là một quá trình bao gồm một loạt các hoạt động được lên kế hoạch để phát triển hoặc thay đổi Sản phẩm Phần mềm. SDLC còn được gọi là Quy trình phát triển phần mềm.

SDLC là một quy trình được ngành công nghiệp phần mềm sử dụng để thiết kế, phát triển và kiểm tra phần mềm chất lượng cao. SDLC nhằm mục đích tạo ra một phần mềm chất lượng cao đáp ứng hoặc vượt quá mong đợi của khách hàng, hoàn thành trong thời gian và chi phí ước tính.

SDLC là một quy trình theo sau cho một dự án phần mềm, trong một tổ chức phần mềm, bao gồm một kế hoạch chi tiết mô tả cách phát triển, bảo trì, thay thế và thay đổi hoặc nâng cao phần mềm cụ thể.

SDLC xác định một phương pháp luận để cải thiện chất lượng của phần mềm và quá trình phát triển tổng thể.

SDLC là một khuôn khổ xác định các tác vụ được thực hiện ở mỗi bước trong quy trình phát triển phần mềm.

Các hoạt động cơ bản của quy trình phát triển phần mềm:

Stage 1: Planning and Requirement Analysis (Lập kế hoạch và Phân tích Yêu cầu)

Stage 2: Defining Requirements (Xác định/Định nghĩa yêu cầu)

Stage 3: Designing the Product Architecture (Thiết kế Kiến trúc Sản phẩm)

Stage 4: Building or Developing the Product (Xây dựng/phát triển sản phẩm)

Stage 5: Testing the Product (Kiểm tra sản phẩm)

Stage 6: Deployment in the Market and Maintenance (Triển khai/phát hành trên thị trường và duy trì/bảo trì)

1.2. Mô hình quy trình

Mô hình quy trình- Procedure model:

- ❖ Mô hình quy trình (Procedure model) là một chiến lược phát triển phần mềm, ở đây: bao gồm các cách thức kết hợp, sử dụng tiến trình (process) phần mềm, cách vận dụng các phương pháp và các công cụ trong mỗi giai đoạn phát triển.
- ❖ Mỗi mô hình quy trình tuân theo một loạt các bước duy nhất cho kiểu của nó để đảm bảo sự thành công trong quá trình phát triển phần mềm.
- ❖ Một số mô hình quy trình phổ biến: Waterfall, Iterative, Spiral, Big Bang, V-Model,…

2. Mô hình thác nước

2.1. Giới thiệu mô hình

Mô hình thác nước (Waterfall model) là Mô hình Quy trình đầu tiên được giới thiệu. Nó cũng được gọi là mô hình vòng đời tuần tự tuyến tính. Nó rất đơn giản để hiểu và sử dụng. Trong mô hình thác nước, mỗi giai đoạn phải được hoàn thành trước khi giai đoạn tiếp theo có thể bắt đầu và không có sự chồng chéo trong các giai đoạn.

Waterfall là mô hình quy trình đầu tiên và nổi tiếng nhất được giới thiệu, với cách tiếp cận Vòng đời phát triển hệ thống (SDLC) lâu đời nhất, được sử dụng để phát triển phần mềm.

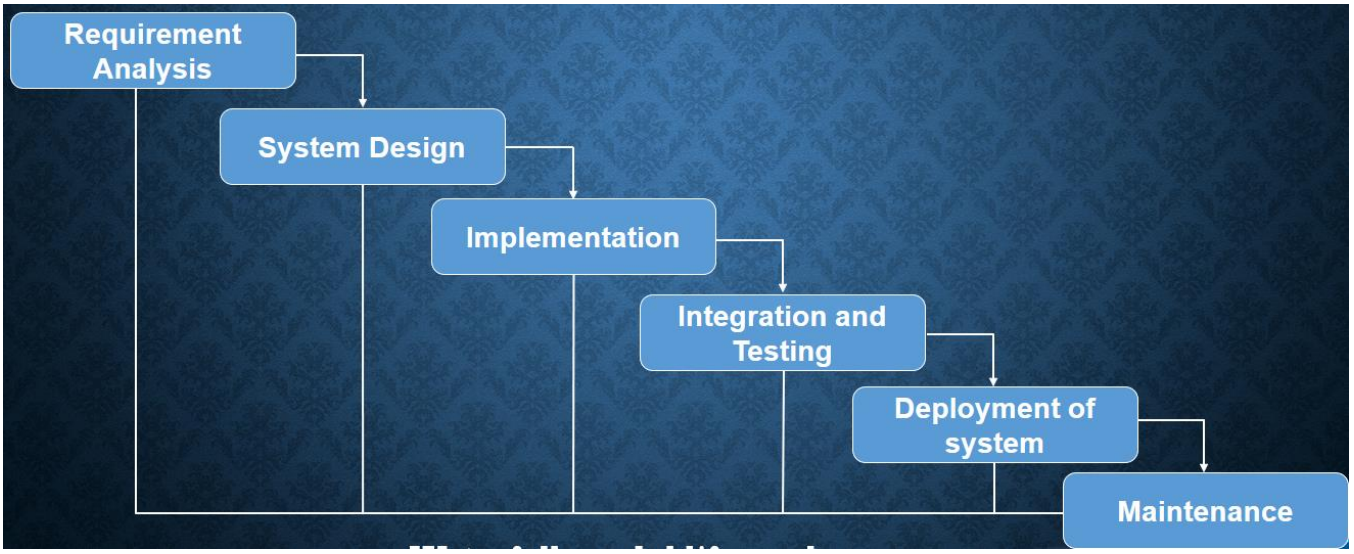
Mô hình thác nước minh họa quá trình phát triển phần mềm theo dòng tuần tự tuyến tính. Điều này có nghĩa là bất kỳ giai đoạn nào trong quá trình phát triển chỉ bắt đầu nếu giai đoạn trước đó hoàn thành. Trong mô hình thác nước, các pha không chồng lên nhau.

2.2. Các pha trong mô hình

Hình 2.1 minh họa về các bước trong mô hình Waterfall, bao gồm:

- ➢ Thu thập và phân tích yêu cầu (Requirement Analysis): Tất cả các yêu cầu có thể có của hệ thống được phát triển đều được ghi lại trong giai đoạn này và được ghi lại trong tài liệu đặc tả yêu cầu để phục vụ cho các giai đoạn sau;

- Thiết kế hệ thống (System Design): Các thông số kỹ thuật yêu cầu từ giai đoạn đầu được nghiên cứu trong giai đoạn này và thiết kế hệ thống được chuẩn bị. Thiết kế hệ thống này giúp xác định các yêu cầu phần cứng và hệ thống cũng như giúp xác định kiến trúc hệ thống tổng thể;
- Thực hiện (Implementation): Với đầu vào từ thiết kế hệ thống, hệ thống được phát triển đầu tiên trong các chương trình nhỏ gọi là đơn vị, được tích hợp trong giai đoạn tiếp theo. Mỗi đơn vị được phát triển và kiểm tra chức năng của nó (hay còn có thể được gọi là Kiểm thử đơn vị);
- Tích hợp và Kiểm thử (Integration and Testing): Tất cả các đơn vị được phát triển trong giai đoạn triển khai được tích hợp vào một hệ thống sau khi thử nghiệm của mỗi đơn vị. Sau khi tích hợp, toàn bộ hệ thống được kiểm tra xem có bất kỳ lỗi và hỏng hóc nào không (giai đoạn này còn được gọi là kiểm thử tích hợp).
- Triển khai hệ thống (Deployment of system): Sau khi kiểm tra chức năng và phi chức năng được thực hiện; sản phẩm được triển khai trong môi trường khách hàng hayđược tung ra thị trường;
- Bảo trì (Maintenance): Có một số vấn đề xảy ra trong môi trường khách hàng. Để khắc phục những vấn đề đó, các bản vá lỗi được phát hành. Ngoài ra để nâng cao sản phẩm một số phiên bản tốt hơn được phát hành. Bảo trì được thực hiện để mang lại những thay đổi này trong môi trường khách hàng.



Hình 2.1 Các pha trong mô hình thác nước

2.3. Đánh giá

Mô hình thác nước minh họa quá trình phát triển phần mềm theo dòng tuần tự tuyến tính. Điều này có nghĩa là bất kỳ giai đoạn nào trong quá trình phát triển chỉ bắt đầu nếu giai đoạn trước đó hoàn thành. Tất cả các giai đoạn này được xếp tầng với nhau, trong đó tiến trình được xem như chảy đều đặn xuống dưới (giống như thác nước) qua các giai đoạn. Giai đoạn tiếp theo chỉ được bắt đầu sau khi đạt được tập hợp mục tiêu đã xác định cho giai đoạn trước và nó được ký kết, vì vậy có tên "Mô hình thác nước". Trong mô hình này, các giai đoạn không chồng chéo lên nhau.

- ***Ưu điểm***: Cho phép thiết lập một lịch trình với các thời hạn cho từng giai đoạn phát triển và một sản phẩm, và có thể tiến hành từng giai đoạn của mô hình quy trình phát triển. Sự phát triển di chuyển từ ý tưởng, thông qua thiết kế, thực hiện, thử nghiệm, cài đặt, khắc phục sự cố và kết thúc là vận hành và bảo trì. Mỗi giai đoạn phát triển diễn ra theo thứ tự nghiêm ngặt. Một số ưu điểm chính của Mô hình thác nước như sau:
  - ✓ Đơn giản, dễ hiểu và sử dụng
  - ✓ Dễ dàng quản lý do độ cứng của mô hình.
  - ✓ Mỗi giai đoạn có các phân phối cụ thể và một quy trình xem xét.
  - ✓ Các giai đoạn được xử lý và hoàn thành cùng một lúc.
  - ✓ Hoạt động tốt cho các dự án nhỏ hơn, nơi các yêu cầu được hiểu rất rõ.
  - ✓ Các giai đoạn được xác định rõ ràng.
  - ✓ Các mốc quan trọng được hiểu rõ.
  - ✓ Dễ dàng sắp xếp các công việc.
  - ✓ Quá trình và kết quả được ghi chép đầy đủ.
- **Nhược điểm:** Nó không cho phép phản ánh hoặc sửa đổi nhiều. Một khi ứng dụng đang trong giai đoạn thử nghiệm, rất khó để quay lại và thay đổi một cái gì đó không được ghi chép đầy đủ hoặc được nghĩ đến trong giai đoạn khái niệm. Những nhược điểm chính của Mô hình thác nước như sau:
  - ✓ Không có phần mềm đang hoạt động nào được sản xuất cho đến cuối vòng đời.

- ✓ Lượng rủi ro cao và không chắc chắn.
- ✓ Không phải là một mô hình tốt cho các dự án phức tạp và hướng đối tượng.
- ✓ Mô hình kém cho các dự án dài và đang diễn ra. Không phù hợp với các dự án mà các yêu cầu có nguy cơ thay đổi từ trung bình đến cao.
- ✓ Vì vậy, rủi ro và sự không chắc chắn là cao với mô hình quy trình này.
- ✓ Rất khó để đo lường sự tiến bộ trong các giai đoạn.
- ✓ Không thể đáp ứng các yêu cầu thay đổi.

