Contents

A

agile process 60 (Các tiến trình linh hoạt)

Lập trình cực hạn (Extreme programming) là một trong số các mô hình mới được gọi chung là các tiến trình linh hoạt.Các tiến trình linh hoạt có đặc điểm là ít tập trung vào phân tích và thiết kế hơn đáng kể so với hầu hết các mô hình vòng đời hiện đại khác. Các tiến trình linh hoạt đòi hỏi thời gian cố định, không phải các tính năng cố định. Một đặc điểm chung khác của các tiến trình linh hoạt là tổ chức một cuộc họp ngắn vào thời gian thường xuyên mỗi ngày

analysis workflow 44 (Quy trình phân tích)

Mục đích của quy trình phân tích là phân tích và tinh chỉnh các yêu cầu để đạt được sự hiểu biết chi tiết về các yêu cầu cần thiết để phát triển một sản phẩm phần mềm một cách chính xác và dễ dàng duy trì nó. Được thực hiện trước quy trình thiết kế, đảm bảo rằng quy trình công việc thiết kế và triển khai được thực hiện một cách chính xác

architecture 49 (Kiến trúc)

Kiến trúc của một sản phẩm phần mềm bao gồm các thành phần tạo tác khác nhau và cách chúng kết hợp với nhau.Kiến trúc của một sản phẩm phần mềm được phát triển bằng cách sử dụng mô hình lặp lại và tăng dần phải có thuộc tính mà nó có thể được mở rộng liên tục (và, nếu cần, có thể dễ dàng thay đổi) để kết hợp phần gia tăng tiếp theo.Kiến trúc càng mạnh thì phần mềm càng có khả năng thay đổi cao. Không thể thiết kế một kiến trúc có thể đương đầu với quá nhiều thay đổi mạnh mẽ.

artifact 41 (Tạo tác)

artifact là 1 trong những loại sản phẩm hữu hình được tạo ra trong quá trình phát triển phần mềm. VD use case, sơ đồ lớp... Mỗi artifact mô tả chức năng kiến trúc thiết kế của phần mềm. Mỗi artifact khác nhau so với quá trình phát triển riêng của nó như kế hoạch dự án, business case

B

baseline 41 (Đường cơ sở)

baseline là một tập hợp hoàn chỉnh các artifact (một artifact là một thành phần cấu thành của một sản phẩm phần mềm). Có 4 đường cơ sở, đường cơ sở đầu tiên là bộ hiện vật ban đầu. Đường cơ sở thứ hai phản ánh việc thực hiện ,cùng với các yêu cầu, phân tích và thiết kế không thay đổi. Đường cơ sở thứ ba giống với đường cơ sở đầu tiên nhưng thiết kế và triển khai đã thay đổi. Đường cơ sở thứ tư là toàn bộ các artifact mới.

C

code-and-fix life-cycle model 52 (mô hình vòng đời xây và sửa)

Mô hình vòng đời xây và sửa là mô hình mà sản phẩm được thực hiện mà không có yêu cầu hoặc thông số kỹ thuật, hoặc bất kỳ nỗ lực nào trong thiết kế. Thay vào đó, các nhà phát triển chỉ cần ném mã lại với nhau và làm lại nó nhiều lần nếu cần để làm hài lòng khách hàng.Chỉ phù hợp với những bài tập lập trình ngắn nhưng mô hình xây và sửa hoàn toàn không đạt yêu cầu đối với các sản phẩm có kích thước hợp lý.Chi phí của phương pháp xây và sửa thực sự lớn hơn nhiều so với chi phí của một sản phẩm được thiết kế tỉ mỉ và cụ thể. Ngoài ra việc bảo trì một sản phẩm có thể cực kỳ khó khăn nếu không có tài liệu kỹ thuật hoặc thiết kế, và khả năng xảy ra lỗi hồi quy là lớn hơn đáng kể

core group 57 (Nhóm cốt lõi):

một nhóm cốt lõi gồm những người bảo trì chuyên dụng chịu trách nhiệm quản lý dự án mã nguồn mở. Một số thành viên của nhóm ngoại vi, tức là những người dùng không phải là thành viên của nhóm cốt lõi, thỉnh thoảng chọn gửi báo cáo lỗi. Các thành viên của nhóm cốt lõi có trách nhiệm đảm bảo rằng những khiếm khuyết này được sửa chữa.

