Software Development Models

21020055 - Trần Thùy Dung

INT2208 5 - Assignment 4th

Lời mở đầu

Mô hình phát triển phần mềm là một cách tiếp cận có hệ thống để có thể quản lý toàn bộ quá trình phát triển phần mềm. Nói cách khác, nó là một tập tất cả các pha, các giai đoạn và hoạt động trong quá trình phát triển phần mềm, trải dài từ khi lên kế hoạch cho đến tận khi bàn giao rồi bảo trì.

Nhờ áp dụng các mô hình phát triển phần mềm, dự án mới đảm bảo hoàn thành kế hoạch đúng thời gian, đạt được yêu cầu đề ra cũng như giảm thiểu lỗi phát sinh, tiết kiệm thời gian và chi phí. Và cũng bởi sự có hệ thống trong công việc phát triển phần mềm, đội ngũ phát triển, cổ đông, khách hàng mới có một ngôn ngữ, một tiếng nói chung - và đây là một điều hết sức quan trọng.

Có rất nhiều loại mô hình phát triển phần mềm khác nhau, mỗi loại đều có ưu và nhược điểm riêng, nhưng đều có chung một mục đích duy nhất - đó là tối ưu hiệu suất. Bài viết này sẽ tập trung vào 3 mô hình nổi tiếng: mô hình thác nước, mô hình chữ V và mô hình xoắn ốc. Không có một mô hình nào tốt trong mọi trường hợp, mà phải luôn biết được rằng tùy mục đích và hoàn cảnh, mô hình nào phát huy tối đa nhất, hay giảm thiểu rủi ro tốt nhất.

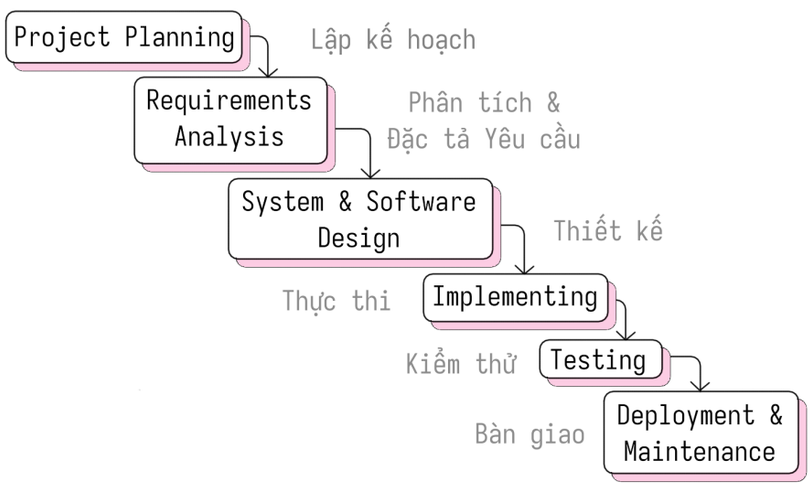
1 MÔ HÌNH THÁC NƯỚC

1.1 Giới thiệu

Công bố lần đầu của mô hình phát triển phần mềm được bắt nguồn từ mô hình kỹ thuật được dùng trong các hệ thống kỹ thuật quân sự lớn - cụ thể, trong bài báo của Winston W. Royce năm 1970.

Mô hình thác nước chia công việc phát triển phần mềm thành các pha tuần tự - hay nói cách khác, một pha chỉ được thực thi sau khi pha trước đó đã hoàn thành. Đầu ra của giai đoạn hiện tại chính là đầu vào của giai đoạn tiếp theo, được thể hiện thông qua mô hình dưới đây:

1.2 Mô hình thác nước



Hình 1: Waterfall Model

Các giai đoạn chính của mô hình phát triển phần mềm bao gồm:

- **Lập kế hoạch:** Người quản lý dự án, đội ngũ phát triển, và cổ đông hợp tác để xác định yêu cầu dự án, thiết lập các mốc thời gian chủ chốt.