❖ **Ứng dụng:** Mỗi phần mềm được phát triển đều khác nhau và đòi hỏi phải tuân theo một cách tiếp cận SDLC phù hợp dựa trên các yếu tố bên trong và bên ngoài. Một số dự án phần mềm thích hợp với mô hình Waterfall như:

- ✓ Các yêu cầu được ghi chép rất đầy đủ, rõ ràng và cố định.
- ✓ Mô tả (định nghĩa) về sản phẩm ổn định.
- ✓ Công nghệ được hiểu và không mang tính năng động.
- ✓ Không có yêu cầu mơ hồ.
- ✓ Có sẵn các nguồn lực dồi dào với kiến thức chuyên môn cần thiết để hỗ trợ sản phẩm.
- ✓ Dự án ngắn hạn.

3. Mô hình tiếp cận lặp

3.1. Giới thiệu mô hình

Mô hình tiếp cận lặp (Iterative model hay Iterative and Incremental model) được đề xuất dựa trên ý tưởng thay vì phải xây dựng và chuyển giao hệ thống một lần thì sẽ được chia thành nhiều vòng, tăng dần.

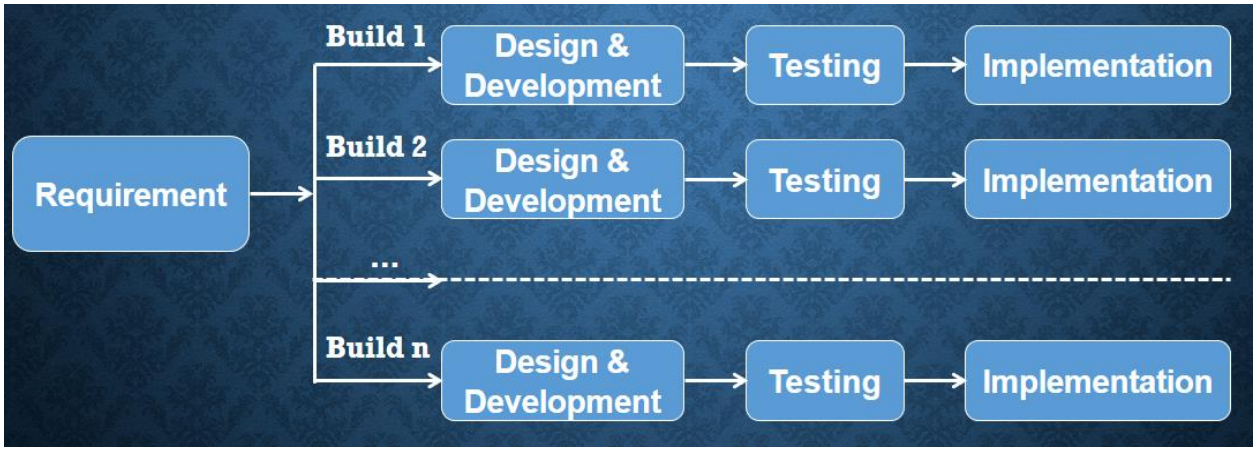
Trong mô hình Iterative, quy trình lặp lại bắt đầu bằng việc triển khai đơn giản một tập hợp nhỏ các yêu cầu phần mềm và cải tiến lặp đi lặp lại các phiên bản đang phát triển cho đến khi hệ thống hoàn chỉnh được triển khai và sẵn sàng được triển khai. Một mô hình vòng đời lặp đi lặp lại không cố gắng bắt đầu với đặc điểm kỹ thuật đầy đủ của các yêu cầu. Thay vào đó, quá trình phát triển bắt đầu bằng cách chỉ định và triển khai chỉ một phần của phần mềm, sau đó sẽ được xem xét để xác định các yêu cầu

tiếp theo. Quá trình này sau đó được lặp lại, tạo ra một phiên bản mới của phần mềm vào cuối mỗi lần lặp lại của mô hình.

Ý tưởng cơ bản đằng sau phương pháp này là phát triển một hệ thống thông qua các chu kỳ lặp đi lặp lại (Iterative) và theo các phần nhỏ hơn tại một thời điểm (Incremental).

Trong mô hình Lặp lại, quy trình lặp lại bắt đầu bằng việc triển khai đơn giản một tập hợp nhỏ các yêu cầu phần mềm và cải tiến lặp đi lặp lại các phiên bản đang phát triển cho đến khi hệ thống hoàn chỉnh được triển khai và sẵn sàng được triển khai. Một mô hình vòng đời lặp đi lặp lại không cố gắng bắt đầu với đặc điểm kỹ thuật đầy đủ của các yêu cầu. Thay vào đó, quá trình phát triển bắt đầu bằng cách chỉ định và triển khai chỉ một phần của phần mềm, sau đó sẽ được xem xét để xác định các yêu cầu tiếp theo. Quá trình này sau đó được lặp lại, tạo ra một phiên bản mới của phần mềm vào cuối mỗi lần lặp lại của mô hình.

3.2. Các pha trong mô hình



Hình 2.2 Các pha trong mô hình Iterative

- Requirement: Phân tích và định nghĩa yêu cầu phần mềm
- Build 1: Triển khai đơn giản với một tập nhỏ các yêu cầu.
- Build i: Cải tiến phiên bản trước đó bằng cách bổ sung thêm các yêu cầu để tạo ra một phiên bản mới.
- Build n: Giai đoạn cuối xây dựng được một phiên bản phần mềm hoàn thiện với đầy đủ các yêu cầu phần mềm.

Ý tưởng cơ bản đằng sau phương pháp này là phát triển một hệ thống thông qua các chu kỳ lặp đi lặp lại (lặp đi lặp lại) và theo các phần nhỏ hơn tại một thời điểm (tăng dần). Quá trình lặp đi lặp lại bắt đầu với việc triển khai đơn giản một tập hợp con các yêu cầu phần mềm và cải tiến lặp đi lặp lại các phiên bản đang phát triển cho đến khi triển khai toàn bộ hệ thống. Ở mỗi lần lặp lại, các sửa đổi thiết kế được thực hiện và các khả năng chức năng mới được thêm vào.

Từ đó, chúng ta có thể thấy rõ một số ưu điểm của mô hình phát triển tăng vòng:

- Sau mỗi lần tăng vòng thì có thể chuyển giao kết quả thực hiện được cho khách hành nên các chức năng của hệ thống có thể nhìn thấy sớm hơn.

- Các vòng trước đóng vai trò là mẫu thử để giúp tìm hiểu thêm các yêu cầu ở những vòng tiếp theo.

- Những chức năng của hệ thống có thứ tự ưu tiên càng cao thì sẽ được kiểm thử càng kỹ.

3.3. Đánh giá

Trong quá trình phát triển phần mềm, các lần lặp lại chu trình phát triển phần mềm có thể cùng diễn ra. Quá trình này có thể được mô tả như là một cách tiếp cận "tiếp thu tiến hóa" hoặc "xây dựng gia tăng".

Trong mô hình gia tăng này, toàn bộ yêu cầu được chia thành nhiều bản dựng khác nhau. Trong mỗi lần lặp lại, mô-đun phát triển trải qua các giai đoạn yêu cầu, thiết kế, triển khai và thử nghiệm. Mỗi bản phát hành tiếp theo của mô-đun bổ sung chức năng cho bản phát hành trước đó. Quá trình tiếp tục cho đến khi hệ thống hoàn chỉnh đã sẵn sàng theo yêu cầu.

Chìa khóa để sử dụng thành công mô hình Iterative là xác nhận nghiêm ngặt các yêu cầu, xác minh & kiểm tra từng phiên bản của phần mềm so với các yêu cầu đó trong mỗi chu kỳ của mô hình.

Mô hình Iterative thích hợp với dự án phần mềm có:

- Các yêu cầu của hệ thống được xác định hoàn chỉnh và rõ ràng.

- Các yêu cầu chính phải được xác định trước khi thực hiện các giai đoạn lặp; tuy nhiên, một số chức năng hoặc các cải tiến được yêu cầu có thể phát triển theo thời gian.

❖ **Ưu điểm:**

- Một số chức năng phần mềm có thể được phát triển nhanh chóng và sớm được đưa vào sử dụng trong vòng đời;
- Kết quả thu được sớm và theo định kỳ;
- Có thể lập kế hoạch phát triển song song;
- Ít tốn kém hơn để thay đổi phạm vi/yêu cầu, hỗ trợ thay đổi yêu cầu;
- Dễ dàng kiểm tra và gỡ lỗi trong quá trình lặp lại nhỏ hơn;
- Một số chức năng phần mềm có thể được phát triển nhanh chóng và sớm được đưa vào sử dụng trong vòng đời;
- Kết quả thu được sớm và theo định kỳ;
- Có thể lập kế hoạch phát triển song song;
- Ít tốn kém hơn để thay đổi phạm vi/yêu cầu, hỗ trợ thay đổi yêu cầu;
- Dễ dàng kiểm tra và gỡ lỗi trong quá trình lặp lại nhỏ hơn;

❖ **Nhược điểm:**

- Cần nhiều tài nguyên hơn;
- Các vấn đề về kiến trúc hoặc thiết kế hệ thống có thể phát sinh;
- Không thích hợp cho các dự án nhỏ;
- Sự phức tạp trong quản lý là nhiều hơn;
- Cần có nguồn lực có kỹ năng cao để phân tích rủi ro;
- Tiến độ dự án phụ thuộc nhiều vào giai đoạn phân tích rủi ro

4. Mô hình bản mẫu

4.1. Giới thiệu mô hình

Mô hình Tạo mẫu phần mềm (Software Prototype model) hay mô hình bản mẫu (Prototyping model) đề cập đến việc xây dựng các nguyên mẫu ứng dụng phần mềm hiển thị chức năng của sản phẩm đang được phát triển, nhưng có thể không thực sự giữ logic chính xác của phần mềm gốc.

Tạo mẫu phần mềm đang trở nên rất phổ biến như một mô hình phát triển phần mềm, vì nó cho phép hiểu các yêu cầu của khách hàng ở giai đoạn phát triển ban đầu. Nó giúp nhận được phản hồi có giá trị từ khách hàng và giúp các nhà thiết kế và phát triển phần mềm hiểu chính xác những gì được mong đợi từ sản phẩm đang được phát triển.

