# **BTN01 - Quy trình phát triển phần mềm**

**Nhóm: 15**

**Thành viên:**

18120056 – Nguyễn Xuân Mai

18120096 – Trương Đại Triều

18120200 – Dương Bội Long

18120221 – Nguyễn Điền Thanh Phong

1. **Quy trình là gì?**

Theo ISO 9000: Quy trình được định nghĩa là “cách thức cụ thể để tiến hành một hoạt động hoặc một quá trình”.

Quy trình là một phương pháp cụ thể để thực hiện công việc.

Quy trình là trình tự (thứ tự, cách thức) thực hiện một hoạt động đã được quy định, mang tính chất bắt buộc, đáp ứng những mục tiêu cụ thể của hoạt động quản trị (quản lý và cai trị). Những hoạt động này bao gồm tất cả các dạng thức hoạt động (hoặc quá trình) trong đời sống xã hội của con người, ví dụ như các hoạt động sản xuất, kinh doanh, dịch vụ, đào tạo, nghiên cứu, tôn giáo, nghệ thuật, chiến tranh.

* Tóm lại: Quy trình là trình tự **phải** tuân theo để tiến hành một công việc nào đó.

Tại sao cần dùng quy trình?

1. Đơn giản hóa đối tượng (nhiệm vụ) phức tạp.

Khi gặp những dự án khó khăn (tất nhiên dự án dễ vẫn được), chúng ta sẽ dựa vào quy trình để thiết lập cần phải làm những việc gì và tiến độ như thế nào. Từ đó có thể chia nhỏ dự án ra và giải quyết từng phần.

1. Dễ tăng năng suất và quy mô sản xuất.

Làm theo những bước được quy định sẵn sẽ giúp làm việc một cách ăn ý hơn. Tránh những trường hợp làm khác ý mọi người làm công việc bị gián đoạn.

1. Tăng cường an toàn lao động và sức khỏe nghề nghiệp.

Dựa trên quy trình đã đề ra, chúng ta sẽ có những biện pháp cụ thể cho từng công việc. Ngược lại, khi có sự cố xảy ra, chúng ta không thể ứng phó kịp.

1. Khả năng cải tiến liên tục để tăng năng suất và chất lượng.

Đánh giá cụ thể từng giai đoạn, từ đó đưa ra biện pháp cải thiện cho giai đoạn đó. Từng giai đoạn hiện thiện sẽ đem lại một quy trình tốt.

1. Dễ tổ chức hoặc gắn kết thành dây chuyền hoạt động, dây chuyền sản xuất.

Chúng ta sẽ định được việc gì làm như thế nào, kết quả của việc đó là gì. Từ đó có thể gắn kết được nhiều dây chuyền sản xuất lại.

1. Dễ kiểm tra, giám sát và đánh giá.

Dựa vào tình hình hiện tại và quy trình đã đặt ra trước, chúng ta dễ dàng đánh giá được tiến độ và chất lượng hoàn thành.

1. Dễ ứng phó khi khủng hoảng nhân sự.

Có những tiêu chí rõ ràng, dựa vào đó dễ dàng tìm được ứng viên.

1. Giảm thiểu lãng phí của "Phương pháp thử và sai", phòng ngừa các rủi ro.

Dễ dàng thấy được, không làm việc theo quy trình dẫn đến đi sai hướng, phải bắt đầu lại từ đầu. Gây lãng phí tài nguyên của cả đội hoặc cả công ty…

1. Dễ bảo mật.

Khi làm việc theo quy trình đã đặt sẵn trước đó, mỗi phần làm xong sẽ có cơ chế bảo mật riêng. Vì tính thống nhất của quy trình nên chúng ta có thể mở khóa những bảo mật đó khi cần thiết.