- **Phân tích và đặc tả yêu cầu:** Hiểu chính xác các yêu cầu và viết tài liệu cho chúng. Lúc này, **SRS** (tài liệu đặc tả yêu cầu) được tạo ra sẽ diễn tả cụ thể hệ thống sẽ làm gì theo ngôn ngữ thông thường.
- **Thiết kế:** Hệ thống hóa và phân tích tất cả những yêu cầu trong SRS để có thể code bằng ngôn ngữ lập trình, xác định cấu trúc phần mềm từ tổng quát đến cụ thể, rồi cho ra SDD (tài liệu thiết kế).
- **Thực thi:** Mọi thông tin đã có trong tài liệu thiết kế, giờ là lúc thực hiện code.
- **Kiểm thử:** Kiểm tra xem phần mềm có thiếu sót hay lỗi cũng như đã đáp ứng yêu cầu đề ra hay chưa.
- **Bàn giao:** Sản phẩm được đưa đến cho khách hàng.

1.3 Ưu và nhược điểm

Nhược điểm	Ưu điểm
Thời gian lâu, chậm có sản phẩm.	Dễ học, dễ áp dụng.
Chi phí bỏ ra nhiều.	Đảm bảo sản phẩm **chất lượng cao**.
Chỉ hợp dự án vừa và nhỏ, có yêu cầu rõ từ đầu.	

a. Nhược điểm

Thời gian lâu, chậm có sản phẩm. Phần lớn lỗi là do người dùng không thạo. Bởi sự tuần tự của mô hình thác nước, sản phẩm đến cuối cùng mới được đem đến tay khách hàng, nên họ phải dùng ngay từ lần đầu thấy. Do đó sẽ mất thời gian và chi phí để đào tạo.

Chi phí lớn. Bởi phải đảm bảo chất lượng ở mỗi bước do không có cơ hội sửa chữa, công sức bỏ ra sẽ lớn hơn. Nếu không, chẳng may tại một pha gần cuối lại phát sinh vấn đề, thì cả dự án sẽ không còn. Nhiều chi phí sẽ bỏ ra để đề phòng những trường hợp xấu nhất.

Chỉ hợp với dự án vừa và nhỏ, có yêu cầu rõ từ đầu. Dưới góc độ nhà đầu tư và khách hàng, đã bỏ tiền ra thì dù cho dự án có lớn, họ cũng mong muốn hoàn thiện sớm. Nếu như với dự án lớn, thì không một ai có thể chờ đợi trải dài quá trình tuần tự của mô hình thác nước. Đồng thời, những dự án phức tạp rất khó để xác định rõ yêu cầu ngay từ đầu. Cho nên chung quy, mô hình này chỉ hợp với dự án nhỏ để tạo ra sản phẩm chất lượng.

b. Ưu điểm

Dễ học, dễ áp dụng. Bởi sự tuần tự, tuyến tính và định nghĩa rõ ràng, mô hình thác nước rất dễ để triển khai.

Đảm bảo sản phẩm chất lượng cao. Không chỉ chất lượng cao, mà thậm chí là hàng đầu cho dù áp dụng vào thời đại ngày nay. Bởi vì mọi pha đều phải đảm bảo chắc chắn, nên sẽ dẫn đến chất lượng tốt.

1.4 Trường hợp áp dụng

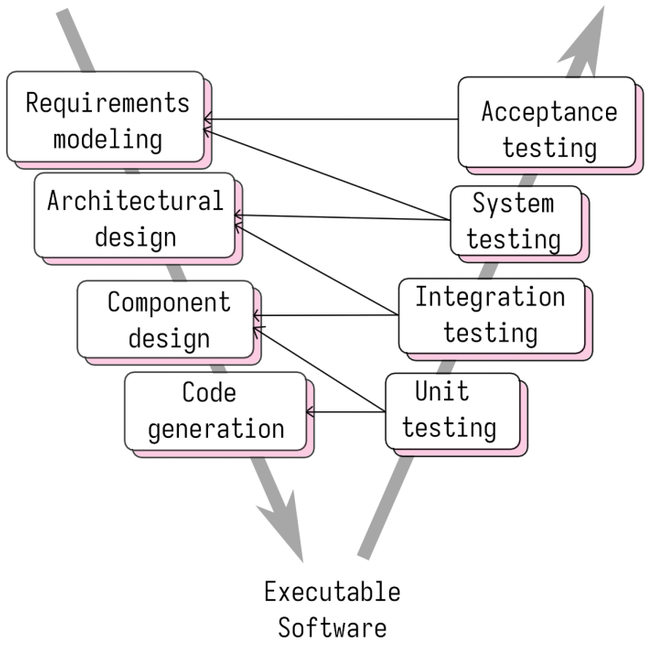
Như đã đề cập, mô hình thác nước chỉ phù hợp khi yêu cầu đã rõ ràng ngay từ đầu và ít khi có sự thay đổi.

