Bài: Giới thiệu về Phân tích thiết kế phần mềm

Xem bài học trên website để ủng hộ Kteam: Giới thiệu về Phân tích thiết kế phần mềm

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Để đảm bảo một phần mềm được tạo ra với chất lượng tốt nhất. Dễ dàng biết được sản phẩm sẽ tạo ra trước khi bắt tay vào những đoạn code đầu tiên. Hay để dễ dàng phát triển phần mềm ở những version sau. Thì việc PHÂN TÍCH THIẾT KẾ PHẦN MỀM là cực kì quan trọng.

Chúng ta cùng nhau tìm hiểu về bộ môn **Phân tích và thiết kế hệ thống** cùng **Kteam** nhé.

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- Đã từng sử dụng qua vài phần mềm
- Đã từng suy nghĩ đến việc cấu thành của một phần mềm ra sao
- Biết sơ bộ về tư duy logic của lập trình
- Biết sử dụng máy tính cũng như các công cụ thành thạo.

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- Thế nào là phần mềm chất lượng
- Quy trình phần mềm

Thế nào là một phần mềm chất lượng

Một phần mềm chất lượng cần đảm bảo các tiêu chí sau đây:

Tính đúng đắn

- **Đầy đủ**: Đầy đủ các yêu cầu của khách hàng đưa ra. Đầy đủ yêu cầu của đặc tả mong muốn.
- **Chính xác**: Chính xác những mong muốn từ đặc tả. Không thừa. Nếu mong muốn đưa ra những tính năng khác mà bản thân cho là phù hợp. Có thể **gửi đề nghị** cho khách hàng để thảo luận.

Tính tiện dụng

- **Dễ học**: Dễ học cách sử dụng phần mềm.
- **Dễ sử dụng**: Dễ dàng sử dụng phần mềm. Không có những luồng đi thừa, phức tạp không cần thiết.
- **Giao diện trực quan**: Giao diện dễ dàng nắm bắt từ phía người dùng.
- **Tự nhiên**: Thao tác với giao diện dễ dàng và tự nhiên. Như các tab của button tuần tự nhau. Hay tiện lợi như ô nhập password. Nếu không nhập thì sẽ hiển thị nội dung là nhập password...

Tính hiệu quả

- **Tối ưu CPU**: Đảm bảo không lấy tài nguyên dư thừa dẫn đến chiếm dụng CPU. Không thao tác lặp lại thừa, xin và giải phóng bộ nhớ liên tục. Thuật toán tối ưu xử lý về tốc độ và tài nguyên.
- **Tối ưu bộ nhớ**: Đảm bảo không xin cấp phát vùng nhớ vô tội vạ. Cấu trúc lưu trữ phù hợp.
- **Tối ưu thiết bị sử dụng**: Phù hợp và mượt mà trên thiết bị chạy phần mềm. Không crash, giật lag...

Tính tương thích

- **Import/Export dữ liệu**: Có thể dễ dàng nhập xuất dữ liệu để phần mềm thao tác cũng như người dùng sử dụng.

- **Tương tác**: Đảm bảo tương tác với người dùng, hệ điều hành hợp lý. Không để xảy ra tình trạng đi ngược logic của hệ điều hành và thói quen của người dùng.

Tính tiến hóa

Là một trong các tính chất quan trọng nhất được quan tâm xem xét trong ngành *Công Nghệ Phần mềm*. Một phần mềm chỉ sử dụng được tại một thời điểm. Và không thể nâng cấp lên theo công nghệ thì đó là một phần mềm tồi.

Tính dễ kiểm tra

Việc kiểm tra các thành phần phù hợp với yêu cầu phần mềm là dễ dàng nhất có thể được.

Tính dễ sửa lỗi

Khi có sự không phù hợp so với yêu cầu trong quá trình kiểm tra một thành phần. Việc phát hiện chính xác vị trí lỗi và sửa lỗi nhanh nhất có thể được.

Tính dễ bảo trì

Khi cần nâng cấp, cải tiến một thành phần theo yêu cầu mới. Việc cập nhật phần mềm là nhanh, chính xác nhất có thể được và đặc biệt là cố gắng hạn chế ảnh hưởng đến các thành phần khác.