1. **Quy trình phần mềm:**

**\*Khái niệm:** Một quy trình phần mềm (còn được gọi là phương pháp luận phần mềm) mà một chuỗi các hoạt động liên quan mà đưa đến việc tạo ra một sản phẩm phần mềm. Các hành động này có thể liên quan đến việc phát triển một phần mềm từ đầu, hoặc sửa đổi hệ thống hiện có.

Bất cứ phần mềm nào cũng phải đi qua 4 bước thiết kế như sau:

1. **Xác định yêu cầu kỹ thuật**: xác định các chức năng của phần mềm và những yêu cầu xung quanh nó.
2. **Thiết kế và kế thừa bổ sung phần mềm**: Phần mềm được thiết kế và lập trình.
3. **Kiểm thử phần mềm**: Phần mềm phải được xác nhận đã đạt yêu cầu và đáp ứng được nhu cầu khách hàng.
4. **Nâng cấp phần mềm** (bảo trì phần mềm): Phần mềm được chỉnh sửa và nâng cấp để đáp ứng yêu cầu khách hàng và sự thay đổi của thị trường.

**\* Tầm quan trọng của quy trình phần mềm:**

**1. Mô-đun giúp chương trình trở nên đơn giản, và đơn giản tạo sự hiệu quả:**

- Phương pháp chia một chương trình lớn thành những phiên làm việc nhỏ hơn theo yêu cầu của dự án, đống thời việc chia nhỏ dự án cũng khiến cho tiến độ được đảm bảo, và khối lượng công việc được chia nhỏ giúp dễ dàng thực hiện.

**2. Duy trì dễ dàng:**

- Các phiên làm việc nhỏ được chia từ dự án lớn giúp dễ tìm được bugs, và dễ debugs, nâng cấp các hàm,…. Một thiết kế phần mềm tốt sẽ khiến các mô-đun hoạt động gần như tách biệt nhau, nên khi có sự thay đổi ở 1 mô-đun, ta không cần phải thay đổi cả cấu trúc cả một phần mềm.

**3. Luồng chức năng và hiệu suất:**

- Một thiết kế đủ tốt sẽ thể hiện rõ được luồng chức năng khi chương trình đang chạy ngay từ khi khởi đầu đến khi xuất ra đầu ra; từ đó dễ dàng tính được hiệu suất làm việc của chương trình để có các bước nâng cấp phù hợp, trách lãng phí tài nguyên.

**4. Linh động và dễ giám sát:**

- Tính linh động thực sự hiệu quả trong trường hợp muốn thay đổi một mô-đun này bằng một mô-đun khác có thiết kế tốt hơn, khi đó, chỉ cần xác định mô-đun làm việc đó và thay đổi nó. Chính vì vậy, khả năng mở rộng phần mềm sẽ cao hơn.

- Tính giám sát giúp cho dễ dàng trong việc kiểm thử phần mềm, dễ nhìn ra được cách nó hoạt động; điều này giúp ích trong việc bảo trì phần mềm sau này.

1. **Vẽ & giải thích các “pha” (phase) trong quy trình sản xuất phần mềm *thác nước* (waterfall)**

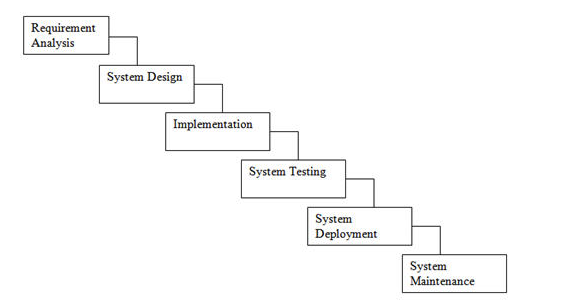
Mô hình thác nước là một ví dụ của mô hình Sequential (Tuần tự). Trong mô hình này, hoạt động phát triển phần mềm được chia thành các giai đoạn khác nhau và từng giai đoạn bao gồm hàng loạt các nhiệm vụ và có các mục tiêu khác nhau.