2 MÔ HÌNH CHỮ V

1.1 Cảm hứng

Quá trình kiểm thử bao gồm 2 phần chính: sinh test và chạy test. Việc lỗi có phát hiện được ra hay không phụ thuộc vào **bộ test được sinh**. Mô hình thác nước phải trải quả nhiều bước rồi mới đến khâu kiểm thử, rồi mới sinh test. Bởi bộ test được sinh muộn, nên sẽ không cụ thể mà rất chung chung, khả năng phát hiện lỗi sẽ thấp. Cho nên mô hình chữ V được sinh ra, là cải tiến của mô hình thác nước nhưng có những điểm mạnh hơn trong khâu kiểm thử. Cụ thể, bộ test được sinh sớm ngay từ khi mỗi pha được hoàn thành, do đó sẽ đánh giá khách quan và tăng khả năng phát hiện lỗi.

1.2 Mô hình chữ V



Mô hình chữ V chia quá trình kiểm thử thành 4 mức độ:

- **Acceptance testing.**
- **System testing.**
- **Integration testing.**
- **Unit testing.**

Một chương trình bao gồm nhiều mô-đun, và mỗi mô-đun lại có nhiều hàm hay phương thức. Bởi vậy nên kiểm thử có 2 mức độ đầu tiên và thấp nhất, là **kiểm thử đơn vị** cho hàm, phương thức; và rồi **kiểm thử tích hợp** dành cho các mô-đun.

a. Unit testing

Mục đích: Đây là pha đầu tiên trong quá trình kiểm thử, để phát hiện xem phần mềm có thiếu sót hay lỗi nào trong mã nguồn hay không, có hoạt động đúng như dự tính hay không, trước khi các mô-đun được tích hợp. Trong pha này, từng mỗi một thành phần hay mô-đun được **kiểm thử độc lập**.

Phần mã nguồn được kiểm thử - hay **unit** (đơn vị) là phần nhỏ nhất có thể kiểm thử được, chẳng hạn như `method` (phương thức) hoặc `function` (một hàm), hiểu đơn giản là một khối lệnh để làm 1 hành động cụ thể. Phần code này sẽ được kiểm thử **không liên quan đến** hệ thống bên ngoài: không liên quan đến cơ sở dữ liệu, mạng, hay các tệp hệ thống cũng như cấu hình hệ thống - thay vào đó, hệ thống bên ngoài sẽ được *mô phỏng* lại cho mục đích kiểm thử. Tùy vào ứng dụng hay ngôn ngữ lập trình khác nhau, mà định nghĩa **unit** có thể khác. Chẳng hạn trong Java, đó có thể là một method trong 1 class, trong khi trong Javascript lại là một hàm trong một module. Một unit có thể là tập các hàm hay phương thức, nhưng không phải là tập nhiều unit hoặc tương tác với nhiều unit khác.

b. Integration testing

Mục đích: Sau khi các mô-đun riêng lẻ đã được kiểm thử đơn vị xong sẽ được kết hợp lại và kiểm thử theo từng nhóm. Mục tiêu của pha là để kiểm tra xem liệu các mô-đun có hoạt động với nhau một cách chính xác hay không, và liệu có vấn đề tương thích hay xung đột nào giữa chúng.

Kiểm thử tích hợp là quá trình kết hợp các mô-đun theo một cách cho trước, rồi kiểm tra tương tác giữa chúng. Phương thức làm điều này có thể là **bottom-up** (từ dưới lên) hoặc **top-down** (từ trên xuống), hay **sandwich** (kết hợp bottom-up và top-down), phụ thuộc vào việc tiếp cận từ mô-đun cấp cao hay cấp thấp trước - hoặc cả hai.

Sau khi hai pha kiểm thử đầu đã hoàn thành, lần này đi đến một mức độ cao hơn, **cả phần mềm** sẽ được kiểm thử để đảm bảo những yêu cầu và đặc tả cho trước. Cùng là kiểm thử phần mềm, và giống nhau về mặt bản chất, tuy nhiên hai pha có đối tượng khác nhau.

c. System testing

Mục đích: Tại pha này, phần mềm sẽ được kiểm thử xem liệu có thỏa mãn yêu cầu và đặc tả của developer hay không, có vấn đề gì trước khi phát hành.

Đây là kỹ thuật kiểm thử hộp đen (chỉ quan tâm chức năng chứ không phải phần code bên trong), tập trung kiểm thử dưới góc độ **đầu cuối** (từ đầu đến cuối) thay vì chỉ là chương trình (mô-đun) đơn lẻ.