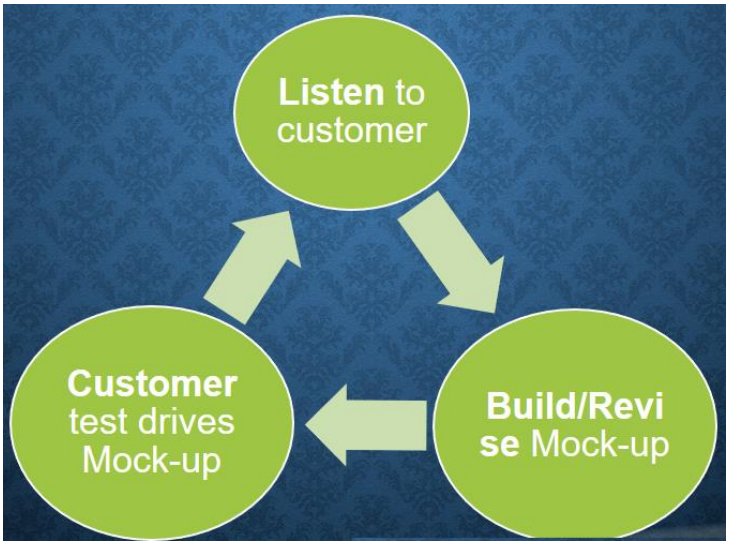
Tạo mẫu phần mềm là gì?

- Nguyên mẫu là một mô hình phần mềm hoạt động với một số chức năng hạn chế.
- Prototyping được sử dụng để cho phép người dùng đánh giá các đề xuất của nhà phát triển và thử chúng trước khi triển khai. Nó cũng giúp hiểu các yêu cầu dành riêng cho người dùng và có thể chưa được nhà phát triển xem xét trong quá trình thiết kế sản phẩm.

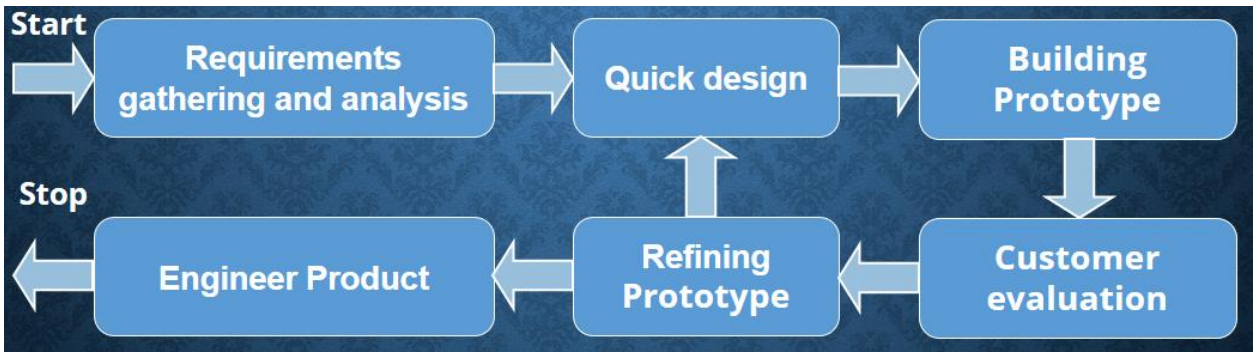
4.2. Các pha trong mô hình

Hình 2.3 mô tả 3 giai đoạn trong mô hình Prototyping:

- Nge khách trình bày;
- Tạo/sửa bản mẫu;
- Khách kiểm tra bản mẫu



Hình 2.3 Các giai đoạn trong mô hình Prototyping



Hình 2.4 Các pha trong mô hình Prototyping

- Thu thập và phân tích yêu cầu (Requirements gathering and analysis): Trong giai đoạn này của mô hình nguyên mẫu, việc xác định yêu cầu cơ bản được thực hiện bởi các thành viên trong nhóm phân tích sản phẩm. Nó liên quan đến việc hiểu yêu cầu chức năng của người dùng bởi nhóm phân tích yêu cầu. Sau khi hoàn toàn hiểu nhu cầu người dùng, các thành viên trong nhóm đã sẵn sàng một tài liệu công cụ yêu cầu sản phẩm. Nhưng các chi tiết phức tạp hơn của thiết kế bên trong và các khía cạnh bên ngoài như hiệu suất và bảo mật có thể bị bỏ qua ở giai đoạn này.
- Thiết kế nhanh (Quick design): Trong giai đoạn thiết kế nhanh của nhóm phát triển mô hình nguyên mẫu đang phát triển nguyên mẫu ban đầu của hệ thống. Trong giai đoạn này của mô hình, các yêu cầu rất cơ bản được giới thiệu và giao diện người dùng được cung cấp. Các tính năng này được sử dụng trong thiết kế nhanh có thể không hoạt động chính xác theo cách giống như phần mềm thực tế được phát triển. Nhưng môi trường làm việc tổng thể xung quanh được sử dụng để mang lại giao diện tương tự như khi phát triển sản phẩm cuối cùng cho khách hàng trong nguyên mẫu được phát triển.
- Xây dựng nguyên mẫu(Building Prototype): Trong giai đoạn này của mô hình nguyên mẫu sẽ thiết kế nhanh từ đầu ra của thiết kế nhanh và xây dựng thiết kế hệ thống đó tại địa phương. Nguyên mẫu tòa nhà địa phương này mang lại một cái nhìn và cảm giác tương tự như sản phẩm cuối cùng được thiết kế.
- Đánh giá của khách hàng(Customer evaluation): Sau khi xây dựng nguyên mẫu của hệ thống, quá trình đánh giá của khách hàng đang được tiến hành trong giai đoạn đánh giá của khách hàng về mô hình Nguyên mẫu. Khách hàng kiểm tra

chức năng làm việc của nguyên mẫu xây dựng và đưa ra phản hồi cho nhóm phát triển. Những phản hồi này từ khách hàng và người xếp chồng của sản phẩm được thu thập một cách có tổ chức và được sử dụng để cải tiến thêm cho sản phẩm đang trong giai đoạn phát triển.

- ❖ Tinh chế sản phẩm(Refining Prototype): Giai đoạn này còn được gọi là sửa đổi và nâng cao nguyên mẫu. Trong sản phẩm tinh chỉnh, nhóm phát triển thảo luận về phản hồi và đánh giá của khách hàng về nguyên mẫu xây dựng và nếu có bất kỳ vấn đề nào xảy ra trong sản phẩm thì nhóm phát triển sẽ cố gắng tinh chỉnh dự án và liên tục thực hiện mong muốn của khách hàng về hệ thống. Giai đoạn này, một số cuộc đàm phán xảy ra với khách hàng dựa trên một số yếu tố khác nhau như hạn chế về thời gian, ngân sách và tính khả thi về mặt kỹ thuật của việc triển khai thực tế. Tất cả những thay đổi được chấp nhận này lại được đưa vào Nguyên mẫu mới được phát triển và chu trình lặp lại cho đến khi đáp ứng được kỳ vọng của khách hàng.
- ❖ Sản phẩm của kỹ sư (Engineer Product): Sau khi tất cả các phản hồi liên tiếp và các đánh giá tích cực hình thành cho khách hàng, nguyên mẫu xây dựng do nhóm thiết kế và phát triển phần mềm tiếp quản. Sản phẩm thực tế được thiết kế và phát triển trong giai đoạn này của mô hình nguyên mẫu.

4.3. Đánh giá

Mô hình **Prototyping** thích hợp với dự án phần mềm:

- ❖ **Prototyping** hữu ích nhất trong việc phát triển các hệ thống có mức độ tương tác cao của người dùng như hệ thống trực tuyến. Các hệ thống cần người dùng điền vào các biểu mẫu hoặc duyệt qua các màn hình khác nhau trước khi dữ liệu được xử lý có thể sử dụng tạo mẫu rất hiệu quả để cung cấp giao diện chính xác ngay cả trước khi phần mềm thực tế được phát triển.
- ❖ Khi mới rõ mục đích chung chung của phần mềm, chưa rõ chi tiết đầu vào hay xử lý ra sao hoặc chưa rõ yêu cầu đầu ra;
- ❖ Dùng như “Hệ sơ khai” để thu thập yêu cầu người dùng qua các thiết kế nhanh;
- ❖ Các giải thuật, kỹ thuật dùng làm bản mẫu có thể chưa nhanh, chưa tốt, miễn là có mẫu để thảo luận gợi yêu cầu của người dùng

Ưu điểm:

- ❖ Ưu điểm để phát triển phần mềm trong mô hình nguyên mẫu là mô hình này cho phép giao diện người dùng cao của khách hàng với hệ thống đã phát triển. Phương pháp làm việc của mô hình nguyên mẫu là xây dựng một mô hình cung cấp giao diện thực tế của hệ thống đang được phát triển để khách hàng đánh giá và phản hồi về chức năng hệ thống. Các lỗi và sự cố có thể được phát hiện rất dễ dàng.
- ❖ Tăng sự tham gia của người dùng vào sản phẩm ngay cả trước khi triển khai.Kể từ khi mô hình hoạt động của hệ thống được hiển thị, người dùng hiểu rõ hơn về hệ thống đang được phát triển.
- ❖ Giảm thời gian và chi phí vì các lỗi có thể được phát hiện sớm hơn nhiều.Phản hồi của người dùng nhanh hơn có sẵn dẫn đến các giải pháp tốt hơn.
- ❖ Chức năng bị thiếu có thể được xác định một cách dễ dàng.Các chức năng khó hiểu hoặc khó có thể được xác định.

Nhược điểm:

- ❖ Liên quan đến nhược điểm của việc phát triển hệ thống trong mô hình nguyên mẫu làm tăng độ phức tạp của hệ thống do phạm vi chức năng của hệ thống được mở rộng ra ngoài chức năng đã được lập lịch trước của hệ thống. Ứng dụng không hoàn chỉnh có thể khiến ứng dụng không được sử dụng làm hệ thống đầy đủ và chức năng của sản phẩm.
- ❖ Rủi ro phân tích yêu cầu không đủ do phụ thuộc quá nhiều vào nguyên mẫu.
- ❖ Người dùng có thể bị nhầm lẫn giữa nguyên mẫu và hệ thống thực tế.
- ❖ Trên thực tế, phương pháp luận này có thể làm tăng độ phức tạp của hệ thống vì phạm vi của hệ thống có thể mở rộng ra ngoài kế hoạch ban đầu.
- ❖ Các nhà phát triển có thể cố gắng sử dụng lại các nguyên mẫu hiện có để xây dựng hệ thống thực tế, ngay cả khi nó không khả thi về mặt kỹ thuật.
- ❖ Nỗ lực đầu tư vào việc xây dựng nguyên mẫu có thể là quá nhiều nếu nó không được giám sát đúng cách.