Mô hình Thác nước là giai đoạn đầu trong quá trình SDLC. Trên thực tế, nó là mô hình đầu tiên được sử dụng rộng rãi trong ngành công nghiệp phần mềm. Nó được chia thành các pha, đầu ra của một pha trở thành đầu vào của pha tiếp theo. Nó là giai đoạn bắt buộc được hoàn thành trước khi bắt đầu giai đoạn tiếp theo. Nói tóm lại, không có sự chồng chéo nào trong mô hình thác nước.

\*Vòng đời phát triển phần mềm (SDLC – Software Development Life Cycle) là một quá trình theo sau cho một dự án phần mềm, trong một tổ chức phần mềm. Nó bao gồm một kế hoạch chi tiết mô tả làm thế nào để phát triển, duy trì, thay đổi hoặc nâng cấp phần mềm cụ thể.

**Một vài mô hình SDLC khác: Mô hình Agile, Scrum, mô hình xoắn ốc,..**

Trong thác nước, sự phát triển của một pha chỉ bắt đầu khi giai đoạn trước hoàn thành. Do tính chất này, mỗi giai đoạn của mô hình thác nước phải được xác định khá chính xác. Các giai đoạn chuyển từ mức cao xuống mức thấp hơn, giống như một thác nước nên mô hình này được đặt tên là mô hình thác nước.



**6 pha của mô hình thác nước**

* **Phân tích yêu cầu:**

1. Nắm được tất cả các yêu cầu 2. Thảo luận và hướng dẫn để hiểu các yêu cầu, hiểu những gì cần thiết kế và chức năng, mục đích của nó là gì 3. Thực hiện thử nghiệm các yêu cầu để đảm bảo rằng các yêu cầu có thể kiểm chứng được hay không.

Sau khi đã ghi nhận các yêu cầu của khách hàng về bộ phận phân tích sẽ thực hiện làm rõ các yêu cầu và hiện thực hóa bằng “Tài liệu đặc tả“. Đây là tài liệu rất quan trọng đối với qúa trình phát triển phần mềm vì bao gồm tất cả các yêu cầu sản phẩm được thiết kế và phát triển trong suốt vòng đời dự án. Các bộ phận liên quan như lập trình, kiểm thử viên,… sẽ thực hiện công việc dựa trên mô tả các chức năng chi tiết trong tài liệu và nó sẽ trả lời câu hỏi “Phần mềm sẽ làm gì ?“.

* **Thiết kế hệ thống**

1. Theo yêu cầu, tạo ra thiết kế 2. Thảo luận về yêu cầu phần cứng /phần mềm. 3. Tài liệu thiết kế

Pha này phần lớn giải quyết các yêu cầu thiết kế về kỹ thuật, như ngôn ngữ lập trình, tầng dữ liệu, dịch vụ,…Một bản thiết kế điển hình sẽ được hoàn thành một cách càng cụ thể càng tốt. Nó sẽ mô tả chính xác logic của hệ thống được đề cập trong phần phân tích sẽ được thực thi như nào.

* **Thực hiện**

1. Theo thiết kế tạo ra các chương trình / code 2. Tích hợp code cho giai đoạn tiếp theo. 3. Unit testing

Công việc viết mã cuối cùng được thực hiện ở pha thứ tư này, nó sẽ thực thi toàn bộ các mô hình, logic của hệ thống, và các dịch vụ tích hợp đã được làm rõ trong các pha trước.