Quá trình này bao gồm kiểm thử phần mềm trong những tình huống và môi trường khác nhau, bao gồm kiểm thử những khía cạnh cần thiết, chẳng hạn như chức năng, tính bảo mật, hiệu năng, hay tính khả dụng...

d. Acceptance testing

Mục đích: Cuối cùng, phần mềm sẽ được kiểm thử xem liệu có thỏa mãn yêu cầu và đặc tả của **khách hàng** hay không. Lần này, người thực hiện là khách hàng hoặc người dùng cuối, đôi khi cần đến kiểm thử ở môi trường thực, để đưa ra quyết định xem sản phẩm được đưa vào sản xuất hay không.

Quá trình kiểm thử bao gồm việc mô phỏng các tình huống thực tế để đảm bảo các tiêu chí của phần mềm, đồng thời tạo ra các trường hợp, kịch bản thử nghiệm phản ánh góc độ của người dùng cuối rồi thử nghiệm phần mềm trong những trường hợp này.

1.3 Ưu và nhược điểm

Như đã nói đến ở trên, mô hình chữ V là cải tiến của mô hình thác nước, kế thừa một số điểm mạnh và điểm yếu, nhưng đặc biệt mạnh về khâu kiểm thử.

a. Ưu điểm

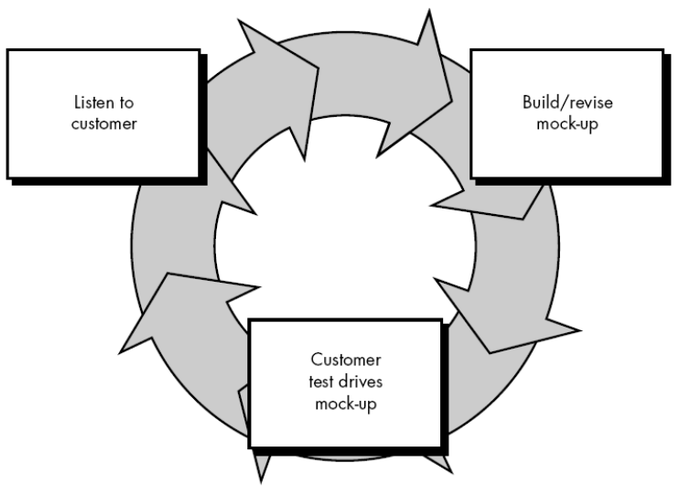
Chú tâm vào kiểm thử suốt thời gian phát triển. Giảm thiểu các vấn đề sinh ra do sinh test muộn giúp tăng chất lượng kiểm thử cũng như dự án và giảm thời gian phát triển.

b. Nhược điểm

Không linh hoạt. Vì là một biến thể của mô hình thác nước, mô hình V cũng thiếu linh hoạt và cần có sự rõ ràng trong yêu cầu và đặc tả từ đầu.

3 BẢN MẪU

Mô hình thác nước có một hạn chế, đó là không phải luôn luôn có thể xác định yêu cầu rõ ngay từ ban đầu. Cho nên bản mẫu được sinh ra, đề xuất cách để khách hàng làm việc với đội ngũ phát triển, giải quyết bài toán yêu cầu không rõ ràng. Mặc dù mô hình bản mẫu có thể áp dụng cho cả quá trình phát triển phần mềm, nhưng người ta thường sử dụng nó như **một kỹ thuật thu thập yêu cầu** hơn.



Thay vì hỏi khách hàng muốn gì, đội ngũ sẽ đưa ra bản mẫu (thử nghiệm phần mềm), tập trung vào những gì mà người dùng "thấy". Khách hàng sẽ đưa ra ý kiến của mình, từ đó đội ngũ phát triển sẽ thiết kế rồi thực thi theo, là tiền đề để áp dụng bất cứ mô hình phát triển nào sau đó. Cho nên, cả khách hàng lẫn kỹ sư phầm mềm đều ưa thích bản mẫu.

Nhược điểm

Dù tốt như vậy, bản mẫu cũng có những nhược điểm nhất định.

- **Chi phí** để làm bản mẫu.
- **Sự tham gia của khách hàng** là quan trọng, nếu không, chất lượng bản mẫu sẽ không tốt. Nhưng thực tế không dễ dàng gì để có sự tham gia của khách hàng.
- **Bảo trì.** Do vội vã triển khai để có được sản phẩm hoạt động nhanh chóng nên những công cụ tiện lợi được sử dụng ngay lập tức mà quên đi mất về hậu quả lâu dài.