5. Mô hình xoắn ốc

5.1. Giới thiệu mô hình

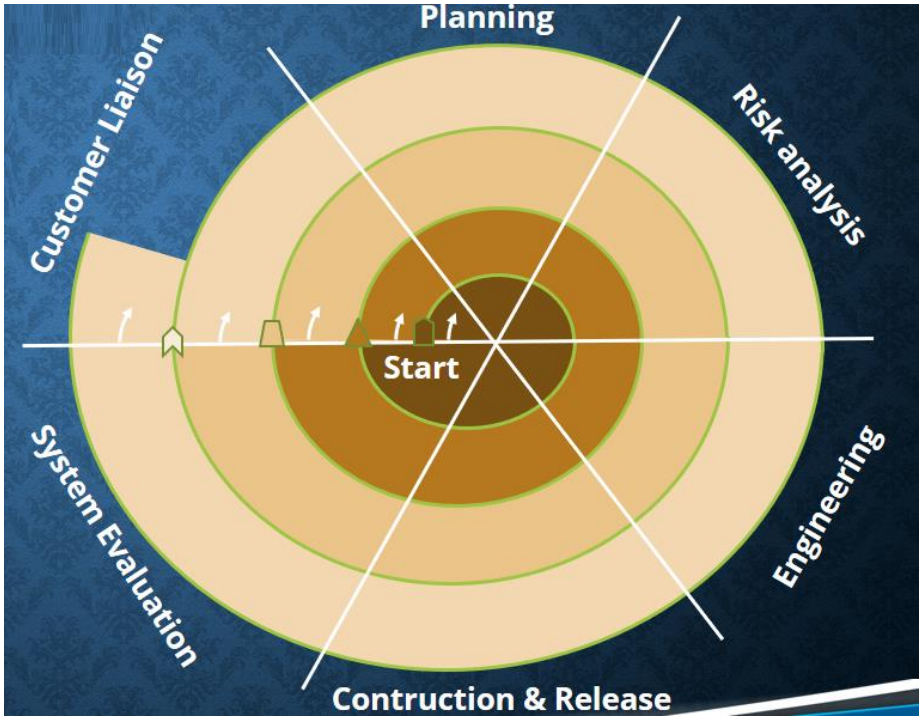
Mô hình xoắn ốc (Spiral model) kết hợp ý tưởng phát triển lặp đi lặp lại có hệ thống, được kiểm soát của mô hình thác nước. Mô hình xoắn ốc này là sự kết hợp giữa mô hình quá trình phát triển lặp đi lặp lại và mô hình phát triển tuyến tính tuần tự. Mô hình thác nước với sự nhấn mạnh rất cao vào phân tích rủi ro. Nó cho phép phát hành gia tăng sản phẩm hoặc cải tiến gia tăng thông qua mỗi lần lặp lại theo vòng xoắn ốc.

Đường phát triển xoắn ốc, tính từ trong ra được phân chia thành các nhiệm vụ:

- Phát triển ý tưởng (Concept Development)
- Phát triển hệ thống (System Development)
- Cải tiến hệ thống (System Enhencement)
- Bảo trì hệ thống (System Maintenance)

5.2. Các pha trong mô hình

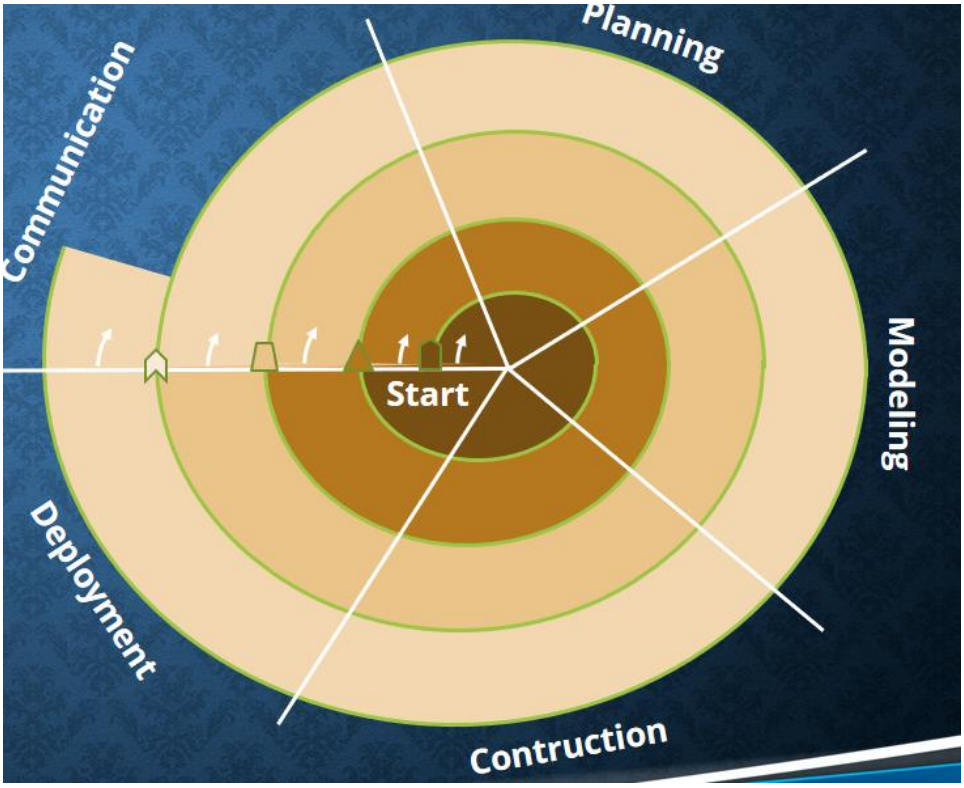
Các pha (các vùng làm việc) được phân chia thành các cung. Mỗi một biến thể của mô hình xoắn ốc có thể có từ 3 đến 6 vùng.



Hình 2.5 Mô hình xoắn ốc gồm 6 vùng làm việc.

Hình 2.5 minh họa về mô hình xoắn ốc với 6 vùng làm việc:

- Liên hệ, giao tiếp với khách hàng: giữa người phát triển và khách hàng để tìm hiểu yêu cầu, ý kiến
- Lập kế hoạch: ước tính; lập kế hoạch;
- Phân tích rủi ro;
- Thực thi kỹ thuật: Xây dựng một/một số biểu diễn của ứng dụng
- Xây dựng và phát hành: Code, test
- Đánh giá hệ thống: Khách hàng tham gia đánh giá hệ thống

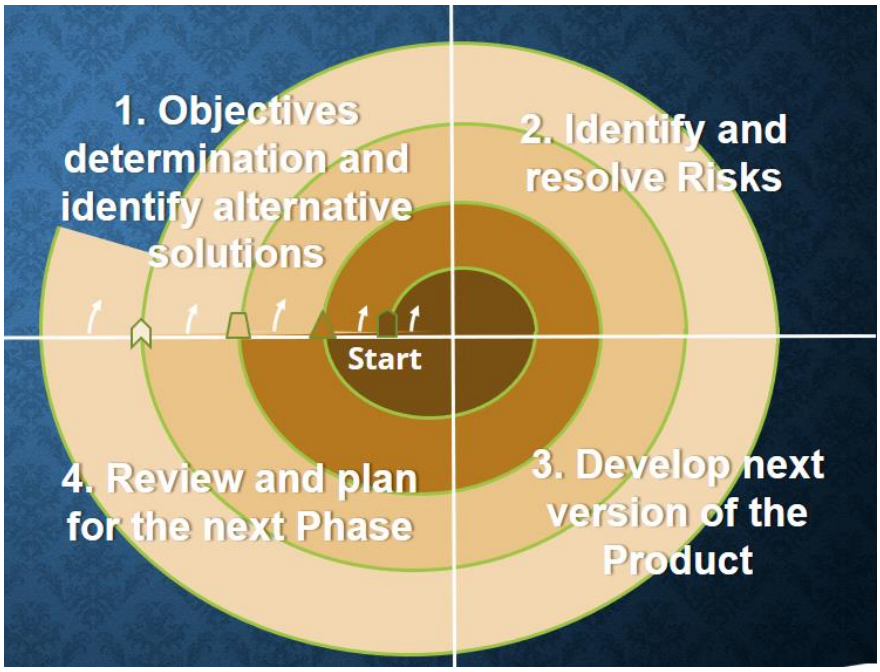


Hình 2.6 minh họa về mô hình xoắn ốc với 5 vùng làm việc.

Hình 2.6 là một minh họa về mô hình xoắn ốc với 5 vùng làm việc:

- Liên hệ, giao tiếp với khách hàng: giữa người phát triển và khách hàng để tìm hiểu yêu cầu, ý kiến
- Lập kế hoạch: ước tính; lập kế hoạch; phân tích rủi ro
- Mô hình hóa: phân tích; thiết kế

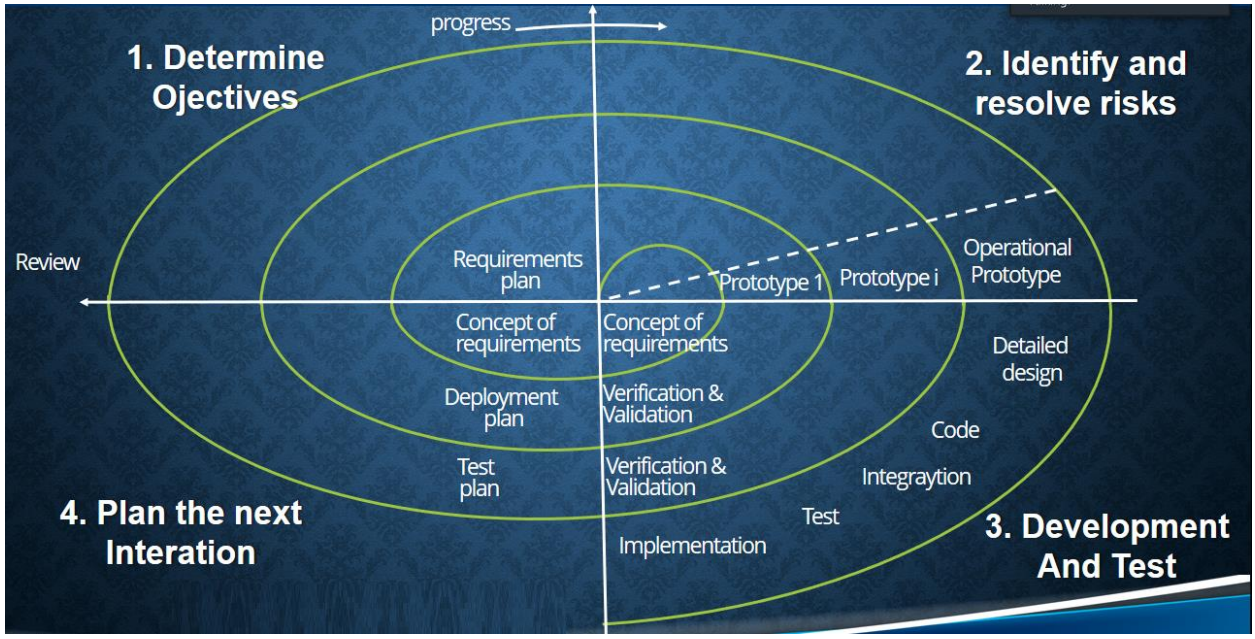
- Xây dựng và phát hành: Code, test
- Triển khai: giao hàng; Phản hồi



Hình 2.7 minh họa về mô hình xoắn ốc với 4 vùng làm việc.