core workflow 44 (quy trình làm việc cốt lõi):

Có năm quy trình làm việc cốt lõi, quy trình làm việc yêu cầu, quy trình phân tích, quy trình thiết kế, quy trình thực hiện và quy trình kiểm tra, tất cả năm quy trình này đều được thực hiện trong vòng đời của một sản phẩm phần mềm. Tuy nhiên, đôi khi một quy trình làm việc chiếm ưu thế hơn bốn quy trình còn lại**.**

D

design workflow 44 (quy trình thiết kế):

có mục đích là Mịn hóa và mô hình hóa kết quả pha phân tích cho đến khi có thể code được từng modul trên một ngôn ngữ lập trình tương ứng. Kết quả cần đạt được đó chính là: Bản mẫu các lớp, thuộc tính và phương thức + thuật toán xử lí trong các phương thức để có thể cài đặt được ngay.

E

evolution-tree life-cycle model 40 (mô hình vòng đời cây tiến hóa):

Mô hình cây tiến hóa là một ví dụ về mô hình vòng đời , nghĩa là một chuỗi các bước được thực hiện trong khi sản phẩm phần mềm được phát triển và duy trì. Ở mô hình vòng đời cây tiến hóa, Các ô ngoài cùng bên trái đại diện cho Tập 1.Tiếp theo lần lượt là các yêu cầu , phân tích , thiết kế và thực hiện.Trong tập 3, thiết kế đã được thay đổi. Đặc biệt, một thuật toán nhận dạng hình ảnh nhanh hơn đã được sử dụng. Thiết kế được sửa đổi (Thiết kế 3)

dẫn đến việc triển khai bản sửa đổi.Cuối cùng, trong Tập 4, các yêu cầu đã được thay đổi (Yêu cầu 4) để tăng độ chính xác. Điều này dẫn đến các thông số kỹ thuật được sửa đổi , thiết kế sửa đổi và triển khai sửa đổi .

extreme programming 59 (Lập trình cực đoan)

là một cách tiếp cận mới gây tranh cãi trong việc phát triển phần mềm dựa trên mô hình lặp đi lặp lại và tăng dần. Bước đầu tiên là nhóm phát triển phần mềm xác định các tính năng (câu chuyện) khác nhau mà khách hàng muốn sản phẩm hỗ trợ. Đối với mỗi tính năng như vậy, nhóm sẽ thông báo cho khách hàng thời gian triển khai tính năng đó và chi phí sẽ là bao nhiêu. Khách hàng chọn các tính năng được đưa vào mỗi bản dựng kế tiếp bằng cách sử dụng phân tích chi phí - lợi ích, trên cơ sở thời lượng và ước tính chi phí do nhóm phát triển cung cấp cũng như lợi ích tiềm năng của tính năng đối với doanh nghiệp của họ

F

failure report 57

(*Báo cáo thất bại*) Là báo cáo về hành vi không chính xác quan sát được.

Phần mềm mã nguồn đóng được duy trì và kiểm tra bởi nhóm nhân viên của tổ chức sở hữu phần mềm. Đôi khi, người dùng gửi những báo cáo lỗi. Tuy nhiên, những điều đó bị hạn chế bởi failure reports (*Báo cáo thất bại*).

Ngược lại, phần mềm mã nguồn mở, tất cả người dùng đều có quyền truy cập vào mã nguồn và được khuyến khích gửi báo cáo lỗi. Dù vậy, chỉ một ít người có khuynh hướng và thời gian, cũng như các kỹ năng cần thiết, để sử dụng mã nguồn và gửi *fault report* - báo cáo lỗi (“bản sửa lỗi”); do đó hầu hết các báo cáo lỗi là báo cáo thất bại.

fault report 57

(*Báo cáo lỗi*) Là báo cáo mô tả nơi mã nguồn bị sai và cách để làm nó đúng.