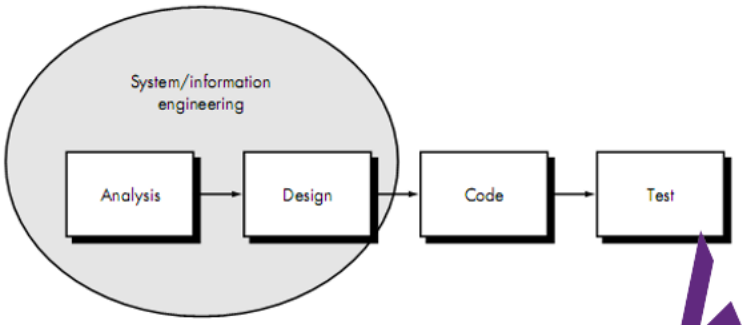
Tính tái sử dụng

Các thành phần đã thực hiện có thể dùng lại trong các phần mềm cùng lớp hoặc cùng lĩnh vực với thời gian và công sức ít nhất có thể được.

Quy trình phần mềm

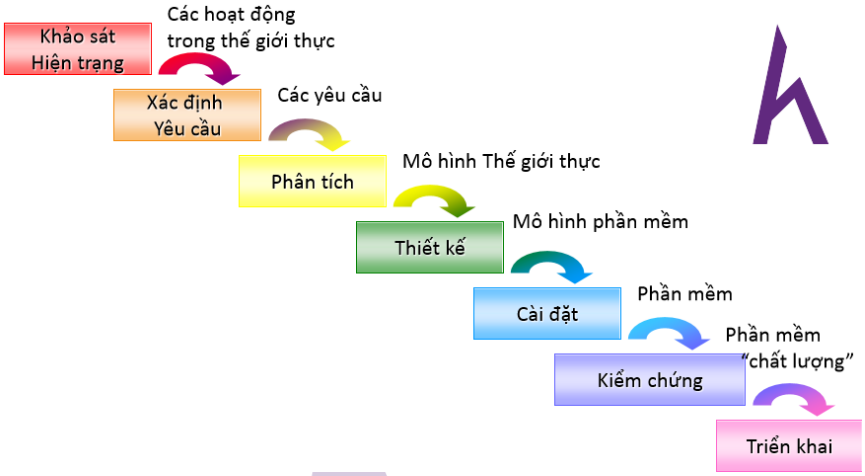
Chi phí sửa lỗi phát sinh tại giai đoạn càng sớm của quy trình phần mềm càng thấp.

Quy trình thác nước (Waterfall - Linear sequential model)



Quy trình thác nước (Linear sequential model)

(Nguồn: Roger S Pressman, Software Engineering: A Practitioner's Approach (7th Edition), McGraw-Hill, 2009)



Đi qua các giai đoạn chính:

- Khảo sát hiện trạng
- Xác định yêu cầu
- Phân tích
- Thiết kế
- Cài đặt
- Kiểm chứng
- Triển khai

> **Phải kết thúc giai đoạn rồi mới qua giai đoạn kế tiếp.**

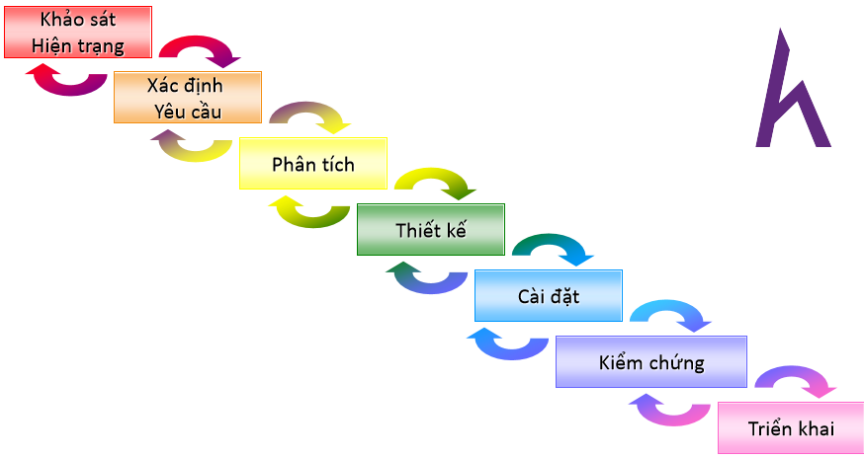
Lợi:

- Dễ dàng nắm được toàn bộ thông tin của giai đoạn trước để thực hiện giai đoạn hiện tại.
- Nhanh, gọn.