Hình 2.7 và Hình 2.8 là minh họa về mô hình xoắn ốc với 4 vùng làm việc:

- Xác định mục tiêu và xác định các giải pháp thay thế: Các yêu cầu được thu thập từ khách hàng và các mục tiêu được xác định, xây dựng và phân tích khi bắt đầu mọi giai đoạn. Sau đó, các giải pháp thay thế có thể cho giai đoạn được đề xuất trong góc phần tư này.
- Xác định và giải quyết Rủi ro: Trong góc phần tư thứ hai, tất cả các giải pháp khả thi được đánh giá để chọn ra giải pháp tốt nhất có thể..
- Phát triển phiên bản tiếp theo của Sản phẩm: Trong góc phần tư thứ ba, các tính năng đã xác định được phát triển và xác minh thông qua thử nghiệm. Ở cuối góc phần tư thứ ba, phiên bản tiếp theo của phần mềm đã có sẵn.
- Xem xét và lập kế hoạch cho Giai đoạn tiếp theo: Trong góc phần tư thứ tư, Khách hàng đánh giá phiên bản đã phát triển cho đến nay của phần mềm. Cuối cùng, lập kế hoạch cho giai đoạn tiếp theo được bắt đầu.



Hình 2.8 minh họa về các bước trong mô hình xoắn ốc với 4 vùng làm việc.

5.3. Đánh giá

Spiral chú trọng vào phân tích rủi ro dự án: Rủi ro là bất kỳ tình huống bất lợi nào có thể ảnh hưởng đến việc hoàn thành thành công một dự án phần mềm. Tính năng quan trọng nhất của mô hình xoắn ốc là xử lý những rủi ro chưa biết này sau khi dự án đã bắt đầu. Việc giải quyết rủi ro như vậy được thực hiện dễ dàng hơn bằng cách phát triển một mẫu thử nghiệm. Mô hình xoắn ốc hỗ trợ đối phó với rủi ro bằng cách cung cấp phạm vi để xây dựng một nguyên mẫu ở mọi giai đoạn phát triển phần mềm.

Mỗi giai đoạn trong mô hình được bắt đầu với yêu cầu/mục tiêu thiết kế và kết thúc với việc khách hàng kiểm tra tiến độ của từng giai đoạn.

Mô hình Nguyên mẫu cũng hỗ trợ xử lý rủi ro, nhưng rủi ro phải được xác định hoàn toàn trước khi bắt đầu công việc phát triển của dự án. Nhưng trong thực tế, rủi ro của dự án có thể xảy ra sau khi công việc phát triển bắt đầu, trong trường hợp đó, chúng ta không thể sử dụng Mô hình tạo mẫu. Trong mỗi giai đoạn của Mô hình xoắn ốc, các tính năng của sản phẩm được xác định ngày tháng và phân tích, cũng như các rủi ro tại thời điểm đó được xác định và được giải quyết thông qua quá trình tạo mẫu. Do đó, mô hình này linh hoạt hơn nhiều so với các mô hình SDLC khác.

Mô hình sử dụng prototyping như một cơ chế giảm rủi ro và cho phép phát triển các prototype ở bất kỳ giai đoạn nào của quá trình phát triển.

Mỗi giai đoạn trong mô hình được bắt đầu với yêu cầu/mục tiêu thiết kế và kết thúc với việc khách hàng kiểm tra tiến độ của từng giai đoạn.

Mô hình xoắn ốc được gọi là Meta model vì nó thay thế tất cả các mô hình SDLC khác.

Mô hình xoắn ốc là một cách tiếp cận thực tế để phát triển các sản phẩm phần mềm quy mô lớn bởi vì phần mềm phát triển khi quá trình tiến triển (the software evolves as the process progresses). Ngoài ra, nhà phát triển và khách hàng hiểu rõ hơn và phản ứng với các rủi ro ở mỗi cấp độ (level) phát triển.

Nó duy trì cách tiếp cận có tính hệ thống, giống như mô hình vòng đời (Life Cycle model) nhưng kết hợp nó thành một framework lặp lại và được phản ánh nhiều hơn từ thế giới thực.

Nếu được sử dụng đúng cách, mô hình này sẽ giảm thiểu rủi ro trước khi chúng trở thành vấn đề. Vì các rủi ro kỹ thuật được xem xét ở tất cả các giai đoạn.

Mô hình **Spiral** thích hợp với dự án phần mềm:

- Khi dự án có quy mô lớn.
- Khi việc đánh giá (phân tích) các chi phí và các rủi ro là quan trọng.
- Bất cứ lúc nào cũng có thể có yêu cầu thay đổi từ phía khách hàng.
- Khi dự án được yêu cầu release thường xuyên.
- Khi yêu cầu không rõ ràng và phức tạp.
- Đối với các dự án có độ rủi ro từ trung bình đến cao.
- Những người sử dụng không chắc chắn về các nhu cầu của họ.
- Các yêu cầu phần mềm phức tạp và lớn.
- Cần phát triển một dòng sản phẩm mới (*New product line*).
- Khi có các thay đổi quan trọng (cần nghiên cứu và khảo sát cẩn thận)

Ưu điểm:

- Xử lý rủi ro: Các dự án có nhiều rủi ro chưa biết xảy ra trong quá trình phát triển, trong trường hợp đó, Mô hình xoắn ốc là mô hình phát triển tốt nhất để tuân theo do phân tích rủi ro và xử lý rủi ro ở mọi giai đoạn.
- Tốt cho các dự án lớn: Nên sử dụng Mô hình xoắn ốc trong các dự án lớn và phức tạp.
- Tính linh hoạt trong Yêu cầu: Các yêu cầu thay đổi trong Yêu cầu ở giai đoạn sau có thể được kết hợp chính xác bằng cách sử dụng mô hình này.
- Sự hài lòng của khách hàng: Khách hàng có thể thấy sự phát triển của sản phẩm ở giai đoạn đầu của quá trình phát triển phần mềm và do đó, họ quen với hệ thống bằng cách sử dụng nó trước khi hoàn thành tổng sản phẩm.

Hạn chế:

- Phức tạp: Mô hình xoắn ốc phức tạp hơn nhiều so với các mô hình SDLC khác.
- Đắt: Mô hình xoắn ốc không thích hợp cho các dự án nhỏ vì nó đắt.
- Phụ thuộc quá nhiều vào Phân tích rủi ro: Việc hoàn thành dự án thành công phụ thuộc rất nhiều vào Phân tích rủi ro. Nếu không có các chuyên gia dày dặn kinh nghiệm, sẽ là một thất bại trong việc phát triển một dự án sử dụng mô hình này.
- Khó khăn trong việc quản lý thời gian: Do số lượng giai đoạn không được biết trước khi bắt đầu dự án nên việc ước tính thời gian là rất khó khăn..

6. Phương pháp phát triển phần mềm linh hoạt

6.1. Giới thiệu về phương pháp phát triển phần mềm linh hoạt

Từ đầu những năm 1990, các phương pháp **phát triển phần mềm linh hoạt** (Agile software development, còn gọi tắt là Agile) ra đời nhằm đáp ứng yêu cầu ngày càng đa dạng và phức tạp của thực tiễn phát triển phần mềm trên thế giới.

Agile là một phương pháp phát triển phần mềm linh hoạt mà chu trình của nó thể hiện ở các vòng đời con liên tiếp nhau. Kết quả trong từng vòng đời con sẽ được phát hành với một chức năng được hoàn thành

Agile là một tập hợp các phương thức phát triển lặp và tăng dần trong đó các yêu cầu và giải pháp được phát triển thông qua sự liên kết cộng tác giữa các nhóm tự quản và liên chức năng.

Agile là cách thức làm phần mềm linh hoạt để làm sao đưa sản phẩm đến tay người dùng càng nhanh càng tốt càng sớm càng tốt và được xem như là sự cải tiến so với những mô hình cũ.

Các phương pháp Agile phổ biến:

- Khung Scrum
- Phát triển phần mềm thích ứng (ASD)
- Phương pháp phát triển hệ thống động (DSDM)
- Phát triển theo hướng tính năng (FDD)
- Phát triển phần mềm tinh gọn (LSD).

Triết lý của Agile:

- Cá nhân & tương tác hơn là quy trình & công cụ;
- Cung cấp phần mềm chạy tốt hơn là bộ tài liệu hoàn chỉnh;
- Cộng tác với khách hàng, hơn là sự thương lượng trong hợp đồng;
- Thích ứng với thay đổi, hơn là tuân thủ theo kế hoạch.

12 nguyên tắc Agile trong Tuyên ngôn Agile (Agile Manifesto[i])

- Thỏa mãn yêu cầu của khách hàng - là ưu tiên hàng đầu thông qua việc chuyển giao những sản phẩm giá trị trong thời gian sớm và liên tục.
- Sẵn sàng cho những thay đổi - thậm chí những thay đổi này xuất hiện muộn. Quy trình Agile linh hoạt trong việc ứng phó với sự thay đổi từ khách hàng, gia tăng tính cạnh tranh cho khách hàng.
- Cung cấp phần mềm hoạt động được trong thời gian ngắn từ 1 vài tuần đến 1 vài tháng, với sự ưu tiên thời gian ngắn hơn.
- Cộng tác cùng làm việc: Người kinh doanh và người lập trình phải làm việc cùng nhau mỗi ngày trong suốt dự án.
- Tạo động lực làm việc: Xây dựng các dự án xung quanh cá nhân có động lực. Cho họ môi trường làm việc thuận lợi và sự hỗ trợ cần thiết. Hãy có niềm tin rằng họ sẽ làm tốt công việc của mình.
- Đối thoại trực tiếp mặt đối mặt là phương pháp hữu hiệu nhất trong việc truyền đạt thông tin.
- Phần mềm chạy được là thước đo chính của tiến độ dự án.
- Phát triển bền vững và duy trì việc phát triển liên tục. Các nhà tài trợ, người phát triển và người dùng nên có thể duy trì tốc độ không đổi vô thời hạn.
- Liên tục quan tâm đến kỹ thuật và thiết kế để tăng cường tính linh hoạt
- Đơn giản - nghệ thuật tối đa hóa số lượng công việc không làm - là điều cần thiết
- Nhóm tự tổ chức. Các kiến trúc, yêu cầu và thiết kế tốt nhất xuất hiện từ các nhóm tự tổ chức.
- Tự phản ánh thường xuyên. Trong khoảng thời gian đều đặn, nhóm phản ánh về cách trở nên hiệu quả hơn, sau đó điều chỉnh cho phù hợp.

Các dự án áp dụng được Agile Model:

- Khi dự án có những thay đổi cần thiết phải thực hiện;
- Các yêu cầu mới ít ảnh hưởng tới lịch trình dự án;
- Phát triển dự án phần mềm cần có có sự linh động.