Trong phần mềm mã nguồn đóng, người dùng không truy cập được vào mã nguồn, nên họ không thể gửi *fault reports*.

Ngược lại, phần mềm mã nguồn mở, tất cả người dùng đều có quyền truy cập vào mã nguồn và được khuyến khích gửi báo cáo lỗi. Dù vậy, chỉ một ít người có khuynh hướng và thời gian, cũng như các kỹ năng cần thiết, để sử dụng mã nguồn và gửi *fault report*.

feature creep 43(Tính năng creep - tạm dịch: rủi ro về tính năng)

Là một sự kế thừa nhỏ, gần như không đáng kể, bổ sung cho các yêu cầu. Nhưng những thay đổi thường xuyên, bất kể có nhỏ như thế nào, đều ảnh hưởng không tốt đến sức khỏe của phần mềm. Việc mở rộng hoặc bổ sung quá mức các tính năng mới trong một sản phẩm, vượt ra ngoài chức năng cơ bản của sản phẩm và có thể dẫn đến phần mềm bị phình ra và quá phức tạp, thay vì thiết kế đơn giản.

Nguyên nhân là do trong một số trường hợp, vì mục tiêu thay đổi, người quản lý liên tục thay đổi ý định về chức năng của sản phẩm phần mềm đang được phát triển.

freeze 62 (Đóng băng)

Là không có thay đổi nào khác được thực hiện đối với các thông số kỹ thuật. Điều này xảy ra trong mô hình Đồng bộ và Ổn định, Ổn định được thực hiện ở cuối mỗi bản dựng; bất kỳ lỗi nào được phát hiện đều được sửa và đóng băng.

I

implementation workflow 44(*Quy trình thực hiện*)

Là 1 trong 5 quy trình làm việc cốt lõi, bao gồm: quy trình yêu cầu, quy trình phân tích, quy trình thiết kế, quy trình thực hiện và quy trình kiểm tra.

(*Dựa vào nội dung trong chương 3, trang 83*) Mục tiêu của quy trình thực hiện là thực hiện sản phẩm phần mềm mục tiêu bằng (các) ngôn ngữ triển khai đã chọn. Một sản phẩm phần mềm nhỏ đôi khi được nhà thiết kế cố vấn. Ngược lại, một sản phẩm phần mềm lớn được phân chia thành các hệ thống con nhỏ hơn, sau đó được thực hiện song song bởi các nhóm viết mã.

incrementation 44 **(Sự gia tăng)**

Là 1 trong 2 khía cạnh nội tại của kỹ thuật phần mềm (cùng với Iteration - Sự lặp lại); phát triển phần mềm gia tăng đã hơn 45 tuổi.

Ví dụ về quá trình gia tăng: có thể xây dựng một requirements document (tài liệu yêu cầu) bằng cách xem xét bảy yêu cầu mà được cho là quan trọng nhất. Sau đó, sẽ xem xét bảy yêu cầu quan trọng nhất tiếp theo, và cứ như vậy.

Iteration(lặp)

là sự lặp lại của một hàm hoặc thủ tục trong một chương trình máy tính. Lặp đi lặp lại các chức năng rất phổ biến trong lập trình máy tính, vì chúng cho phép nhiều khối dữ liệu được xử lý theo thứ tự

Là 1 khía cạnh bên trong của kỹ nghệ phần mềm và các mô hình vòng đời lặp đi lặp lại đã được được sử dụng để đảm bảo chất lượng phần mềm không có lỗi

iterative-and-incremental life-cycle model 44(mô hình vòng đời lặp đi lặp lại và gia tăng)

là một phương pháp phát triển phần mềm đó được mô hình hóa xung quanh một tăng dần tính năng bổ sung và phát hành theo chu kỳ và nâng cấp pattern.Iterative và phát triển phần mềm gia tăng bắt đầu với việc lập kế hoạch và tiếp tục thông qua chu kỳ phát triển lặp đi lặp

lại liên quan đến thông tin phản hồi sử dụng liên tục và gia tăng Ngoài các tính năng kết thúc với việc triển khai các phần mềm hoàn thành vào cuối mỗi chu kỳ. Nó là một trong những phương pháp phát triển phần mềm Agile, quá trình thống nhất hợp lý và lập trình cực đoan.