Hại:

- Khi có lỗi ở giai đoạn trước đó. Sẽ rất khó để sửa lỗi. Hay phải tốn nhiều chi phí để sửa lỗi.

Quy trình thác nước cải tiến



Đi qua các giai đoạn chính:

- Khảo sát hiện trạng
- Xác định yêu cầu
- Phân tích
- Thiết kế
- Cài đặt
- Kiểm chứng
- Triển khai

Bản chất tương tự như mô hình thác nước. Nhưng có thể quay lại bước trước đó để hoàn thiện hay sửa lỗi rồi mới tiếp tục.

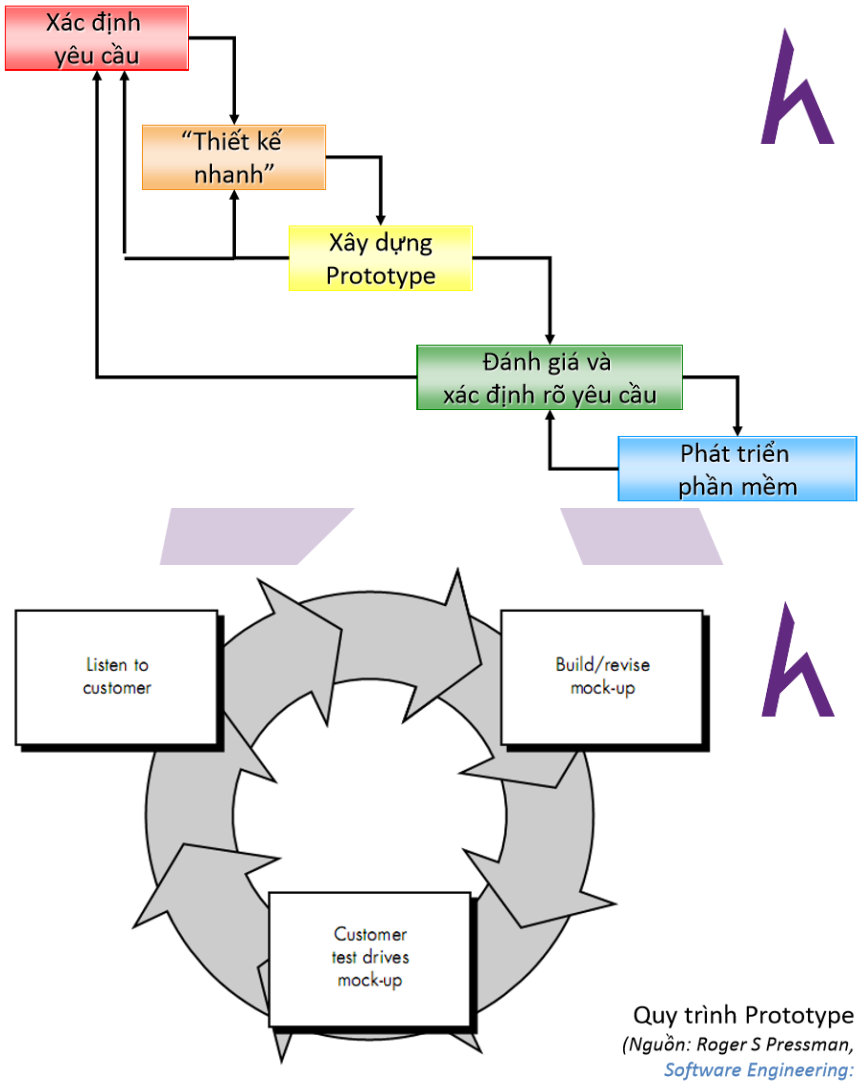
Lợi:

- Dễ dàng nắm được toàn bộ thông tin của giai đoạn trước để thực hiện giai đoạn hiện tại.
- Nhanh, gọn, giải quyết vấn đề của thác nước truyền thống là lỗi ở bước trước thì có thể được quay lại sửa.

Hại:

- Vẫn còn chưa tối ưu với những hệ thống dài hơi. Cần vừa triển khai vừa nghiên cứu.