Những lợi ích trong cách tiếp cận theo mô hình Agile:

- Khách hàng thường xuyên có cơ hội thấy và trải nghiệm thực tế sản phẩm được chuyển giao từng giai đoạn, giúp họ có những quyết định và thay đổi trong quá trình phát triển sản phẩm
- Khách hàng có nhận thức mạnh mẽ về quyền sở hữu trong quá trình làm việc trực tiếp với nhóm dự án
- Với phương pháp quản lý Agile, sản phẩm có thể chuyển giao nhanh với những tính năng hoàn thiện cơ bản
- Sự phát triển tập trung vào người dùng cuối cùng hơn, vì sự tương tác thường xuyên và trực tiếp với khách hàng trong quá trình thực hiện

Hạn chế của phương pháp Agile

- Phụ thuộc vào khách hàng: Mức độ tham gia của khách hàng rất cao đôi khi là vấn đề cho một số khách hàng – những người không thật sự hứng thú với cách tiếp cận này
- Mô hình Agile thật sự hiệu quả khi các team member hoàn toàn tập trung vào dự án
- Ảnh hưởng tới tiến độ dự án: Giao hàng đúng tiến độ và việc thường xuyên thay đổi mức độ ưu tiên, có khả năng dẫn đến một số tính năng không được chuyển giao đúng thời hạn
- Phát sinh chi phí dự án: Phát sinh thêm một số sprint nếu cần thiết và ảnh hưởng đến chi phí dự án

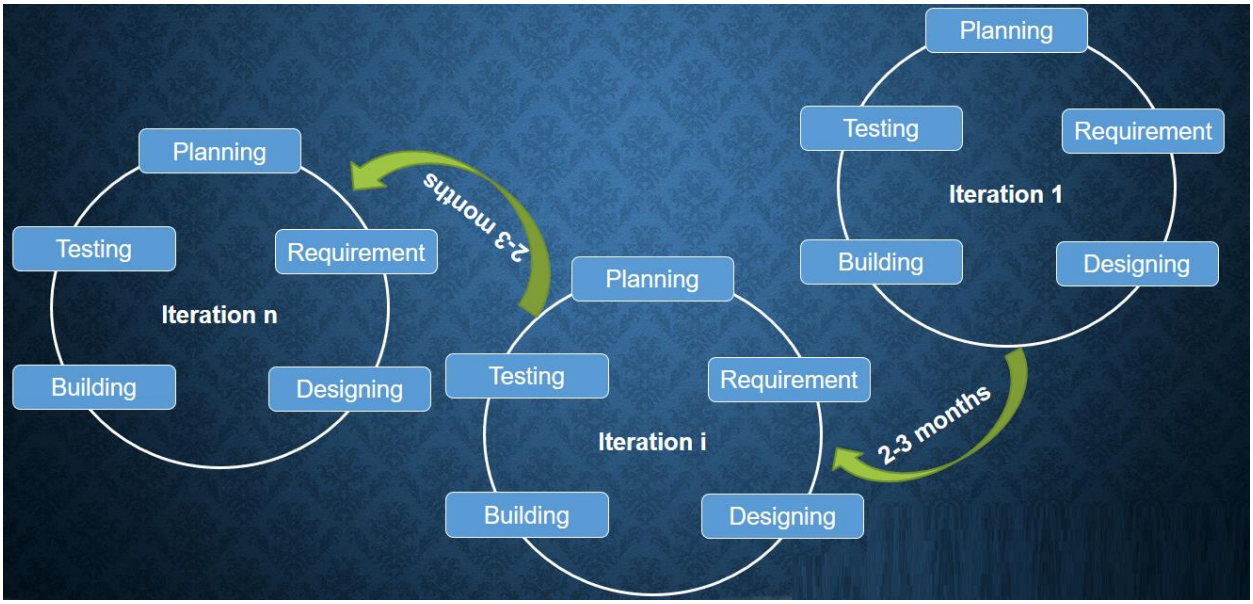
6.2. Các hoạt động của Agile

Hình 2.9 minh họa về các giai đoạn và các bước trong quy trình phát triển. Các phương pháp Agile chia sản phẩm thành các bản xây dựng gia tăng nhỏ. Các bản dựng này được cung cấp trong các lần lặp lại. Mỗi lần lặp lại thường kéo dài từ khoảng một đến ba tháng. Mỗi lần lặp lại liên quan đến các nhóm chức năng chéo làm việc đồng thời trên các lĩnh vực khác nhau như:

- Lập kế hoạch;

- Phân tích yêu cầu;
- Thiết kế;
- Mã hóa;
- Kiểm thử đơn vị
- Kiểm tra chấp nhận

Vào cuối quá trình lặp lại, sản phẩm được đưa đến cho khách hàng và các bên liên quan quan trọng.



Hình 2.9 Minh họa về các bước trong mô hình linh hoạt.

7. Khung quy trình phát triển phần mềm linh hoạt Scrum

7.1. Giới thiệu về Scrum

Khung quy trình Scrum là một mô hình làm việc rất phổ biến, được xây dựng dựa trên những nguyên tắc của Agile.

Phương pháp Scrum đơn giản hóa quy trình để phát triển phần mềm đáp ứng các nhu cầu nghiệp vụ.

Vì có cơ sở là Agile nên mô hình Scrum cũng dựa trên cơ chế lặp và sự tăng trưởng.

Những dự án áp dụng Scrum được phát triển qua một chuỗi các vòng Sprint lặp lại mỗi 1-4 tuần. Trong vòng Sprint, một hạng mục cụ thể của phần mềm sẽ được xác

định, phát triển và kiểm tra. Sau đó, hạng mục này được thêm vào (increment) và chuyển giao khi kết thúc vòng sprint.

Mô hình Scrum hoạt động dựa trên ba nguyên lý cốt lõi như sau:

- ✓ Transparency (sự minh bạch): bạn phải tường minh các artifact (tạo phẩm hoặc tài liệu) với các bên liên quan. Có vậy thì bạn mới có thể đưa ra được các quyết định đúng đắn, có lợi cho dự án.
- ✓ Inspection (sự thanh tra): bạn phải kiểm tra đều đặn, thường xuyên về tiến độ công việc và các artifact. Qua đó, bạn mới có thể phát hiện những sai sót, vấn đề tiềm ẩn khi phát triển phần mềm.
- ✓ Adaptation (sự thích nghi): bạn phải điều chỉnh quy trình và các artifact nếu xảy ra vấn đề nghiêm trọng. Đó là khi vấn đề vượt quá giới hạn cho phép hoặc khiến sản phẩm không thể được nghiệm thu.

7.2. Nhóm Scrum

Các nhóm Scrum bao gồm:

(1) Product Owner (Chủ Sản phẩm);

(2) Nhóm phát triển (Development Team);

(3) 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.

7.2.1. Product Owner

Product Owner 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, các 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;
- 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 tự mình 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 nhóm. 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.

7.2.2. Development Team

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 thế nào 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;
- 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 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.

7.2.3. 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, các kĩ thuật thực hành và các quy tắc của Scrum. Scrum Master là một lãnh đạo, nhưng cũng là người phục vụ 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 cải tiến các mối tương tác để tối đa hóa giá trị mà Nhóm Scrum tạo ra.

Scrum Master phục vụ Product Owner 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;
- Huấn luyện 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);
- Thúc đẩy các sự kiện Scrum theo yêu cầu hoặc khi cần thiết;

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;
- Giúp đỡ Nhóm Phát triển để 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;
- Huấn luyện Nhóm Phát triển trong trường hợp tổ chức chưa có hiểu biết và ứng dụng đầy đủ về Scrum;

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 (empirical product development);
- Tạo ra sự thay đổi làm tăng năng suất của Nhóm Scrum;

Làm việc với các Scrum Master khác để gia tăng hiệu quả của việc áp dụng Scrum trong tổ chức của mình.

7.3. Sprint trong Scrum

7.3.1. Gới thiệu về Sprint

Sprint trong Scrum là khoảng thời gian mà Nhóm Scrum tiến hành tất cả các hoạt động cần thiết để sản xuất được một phần tăng trưởng có khả năng chuyển giao được.

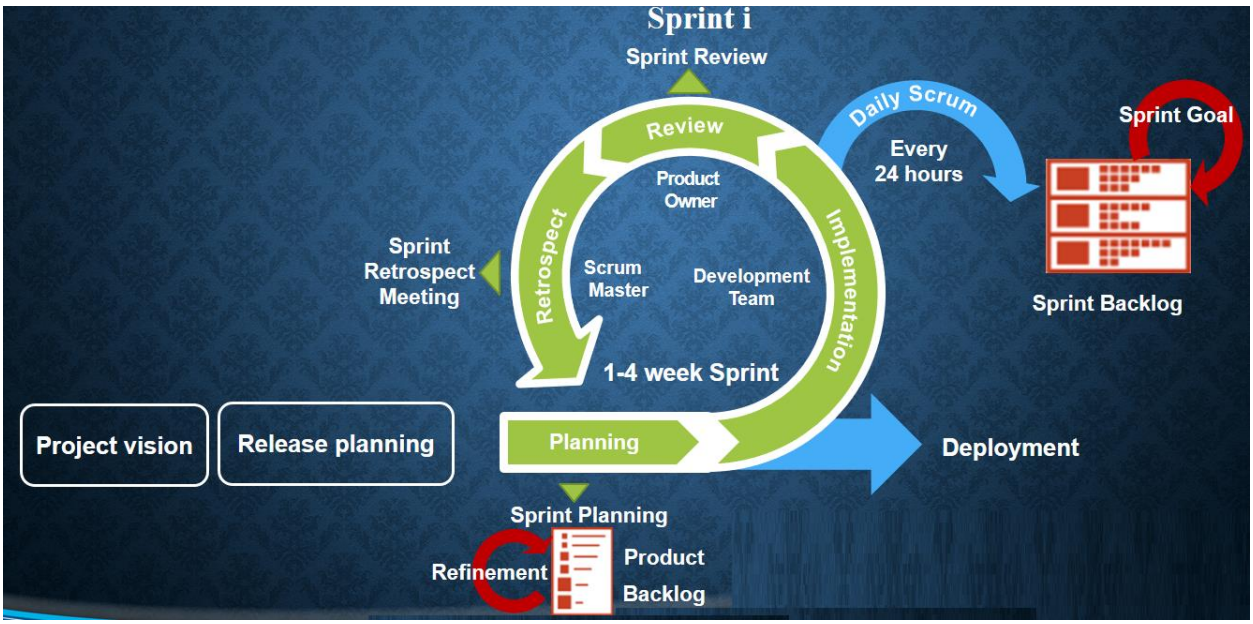
Sprint được đóng khung thời gian, có độ dài không quá một tháng và nhất quán trong suốt quá trình phát triển. Độ dài của Sprint còn phụ thuộc vào bối cảnh của dự án, đặc trưng của dự án và yêu cầu về thông tin phản hồi.

Sprint ngắn gia tăng tính thích ứng với thay đổi và giảm thiểu rủi ro nhưng tăng chi phí quản lý (thời gian cho các cuộc họp tăng lên). Các Sprint diễn ra liên tiếp nhau mà không bị gián đoạn.

Trong suốt Sprint:

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

Hình 2.10. Minh họa về các Sprint trong Scrum.