L

life-cycle model 40 (mô hình vòng đời)

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.

M

Miller's Law 44 (Luật Miller)

Tại bất kỳ thời điểm nào, người ta chỉ có thể tập trung vào tối đa khoảng 7 vấn đề (đơn vị thông tin). Để xử lý các vấn đề lớn sử dụng phương pháp làm mịn từng bước, tập trung xử lý các công việc quan trọng trước.

mitigate risk 63 (Giảm thiểu rủi ro)

Ý tưởng giảm thiểu rủi ro thông qua việc sử dụng các nguyên mẫu và các phương tiện khác là ý tưởng nằm trong mô hình vòng đời xoắn ốc [Boehm, 1988]. Một cách đơn giản hơn để xem xét mô hình vòng đời này là mô hình thác nước với mỗi giai đoạn được phân tích rủi ro trước đó.Trước khi bắt đầu mỗi giai đoạn, giảm thiểu (kiểm soát) rủi ro. Nếu không thể giảm thiểu tất cả các rủi ro quan trọng trong giai đoạn đó, thì dự án sẽ bị chấm dứt ngay lập tức.

model 40

Là một thành phần giữ một nhiệm vụ cụ thể nào đó trong 1 phần mềm, một máy tính hay 1 hệ thống cụ thể. Trong lập trình thì module sẽ ở dạng codes, còn đối với một phần mềm như Software thì module được coi như một compiled. Sẽ được để dưới các dạng như .exe, .com, . dll, .

Moving-target problem (vấn đề thay đổi mục tiêu)

Nghĩa là, trong khi phần mềm đang được phát triển, thì đột nhiên thay đổi các yêu cầu

O

Open-source software (phần mềm mã nguồn mở)

là phần mềm được cá nhân có ý tưởng thiết kế sẵn một phiên bản thử nghiệm rồi chia sẻ miễn phí qua internet, để cho những người hứng thú với nó vào cùng tham gia phát triển một cách tình nguyện và không có tiền công.

P

Pair programming (Lập trình cặp)

Hai lập trình viên làm việc cùng nhau trên một máy tính thực hiện nhiệm vụ và đảm bảo rằng tất cả các trường hợp thử nghiệm hoạt động chính xác.Họ luân phiên đánh máy sau mỗi 15 hoặc 20 phút và thay nhau kiểm tra lỗi sau đó tích hợp vào phiên bản hiện tại của sản phẩm. Lập trình giúp giảm thời gian thực hiện nhiệm vụ đồng thời giúp họ học hỏi và nâng cao trình độ lẫn nhau.Tuy nhiên lập trình cặp không phải lúc nào cũng hoạt động tốt với những cá nhân nhút nhát hoặc hống hách, hoặc với hai lập trình viên thiếu kinh nghiệm.

peripheral group (nhóm ngoại vi)

tức là những người dùng không phải là thành viên của nhóm cốt lõi (core group), họ muốn tham gia nhiều hơn vào những dự án mã nguồn mở nhưng họ không chịu trách nhiệm quản lý chúng, thỉnh thoảng họ gửi báo cáo lỗi cho nhóm cốt lõi.

proof-of-concept prototype

là một mô hình thử nghiệm chỉ để kiểm tra ý tưởng chưa hoàn thiện hay một giả định thiết kế trên một trình mô phỏng. Bằng cách này, hệ thống thực tế không bị xáo trộn; và đối với chi phí chỉ triển khai (), công ty có thể phát hiện tiềm năng phát triển ý tưởng đó trong thế giới thực.