Quy trình Prototype



Quy trình Prototype
(Nguồn: Roger S Pressman, Software Engineering: A Practitioner's Approach (7th Edition), McGraw-Hill, 2009)

Bản chất tương tự như mô hình thác nước. Nhưng sẽ có một vòng lặp tạo ra sản phẩm mẫu để đánh giá và xác định rõ yêu cầu. Khi đã đảm bảo yêu cầu sẽ bước vào phát triển phần mềm.

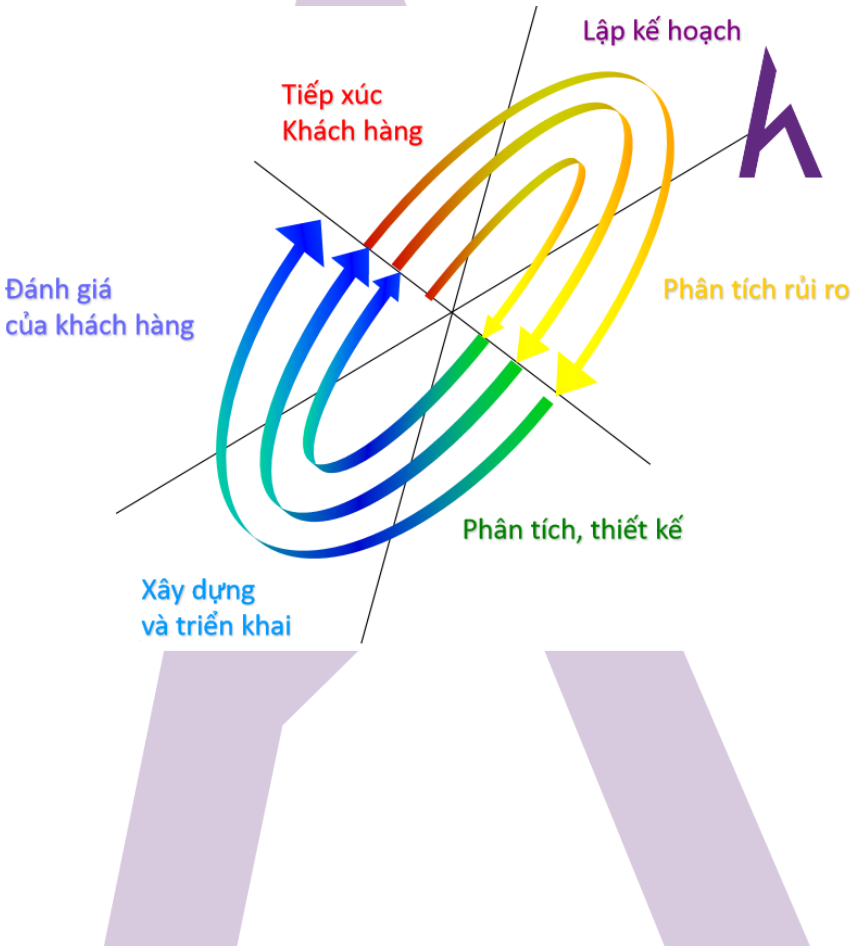
Lợi:

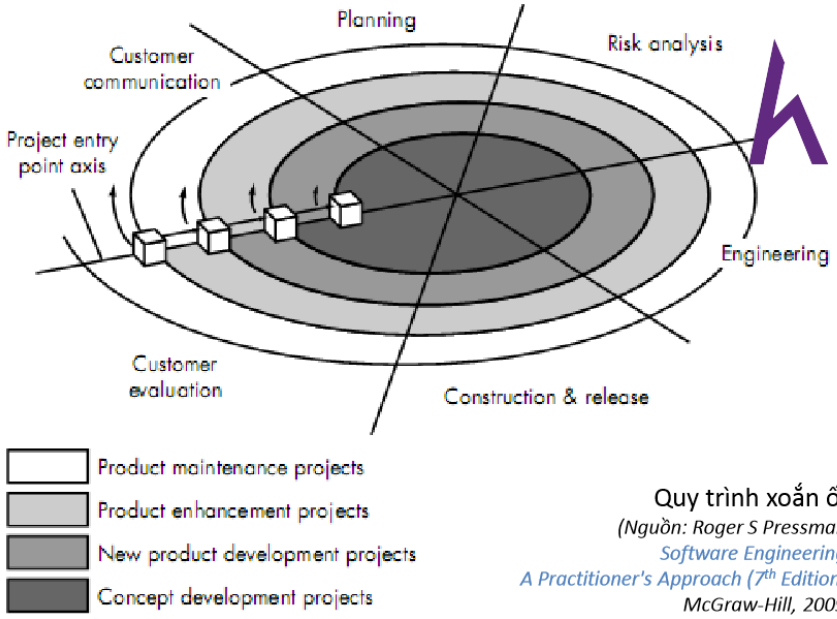
- Có sản phẩm mẫu để đánh giá và xác định yêu cầu.
- Phù hợp với những dự án vừa và nhỏ. Vẫn có thể dùng cho dự án lớn nhưng sẽ không tiện lợi bằng quy trình xoắn ốc.
- Nếu sản phẩm mẫu làm tốt có thể giúp tăng tốc độ triển khai sản phẩm chính.

Hại:

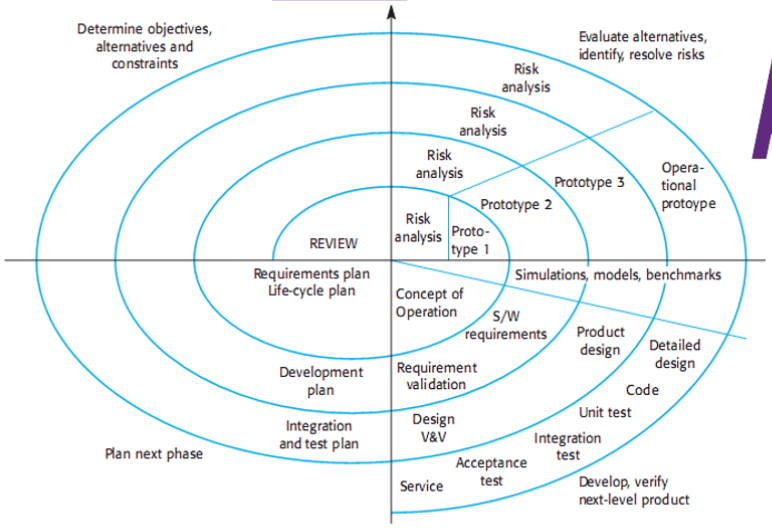
- Vẫn còn chưa tối ưu với những hệ thống dài hơi. Cần vừa triển khai vừa nghiên cứu.
- Tốn chi phí tạo sản phẩm mẫu.

Quy trình xoắn ốc





Quy trình xoắn ốc
(Nguồn: Roger S Pressman, Software Engineering: A Practitioner's Approach (7th Edition), McGraw-Hill, 2009)



Quy trình xoắn ốc của GS. Barry Boehm
(Nguồn: Ian Sommerville, Software Engineering (9th Edition), Addison Wesley, 2010)

Bản chất tương tự như mô hình Prototype. Nhưng sẽ là nhiều lần tạo ra các Prototype và sản phẩm. Mỗi lần như vậy sẽ được đánh giá hoàn thiện hay không. Và tiếp tục dựa trên sản phẩm đã hoàn thiện để thực hiện Prototype sau. Đến khi toàn bộ yêu cầu được hoàn thành. Hoặc có yêu cầu mới.

Lợi:

- Có sản phẩm mẫu để đánh giá và xác định yêu cầu. Có thể triển khai với hệ thống lớn và những dự án chưa có yêu cầu rõ ràng.
- Nếu sản phẩm mẫu làm tốt có thể giúp tăng tốc độ triển khai sản phẩm chính.

Hại:

- Vẫn còn chưa tối ưu với những hệ thống dài hơi. Cần vừa triển khai vừa nghiên cứu.
- Tốn chi phí tạo sản phẩm mẫu.
- Không phù hợp với dự án nhỏ.

Kết luận

Qua bài này các bạn đã nắm được như thế nào là sản phẩm chất lượng và những quy trình cơ bản để tạo ra một sản phẩm phần mềm.

Bài sau chúng ta sẽ cùng tìm hiểu về SƠ ĐỒ USE – CASE.

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".