Hình 2.11. Minh họa về các hoạt động trong mỗi Sprint.

Hình 2.10 và Hình 2.11 là các minh họa về các Sprint trong mô hình Scrum.

7.3.2. Các thành phần Sprint

(1) **Vòng Sprint:** Có thể nói, các vòng sprint giống như "nhịp tim" của Scrum, là nơi biến ý tưởng thành giá trị. Các vòng sprint có độ dài tối đa khoảng một tháng và nhất quán trong suốt quá trình phát triển. Một vòng sprint mới sẽ được bắt đầu tiến hành ngay sau khi vòng sprint trước đó đã kết thúc. Các buổi họp sprint planning, daily Scrum, sprint review và sprint retrospective đều được diễn ra trong các vòng sprint. Một vòng sprint có thể bị huỷ nếu sprint goal đã lỗi thời nhưng chỉ Product Owner được hủy vòng sprint.

(2) **Sprint planning:** Đây là buổi họp mở đầu của vòng sprint, có giới hạn 8 tiếng cho vòng sprint dài một tháng. Tất cả công việc của một vòng sprint sẽ được lên kế hoạch cụ thể trong buổi họp sprint planning. Trong sprint planning, các bên liên quan sẽ phân tích và trả lời lần lượt những câu hỏi dưới đây:

- ✓ Tại sao vòng sprint này lại tạo ra giá trị? (Mục tiêu của vòng sprint này sẽ gồm những gì?)
- ✓ Có thể hoàn thành điều gì trong vòng sprint này? (Vòng sprint này sẽ phải chuyển giao những điều gì?)
- ✓ Làm thế nào để hoàn thành những việc đã chọn? (Làm sao để đạt được những điều chuyển giao đó?)

(3) **Daily Scrum:** Các buổi họp Daily Scrum thường kéo dài 15 phút dành cho các developer của Scrum team. Vào mỗi ngày của vòng sprint, một daily Scrum sẽ được tổ chức vào cùng thời gian và địa điểm. Mục đích là để kiểm tra tiến độ hoàn thành sprint goal và điều chỉnh sprint backlog nếu cần thiết. Ngoài ra, các buổi daily Scrum còn phải đưa ra được kế hoạch làm việc cho 24 giờ tiếp theo.

(4) **Sprint review:** Sprint review là buổi họp kế cuối của vòng sprint với giới hạn thời gian là 4 tiếng cho vòng sprint dài một tháng. Mục đích là kiểm tra kết quả vòng sprint và xác định những thích ứng cần thiết trong tương lai. Kết quả của sprint review là product backlog đã cập nhật cùng những công việc ở vòng sprint tiếp theo.

(5) **Sprint retrospective:** là buổi họp cuối cùng của vòng sprint, giới hạn 3 tiếng cho vòng sprint dài một tháng. Với những vòng sprint có thời hạn ngắn hơn thì buổi họp sprint retrospective cũng sẽ diễn ra ngắn hơn. Mục đích là để tổng duyệt và lập ra kế hoạch gồm những cách tăng chất lượng và hiệu quả. Mục tiêu là xác định được những cải tiến hữu hiệu nhất để triển khai ở vòng sprint tiếp theo. Nói cách khác đây là dịp để Nhóm Scrum nhìn lại quá trình làm việc của một Sprint và xác định những thay đổi cần thiết đối với quy trình để làm việc tốt hơn trong Sprint sau. Những lưu ý khi thực hiện Sprint Retrospective:

- ✓ Hoạt động cải tiến liên tục cần phải trở thành thói quen của từng cá nhân và nhóm, và lâu dần thành văn hóa của tổ chức thì sẽ đạt hiệu quả cao nhất.
- ✓ Hãy cố gắng tạo ra sự hứng thú và tinh thần tích cực trong các thành viên bằng cách động viên thông qua những công việc đã làm tốt.

(6) **Product Backlog** là một danh sách chứa tất cả những thứ cần cho sản phẩm đó. Là một danh sách được quản lý và sắp xếp thứ tự bởi Product Owner. Mỗi một Product chỉ có một Product Backlog. Mỗi một Backlog chỉ do một Product Owner quản lý. Việc sắp xếp Product Backlog sẽ giúp Scrum Team tối ưu hoá giá trị của sản phẩm phát triển qua từng Sprint. Những Product Backlog Items quan trọng và giá trị nhất thường ở vị trí trên đầu Product Backlog. Product Backlog sẽ không bao giờ kết thúc, nó tồn tại và phát triển theo sự phát triển của sản phẩm đó. Product Backlog sẽ chứa đựng Product Backlog Items (PBIs). PBIs là những công việc cần cho sản phẩm đó bao gồm: Tasks, Bugs, Requirement, Non-functional Requirement, New Feature, Technical Debt…. Những PBIs ở trên đầu Product Backlog thường sẽ được thể hiện chi tiết hơn, rõ ràng hơn, đủ nhỏ, và sẵn sàng để Scrum Team xem xét và lựa chọn cho Sprint tiếp theo ở Sprint Planning.

(7) **Sprint Backlog** là một bộ những Product Backlog Items được lựa chọn cho Sprint đó. Nó thường bao gồm: kế hoạch và những danh sách công việc dự đoán cần phải được làm. Sprint Backlog phải minh bạch, tức phải đáp ứng đủ 3 điều:

- o Ai cũng có thể dễ dàng xem.
- o Luôn được cập nhật mới nhất.
- o Và ai cũng hiểu được nó.

(8) **Sprint Backlog** được quản lý bởi Developer, và như là bản kế hoạch của họ cho Sprint đó. Sprint Backlog sẽ được update bất cứ khi nào có sự thay đổi.

7.4. Đánh giá

Phương pháp Scrum đơn giản hóa quy trình để phát triển phần mềm đáp ứng các nhu cầu nghiệp vụ.

Dưới đây là 10 lợi ích quan trọng của Scrum đối với tổ chức, đội nhóm, sản phẩm và cá nhân.

(1) Chất lượng tốt hơn: Các dự án tồn tại là để hoàn thành một mục tiêu hoặc tầm nhìn nào đó. Scrum cung cấp khung làm việc cho phép liên tục phản hồi và phát hiện sai sót nhằm đảm bảo chất lượng cao nhất có thể. Scrum giúp đảm bảo chất lượng bằng những phương pháp sau:

- ✓ Làm rõ và phát triển những yêu cầu đúng thời điểm để những hiểu biết về tính năng sản phẩm luôn được cập nhập.
- ✓ Kết hợp kiểm thử hằng ngày và phản hồi của Product Owner trong quá trình phát triển, giúp cho đội ngũ phát triển nhanh chóng phát hiện vấn đề.
- ✓ Cải thiện đầu ra (sản phẩm hoặc dịch vụ) của Nhóm Scrum thường xuyên và liên tục thông qua các buổi Sơ kết Sprint với các bên liên quan.
- ✓ Thực hiện các buổi Cải tiến Sprint nhằm giúp Nhóm Scrum liên tục cải tiến quy trình, công cụ, mối quan hệ và môi trường làm việc.
- ✓ Hoàn thành công việc sử dụng Định nghĩa Hoàn thành (DoD) cho các công đoạn phát triển, kiểm thử, tích hợp và tài liệu hóa.

(2) Giảm Thời gian Đưa ra Thị trường (time to market): Scrum đã được chứng minh là đem giá trị đến người dùng cuối nhanh hơn phương pháp truyền thống từ 30-40%. Thời gian được rút ngắn nhờ những yếu tố sau:

- ✓ Quá trình phát triển diễn ra sớm hơn do phần xây dựng tài liệu lúc đầu của dự án mô hình waterfall (thường mất hàng tháng trời) đã được thay thế bằng vai trò Product Owner trong Nhóm Scrum nhằm thực hiện các yêu cầu đúng tiến độ và cung cấp giải trình theo thời gian thực.

- ✓ Những yêu cầu ưu tiên cao được tách biệt với yêu cầu ưu tiên thấp. Việc đưa giá trị tăng dần đến người dùng cuối đồng nghĩa với việc những yêu cầu giá trị cao nhất (đồng thời rủi ro cũng cao nhất) sẽ được thực hiện trước những yêu cầu có giá trị và độ rủi ro thấp hơn. Không cần thiết phải chờ đến khi hoàn thành toàn bộ dự án mới tung sản phẩm/dịch vụ ra thị trường.
- ✓ Các chức năng hoàn chỉnh được đóng gói trong mỗi Sprint. Cuối mỗi Sprint, Nhóm Scrum cho ra gói tăng trưởng của sản phẩm/dịch vụ có khả năng chuyển giao được.

(3) Lợi nhuận từ đầu tư (ROI) cao hơn: Thời gian ra thị trường ngắn hơn là lý do chính khiến những dự án Scrum có lợi nhuận từ đầu tư (ROI) cao hơn. Thời gian ra thị trường sớm hơn dẫn tới doanh thu, các lợi ích khác sẽ có sớm hơn. Kết quả là chúng ta tích lũy sớm hơn, do vậy tổng lợi tức thu lại cao hơn theo thời gian. Đây là nguyên lý cơ bản trong tính toán giá trị hiện tại thuần (NPV – Net Present Value). Bên cạnh lợi ích về thời gian tung ra thị trường sớm, ROI của các dự án Scrum còn tăng bởi:

- ✓ Phản hồi thường xuyên thông qua các buổi Sơ kết Sprint trực tiếp từ cổ đông bao gồm khách hàng giúp cho việc sửa lỗi sai diễn ra kịp thời, qua đó giúp tiết kiệm thời gian và chi phí hơn.
- ✓ Nhờ tự động hoá và kiểm thử từ sớm nên giảm được các công việc thừa thãi, và triển khai sản phẩm nhanh hơn.
- ✓ Giảm thiểu chi phí thất bại. Nếu một dự án Scrum thất bại, nó thất bại sớm hơn và nhanh hơn so với dự án Waterfall.

(4) Khách hàng hài lòng hơn: Nhóm Scrum cam kết mang đến sản phẩm và dịch vụ khiến khách hàng hài lòng. Scrum cho phép các nhà tài trợ dự án trở nên vui vẻ hơn thông qua:

- ✓ Hợp tác với khách hàng như những cộng sự và đảm bảo sự liên kết và tham gia của khách hàng trong suốt dự án.
- ✓ Có một Product Owner là chuyên gia về yêu cầu của sản phẩm và nhu cầu của khách hàng.
- ✓ Giữ cho Product Backlog luôn được cập nhật và được ưu tiên hóa để phản ứng nhanh chóng với những thay đổi.