R

Rapid prototype (mẫu thử nhanh)

Một mẫu thử nhanh là một mô hình hoạt động có chức năng tương đương với một tập hợp con của sản phẩm. Ví dụ: nếu sản phẩm mục tiêu là xử lý các khoản phải trả, khoản phải thu và kho, thì mẫu thử nhanh có thể bao gồm một sản phẩm thực hiện xử lý màn hình để thu thập dữ liệu và in báo cáo, nhưng không cập nhật file hoặc xử lý lỗi. Một mẫu thử nhanh cho sản phẩm mục tiêu nhằm xác định nồng độ của enzym trong dung dịch có thể thực hiện phép tính và hiển thị câu trả lời, nhưng không thực hiện bất kỳ xác nhận hoặc kiểm tra tính hợp lý nào của dữ liệu đầu vào.

rapid-prototyping life-cycle model (mô hình bản mẫu nhanh) 55

Mô hình bản mẫu nhanh có đặc trưng là tiến hành làm bản mẫu nhanh (rapid prototype) trong pha lấy yêu cầu và các pha còn lại làm theo thứ tự tuyến tính

refactoring 60 (tái cấu trúc)

Refatoring là cải tiến và làm tốt hơn chất lượng của mã nguồn trong một ứng dụng.
Nó không làm thay đổi các chức năng chính, chức năng chung của ứng dụng,
nhưng nó làm cho ứng dụng dễ bảo trì hơn dễ phát triển hơn trong tương lai.

regression fault 43(Lỗi hồi quy)

- Là một lỗi vô tình được đưa vào một bộ phận của sản phẩm do hậu quả của việc thực hiện một thay đổi dường như không liên quan đến một bộ phận khác của sản phẩm

requirements workflow 44

- Việc xây dựng một sản phẩm phần mềm có 5 quy trình: **yêu cầu**, phân tích, thiết kế, triển khai và thử nghiệm. Trong đó **requirements workflow** là dòng công việc chính ở đầu chu kỳ sống, nhưng tầm quan trọng tương đối của nó giảm dần sau đó. Ngược lại, quy trình thực hiện và kiểm thử chiếm nhiều thời gian của các thành viên trong nhóm phát triển phần mềm vào cuối vòng đời hơn so với lúc đầu

risk 50(*Rủi ro*)

- ***Mô hình lặp đi lặp lại và tăng dần cho phép chúng tôi giảm thiểu rủi ro sớm. Rủi ro luôn liên quan đến việc phát triển và bảo trì phần mềm.*** Ví dụ, trong nghiên cứu trường hợp nhỏ của Winburg, thuật toán nhận dạng hình ảnh ban đầu không đủ nhanh; ***Có một rủi ro luôn luôn hiện hữu là một sản phẩm phần mềm đã hoàn thành sẽ không đáp ứng các hạn chế về thời gian của nó. Việc phát triển từng bước một sản phẩm phần mềm cho phép chúng tôi giảm thiểu những rủi ro như vậy sớm trong vòng đời.*** Ví dụ: giả sử một mạng cục bộ (LAN) mới đang được phát triển và có lo ngại rằng phần cứng mạng hiện tại không đủ cho sản phẩm phần mềm mới. Sau đó, một hoặc hai lần lặp đầu tiên được hướng tới việc xây dựng những phần của phần mềm giao diện với phần cứng mạng. Nếu hóa ra, trái ngược với lo ngại của các nhà phát triển, mạng có khả năng cần thiết, các nhà phát triển có thể tiến hành dự án, chắc chắn rằng rủi ro này đã được giảm thiểu. Mặt khác, nếu mạng thực sự không thể đối phó với lưu lượng bổ sung mà mạng LAN mới tạo ra, điều này sẽ được báo cáo cho khách hàng sớm trong vòng đời, khi chỉ một phần nhỏ ngân sách đã được chi tiêu. Bây giờ, khách hàng có thể quyết định hủy dự án, mở rộng khả năng của mạng hiện có, mua một mạng mới và mạnh hơn hoặc thực hiện một số hành động khác.

robustness 49

- **Độ bền của kiến trúc cơ bản** có thể được xác định tương đối sớm trong vòng đời. Kiến trúc của một sản phẩm phần mềm bao gồm các thành phần tạo tác khác nhau và cách chúng kết hợp với nhau như hướng đối tượng, đường ống và bộ lọc, máy khách-máy chủ… Kiến trúc của một sản phẩm phần mềm được phát triển bằng cách sử dụng mô hình lặp lại và tăng dần phải có thuộc tính mà nó có thể được mở rộng liên tục (và, nếu cần, có thể dễ dàng thay đổi) để kết hợp phần gia tăng tiếp theo. Khả năng xử lý các phần mở rộng và thay đổi như vậy mà không bị phá vỡ được gọi là tính mạnh mẽ.