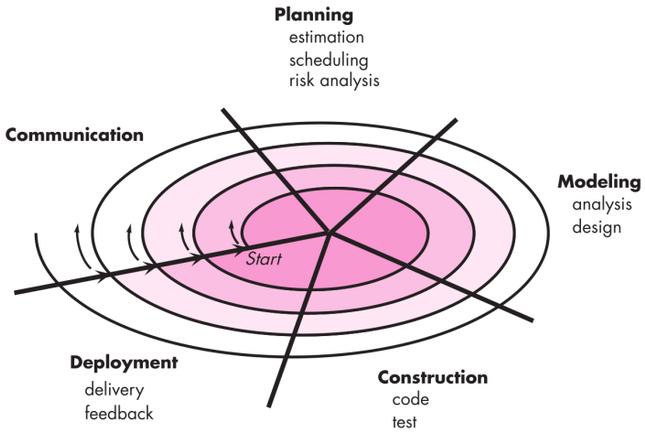
4 MÔ HÌNH XOẮN ỐC

1.1 Cảm hứng

Vậy đối với những dự án lớn thì sao? Các dự án lớn có tỉ lệ thất bại cao hơn nhiều. Một khi thất bại, thì sẽ mất trắng trong khi không tạo ra được sản phẩm. Bởi vậy cho nên, mô hình xoắn ốc được sinh ra.

1.2 Mô hình xoắn ốc

Là sự kết hợp của mô hình thác nước, bản mẫu và **phân tích rủi ro**, mô hình xoắn ốc là điển hình của mô hình phát triển tăng dần. Mỗi một vòng lặp bao gồm toàn bộ giai đoạn của một chu trình phát triển, kèm theo phân tích rủi ro và bản mẫu trước đó.



Đối với mô hình xoắn ốc, những phần quan trọng chủ chốt nhất của phầm mềm sẽ được triển khai trước, và sau đó mới phát triển những phần còn lại sau. Cũng giống như trong xây dựng, có thể xây nhà một tầng nhưng xây móng năm tầng để sau này phát triển tiếp.

Quá trình phân tích rủi ro có đầu vào là những thông tin cần thiết, và người cho ra những dự đoán rủi ro là các chuyên gia đầy kinh nghiệm - họ sẽ đưa ra hai lựa chọn là có làm điều này hay không, và nếu có thì sẽ phải đối mặt với vấn đề gì, đồng thời giải pháp cho vấn đề đó là như thế nào.

1.3 Ưu và nhược điểm

Vấn đề mấu chốt của mô hình xoắn ốc chính là **tính khả thi** của nó. Trong khi chi phí phần lớn bỏ ra cho phân tích rủi ro, không phải công ty nào cũng có chuyên gia để làm công việc đó. Bởi vậy, mô hình này chỉ có giá trị học thuật.

Đồng thời, do có rất nhiều thay đổi và lặp lại, nên mô hình này vô cùng khó đoán, khí đặt ra các cột mốc, thời gian và chi phí để hoàn thành dự án.

Kết luận

Mô hình thác nước là mô hình phát triển tuần tự rất dễ học và triển khai nhưng không phù hợp với dự án lớn bởi cứng nhắc, không linh hoạt của nó. Mô hình chữ V là cải tiến của mô hình thác nước, nhấn mạnh kiểm thử và xác thực, nhưng không thể xử lý những sự kiện đồng thời hay sự lặp lại một cách hiệu quả.

Bản mẫu là một kỹ thuật thu thập yêu cầu hiệu quả và khéo léo, tuy nhiên không phải lúc nào cũng dễ dàng có sự tham gia của khách hàng. Còn mô hình xoắn ốc nhấn mạnh quản lý rủi ro, hợp với những dự án lớn và phức tạp, nhưng có nhược điểm lớn nhất chính là sự thiếu đi chuyên gia phân tích rủi ro dẫn đến thiếu tính khả thi.

Suy cho cùng, các mô hình phát triển phần mềm là vô cùng quan trọng đối với sự phát triển của phần mềm chất lượng cao. Mỗi mô hình có điểm mạnh và điểm yếu riêng, nên cần lựa chọn mô hình phù hợp tùy theo yêu cầu dự án, đội ngũ phát triển và nhiều yếu tốt khác.