- ✓ Trình diễn các chức năng\tính năng hoạt động tốt với cổ đông nội bộ và khách hàng vào mỗi kì Sơ kết Sprint.
- ✓ Đưa sản phẩm đến người dùng cuối nhanh và thường xuyên hơn với nhiều lần phát hành hơn thay vì đưa ra toàn bộ sản phẩm vào phút cuối.
- ✓ Nguồn vốn cho dự án tăng dần thay vì đòi hỏi các khoản cam kết lớn ngay lúc đầu

(5) Tinh thần nhóm cao hơn: Làm việc với những người hạnh phúc và yêu thích công việc của mình có thể rất thoả mãn và vui vẻ. Tự quản khiến cho việc ra quyết định vốn là của nhà quản lý giờ đây cũng là của thành viêc Nhóm Scrum. Scrum cải thiện tinh thần của các thành viên nhóm bằng những cách sau:

- ✓ Là một phần của nhóm tự quản và tự tổ chức khiến cho thành viên trở nên sáng tạo, đột phá và được ghi nhận chuyên môn nhiều hơn.
- ✓ Các nhóm phát triển có thể tổ chức cấu trúc nhóm làm việc tùy theo các phong cách làm việc và cá tính khác nhau.
- ✓ Nhóm Scrum có thể đưa ra các quyết định phù hợp dựa trên sự cân bằng giữa công việc và đời sống cá nhân thành viên nhóm.
- ✓ Mối quan hệ ngang bằng với đại diện kinh doanh (Product Owner) ở cùng một đội giúp cân bằng các ưu tiên kinh doanh và kỹ thuật và phá bỏ rào cản trong tổ chức.
- ✓ ScrumMaster, với vai trò hỗ trợ Nhóm Scrum sẽ xoá bỏ đi những rào cản và bảo vệ Nhóm Phát triển khỏi những can thiệp từ bên ngoài.
- ✓ Tập trung vào các hoạt động bền vững đảm bảo mọi người không cảm thấy kiệt sức vì stress hoặc quá nhiều việc.
- ✓ Làm việc liên chức năng giúp tạo điều kiện cho thành viên Nhóm Phát triển học được kỹ năng mới và phát triển hơn khi hướng dẫn người khác.
- ✓ Khuyến khích cách tiếp cận lãnh đạo-phục vụ (servant- leader) nhằm hỗ trợ các thành viên trong Nhóm Scrum trong việc tự quản và chủ động tránh các phương pháp ra lệnh và kiểm soát.
- ✓ Tạo một môi trường hỗ trợ và tin tưởng lẫn nhau giúp tăng động lực và tinh thần nhân viên.

- ✓ Có những cuộc đối thoại mặt đối mặt giúp hạn chế những ảnh hưởng của giao tiếp không rõ ràng.
- ✓ Cuối cùng, Nhóm Scrum có thể cùng đồng thuận về những quy tắc để hoàn thành công việc.

(6) Gia tăng việc cộng tác và sự sở hữu: Khi Nhóm Scrum nhận trách nhiệm về một dự án hay sản phẩm nào đó, họ có thể đem lại những kết quả tuyệt vời. Nhóm Scrum hợp tác và quản lý chất lượng, quản lý hiệu quả dự án qua những cách sau:

- ✓ Xây dựng Nhóm Phát triển, Product Owner và ScrumMaster làm việc sát sao cùng nhau hằng ngày.
- ✓ Thực hiện các buổi Lập kế hoạch Sprint, cho phép Nhóm Phát triển tổ chức công việc của mình xoay quanh các ưu tiên về kinh doanh.
- ✓ Có các buổi Scrum Hằng ngày để Nhóm Phát triển có thể biết được các công việc đã hoàn thành, công việc cần làm và những khó khăn cản trở là gì.
- ✓ Thực hiện các buổi Sơ kết Sprint để Product Owner có thể vạch ra những quyết định ưu tiên của mình và Nhóm Phát triển có thể trình bày và thảo luận về sản phẩm trực tiếp với các bên liên quan.
- ✓ Thực hiện những buổi Cải tiến Sprint giúp cho thành viên Nhóm Scrum có thể rà soát lại công việc trước đây và đề xuất ra những giải pháp tốt hơn với mỗi Sprint.
- ✓ Làm việc trong môi trường mở giúp cho việc giao tiếp và phối hợp giữa các thành viên nhóm, Product Owner và ScrumMaster diễn ra nhanh hơn.
- ✓ Ra quyết định với sự đồng thuận cao.

(7) Chuẩn đánh giá chính xác hơn: Số liệu được Nhóm Scrum dùng để tính toán thời gian và chi phí, đo lường hiệu suất dự án và đưa ra các quyết định thường có liên quan và chuẩn xác hơn so với số liệu của các dự án truyền thống. Số liệu phù hợp hơn trong những dự án Scrum là vì:

- ✓ Không ai khác ngoài những người thực hiện dự án sẽ phải đánh giá nguồn lực mà dự án cần.

- ✓ Khung thời gian và ngân sách dựa trên sự phát triển thực tế của hiệu suất và khả năng của Nhóm Phát triển.
- ✓ Sử dụng những ước tính tương đối hơn là ước tính số giờ hoặc ngày để điều chỉnh các nỗ lực phù hợp hơn với khả năng và kiến thức của một Nhóm Phát triển.
- ✓ Chưa đến một phút mỗi ngày, lập trình viên có thể cập nhật biểu đồ burndown, cung cấp cái nhìn trực quan về tiến độ tiến tới mục tiêu Sprint của cả Nhóm Phát triển
- ✓ Cuối mỗi Sprint, Product Owner có thể so sánh chi phí thực của dự án (actual cost – AC) cùng chi phí cơ hội của các dự án trong tương lai (opportunity cost – OC) với giá trị mà dự án đang thực hiện lúc này mang lại (V) để biết được khi nào thì nên chấm dứt một dự án và bắt đầu một cái mới. Bạn không cần thiết phải chờ đến khi kết thúc dự án mới biết được giá trị của nó là gì.

(8) Cải thiện sự minh bạch của tiến độ dự án: Trong dự án Scrum, mỗi thành viên của đội dự án (bao gồm cả Nhóm Scrum và bên liên quan) có cơ hội tìm hiểu dự án đang được triển khai ra sao vào bất cứ thời điểm nào. Tính minh bạch và rõ ràng khiến cho Scrum trở thành một mô hình phù hợp cho các nhóm dự án tìm ra vấn đề và dự đoán tiến độ, hướng đi của dự án một cách chính xác hơn. Dự án Scrum cung cấp một cái nhìn rõ ràng về tiến độ bởi:

- ✓ Đề cao sự giao tiếp chân thật, cởi mở giữa Nhóm Scrum, các bên liên quan, khách hàng và bất cứ ai nằm trong tổ chức muốn biết về dự án.
- ✓ Scrum Hằng ngày cung cấp cái nhìn thường xuyên về tiến độ cũng như khó khăn của Nhóm Phát triển.
- ✓ Scrum Hằng ngày xung quanh bảng công việc giúp cho lập trình viên tự tổ chức và xác định công việc có độ ưu tiên cao nhất trong ngày.
- ✓ Sử dụng thông tin từ những buổi Scrum Hằng ngày, biểu đồ Sprint burn-down và bảng công việc giúp cho đội dự án theo dõi được tiến độ của từng Sprint.
- ✓ Buổi Cải tiến Sprint giúp cho thành viên Nhóm Scrum xác định được cái gì hiệu quả và cái gì không nên làm để đưa ra kế hoạch để cải tiến.

- ✓ Trình diễn kết quả của Sprint trong những buổi Sơ kết Sprint. Bất kì ai trong tổ chức đều có thể tham gia một buổi Sơ kết Sprint kể cả khi người đó là thành viên của Nhóm Scrum khác.

(9) Gia tăng sự kiểm soát với dự án: Nhóm Scrum có cơ hội lớn để kiểm soát năng suất dự án và chỉnh sửa cần thiết vì những yếu tố sau:

- ✓ Điều chỉnh ưu tiên xuyên suốt dự án tại mỗi nhịp sprint thay vì những mốc lớn giúp cho tổ chức có được dự án với thời gian và chi phí cố đinh trong khi phí ở có thể thay đổi.
- ✓ Khuyến khích sự thay đổi khiến cho đội dự án có thể phản ứng với những yếu tố bên ngoài như nhu cầu thị trường.
- ✓ Sự tương tác trong buổi Scrum Hằng ngày giúp Nhóm Scrum nhanh chóng tìm ra vấn đề và cùng nhau hoàn thành các yêu cầu.
- ✓ Sự cập nhật hằng ngày trên Sprint Backlog cho phép các biểu đồ Sprint burn-down thể hiện tiến độ của Sprint một cách chính xác, tạo điều kiện cho Nhóm Scrum thay đổi vào thời điểm họ thấy có vấn đề xảy ra.
- ✓ Giao tiếp mặt đối mặt giúp xóa bỏ rào cản giao tiếp và tìm ra các giải pháp.
- ✓ Buổi Sơ kết Sprint giúp cho các bên liên quan thấy sản phẩm đang được thực hiện, có thể đưa ra những phản hồi cho Product Owner cần dùng để dự án đi đúng hướng.
- ✓ Buổi Cải tiến Sprint giúp cho Nhóm Scrum đưa ra những điều chỉnh vào cuối mỗi Sprint để nâng cao chất lượng sản phẩm, tăng năng suất Nhóm Phát triển và làm mịn tiến độ dự án.
- ✓ Sẽ có rất nhiều cơ hội để thanh tra và thích nghi trong các dự án Scrum giúp cho tất cả thành viên đội dự án – Nhóm Phát triển, Product Owner, ScrumMaster và các bên liên quan – thực hiện việc điều chỉnh và tạo ra các sản phẩm tốt đẹp hơn.

(10) Giảm thiểu rủi ro: Scrum giúp giảm thiểu rủi ro dự án thất bại hoàn toàn bằng cách đưa ra sản phẩm hữu hình ngay từ sớm và buộc Nhóm Scrum phải thất bại từ sớm nếu họ chắc rằng sẽ thất bại thông qua những yếu tố sau:

- ✓ Hoàn thành những hạng mục backlog có độ rủi ro cao nhất trước tiên tạo ra lộ trình công việc dài nhất thông qua các vấn đề, thất bại sớm đồng thời ít tốn kém nhất.
- ✓ Triển khai dự án theo các Sprint, đảm bảo có một quãng thời gian ngắn giữa đầu tư dự án ban đầu và việc sản phẩm đó sẽ thất bại nhanh chóng hoặc là cách tiếp cận này có hiệu quả.
- ✓ Ở ngay những Sprint đầu tiên, ta thấy sản phẩm dần được hình thành nên ngay cả khi dự án bị hủy bỏ, với những yêu cầu có giá trị cao nhất và những yêu cầu rủi ro cao nhất đã được phát triển và có thể chuyển giao khách hàng nếu muốn.
- ✓ Phát triển các yêu cầu về định nghĩa hoàn thành ở mỗi Sprint thứ mà các nhà tài trợ dự án tài trợ dự án đã đưa ra, những tính năng hữu dụng, hay bất cứ điều gì có thể diễn ra trong tương lai.
- ✓ Cung cấp phản hồi liên tục về sản phẩm và tiến độ.

[i] http://agilemanifesto.org/principles.html