* **Thử nghiệm hệ thống**

1. Tích hợp unit tested code và kiểm tra nó để đảm bảo nó hoạt động như mong đợi. 2. Thực hiện tất cả các hoạt động thử nghiệm (Functional and non functional) để đảm bảo hệ thống đáp ứng các yêu cầu. 3. Trong trường hợp bất thường, báo cáo. 4. Theo dõi tiến độ về kiểm tra thông qua các công cụ như số liệu truy xuất nguồn gốc, ALM 5. Báo cáo hoạt động thử nghiệm

* **Triển khai Hệ thống**

1. Hãy chắc chắn rằng môi trường đang hoạt động 2. Hãy chắc chắn rằng không có lỗi mở server. 3. Đảm bảo rằng các tiêu chí test được đáp ứng. 4. Triển khai ứng dụng trong môi trường tương ứng. 5. Thực hiện kiểm tra về môi trường sau khi ứng dụng được triển khai để đảm bảo ứng dụng không gặp vấn đề

* **Bảo trì hệ thống**

1. Hãy chắc chắn rằng ứng dụng đang chạy và chạy trong môi trường tương ứng. 2. Trong trường hợp người dùng gặp lỗi, chắc chắn giải quyết và khắc phục các vấn đề. 3. Trong trường hợp bất kỳ vấn đề fixed; code cập nhật được triển khai trong môi trường. 4. Ứng dụng luôn được tăng cường để kết hợp nhiều tính năng, cập nhật môi trường với các tính năng mới nhất

Mô hình Thác nước SDLC được sử dụng khi:

* Yêu cầu ổn định và không thay đổi thường xuyên.
* Một ứng dụng nhỏ.
* Không có yêu cầu mà không hiểu hoặc không rõ ràng.
* Môi trường ổn định
* Các công cụ và công nghệ được sử dụng là ổn định
* Nguồn lực được đào tạo và sẵn sàng.

Ưu, nhược điểm của mô hình thác nước?

Ưu điểm của việc sử dụng mô hình thác nước như sau:

* Đơn giản, dễ hiểu và sử dụng.
* Đối với các dự án nhỏ hơn, mô hình thác nước hoạt động tốt và mang lại kết quả phù hợp.
* Vì các giai đoạn của mô hình thác nước cứng nhắc và chính xác, một pha được thực hiện một lần, nó rất dễ dàng để maintain.
* Các tiêu chí đầu vào và đầu ra được xác định rõ ràng, do đó nó dễ dàng và có hệ thống để tiến hành chất lượng.
* Kết quả được ghi chép tốt.

Nhược điểm của việc sử dụng mô hình thác nước:

* Không thể chấp nhận thay đổi yêu cầu
* Nó trở nên rất khó khăn để di chuyển trở lại giai đoạn. Ví dụ, nếu ứng dụng đã chuyển sang giai đoạn thử nghiệm và có thay đổi về yêu cầu, gặp khó khăn để quay lại và thay đổi nó.
* Việc giao hàng của sản phẩm cuối cùng là muộn vì không có mẫu thử nghiệm được chứng minh trung gian.
* Đối với các dự án lớn và phức tạp, mô hình này không tốt vì yếu tố rủi ro cao hơn.
* Không thích hợp cho các dự án mà yêu cầu được thay đổi thường xuyên.
* Không làm việc cho các dự án dài và đang diễn ra.
* Kể từ khi thử nghiệm được thực hiện ở giai đoạn sau, nó không cho phép xác định những thách thức và rủi ro trong giai đoạn trước đó nên chiến lược giảm thiểu rủi ro rất khó để chuẩn bị.
* Trong mô hình thác nước, điều rất quan trọng là đi theo dấu hiệu của các sản phẩm của từng giai đoạn. Tính đến ngày hôm nay hầu hết các dự án đang di chuyển với các mô hình Agile và Prototype, mô hình thác nước vẫn giữ tốt cho các dự án nhỏ hơn. Nếu yêu cầu là đơn giản và testable, mô hình thác nước sẽ mang lại kết quả tốt nhất.

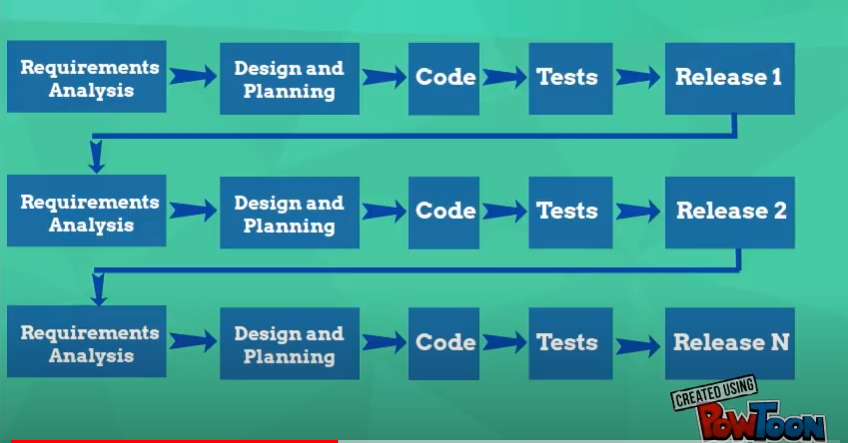
1. **Iterative and Incremental Model – Mô hình lặp & tăng trưởng**

Mô hình lặp & tăng trưởng là một quy trình phát triển phần mềm trong đó các yêu cầu được chia nhỏ thành nhiều module độc lập trong chu trình phát triển phần mềm. Mô hình là sự kết hợp cách mô hình lặp, và mô hình tăng trưởng

Mô hình là một chu trình các vòng lặp, tại mỗi vòng lặp thực hiện các module đơn giản và dễ dàng quản lý hơn. Mỗi vòng lặp trải qua **4 pha**: **phân tích yêu cầu, thiết kế, code, kiểm thử.**



Nguyên lý của mô hình này giống như việc xếp một bức tranh từ các miếng ghép, miếng ghép nào được hoàn thành trước thì sẽ cho ta được một phần bức tranh được thể hiện trước, theo thời gian, số miếng ghép được hoàn thành sẽ gia tăng và sản phầm ngày càng đi vào hoàn thiện.



**Chi tiết các pha:**

|  |  |
| --- | --- |
| Pha | Giải thích |
| Phân tích yêu cầu | Xác định yêu cầu và đặc điểm và tính năng phần mềm từ khách hàng và được tóm tắt lại trong tài liệu đặc tả yêu cầu |
| Thiết kế | Lên kế hoạch, giải pháp và kiến trúc hệ thống, những yêu cầu về phần cứng và tài nguyên. |
| Code | Tiến hành coding các tính năng của phần mềm. |
| Kiểm thử | Đàu tiên, sản phẩm được tester kiểm tra. Sau đó được chuyển giao đến khách hàng để nhận đánh giá, phản hồi, xác định lỗi trước khi sang vòng lặp mới |

**Ưu điểm.**

* Có thể sớm tạo ra nguyên mẫu của sản phẩm trong vòng đời phát triển của nó.
* Độ linh hoạt cao hơn và khi thay đổi yêu cầu dự án thì chi phí sẽ ít hơn nhiều, vì những thay đổi thuộc về module nào thì module đó sẽ thay đổi mà các module khác không hề bị ảnh hưởng.
* Việc phân chia thành các module cũng sẽ làm cho việc test nhẹ nhàng hơn, những module đơn giản thì test cũng đơn giản, sớm kết thúc.
* Giảm chi phí cho lần đầu giao sản phẩm.
* Dễ dàng quản lý các rủi ro có thể phát sinh.

**Nhược điểm***.*

* Cần phải có những khả năng thiết kế tốt và phương pháp tốt, để có thể hiểu rõ được yêu cầu và biết cách phân chi nó ra như thế nào cho hợp lý.
* Chi phí để phát triển theo phương pháp này là rất cao, cao hơn hẳn waterfall.

**Tài liệu tham khảo:**

<https://www.tutorialspoint.com/adaptive_software_development/sdlc_iterative_incremental_model.htm>

<https://en.wikipedia.org/wiki/Iterative_and_incremental_development>