S

spiral life-cycle model 63 (mô hình xoắn ốc)

Mô hình xoắn ốc có đặc trưng là có nhiều vòng lặp nhau, nhưng vòng lặp
sau phát triển rộng hơn vòng trước,
Mỗi pha của mỗi lần lặp:

- Bắt đầu bằng việc quyết định và phân tích rủi ro
- Kết thúc bằng việc đánh giá lỗi và lập kế hoạc cho pha tiếp theo
- Nếu các rủi ro đều không xử lí được thì dừng lại ngay lập tức

stabilize 62 (Ổn định)

- Ổn định được thực hiện ở cuối mỗi bản dựng. Bất kì lỗi còn lại nào được tìm thấy đều được sửa chữa và họ đóng băng bản dựng, nghĩa là sẽ không có thay đổi nào khác đối với các thông số kỹ thuật

stand-up meeting 60 (Cuộc họp trực tiếp)

- Cuộc họp trực tiếp là cuộc họp trong đó những người tham dự thường tham gia khi đang đứng. Mục đích của cuộc họp trực tiếp là để nêu ra vấn đề chứ không phải giải quyết chúng

stepwise refinement 44 (Tinh chỉnh theo từng bước)

- Là tập trung vào những khía cạnh hiện đang là quan trọng nhất và hoãn lại cho về sau những khía cạnh hiện ít quan trọng hơn. Nói cách khác, mọi khía cạnh đều cuối cùng cũng đều được xử lý nhưng theo độ quan trọng.

story 59:

-STORY là Text Files - Storyist Document, dưới định dạng Binary được phát triển bởi Storyist Software.

Tài liệu được tạo ra bởi Storyist, một công cụ câu chuyện viết tạo cho tiểu thuyết gia và biên kịch; tiết kiệm văn bản bằng văn bản cũng như thông tin bố trí, ghi chú và thông tin chương phác thảo; giúp tác giả theo dõi cốt truyện, nhân vật, và các thông tin thiết lập cùng với kịch bản.

synchronize 62 (đồng bộ):

-Sync là một thuật ngữ ám chỉ sự đồng bộ dữ liệu, cụ thể là quá trình thực hiện trao đổi cũng như đồng bộ thông tin giữa 2 nguồn dữ liệu theo trình tự thời gian. Nói một cách dễ hiểu, đồng bộ hóa chính là sự trao đổi, lưu trữ thông tin trên một thiết bị khác so với máy chủ.

synchronize-and-stabilize (đồng bộ hóa và ổn định):

-cân bằng giữa tính linh hoạt và cấu trúc trong phát triển sản phẩm phần mềm. Phương pháp này liên quan đến việc đồng bộ hóa liên tục hoạt động của mọi người với tư cách là các cá nhân làm việc trong một dự án và là thành viên của các nhóm song song

(đọc thêm để hiểu): Giai đoạn phân tích yêu cầu được thực hiện bằng cách phỏng vấn nhiều khách hàng tiềm năng

cho gói và trích xuất danh sách các tính năng có mức ưu tiên cao nhất cho các máy khách. Một cụ thể ca-tài liệu tion hiện đã được soạn thảo. Tiếp theo, tác phẩm được chia thành ba hoặc bốn lần xây dựng. Lần đầu tiên bản dựng bao gồm các tính năng quan trọng nhất, bản dựng thứ hai bao gồm các tính năng quan trọng nhất tiếp theo các tính năng, v.v. Mỗi công trình được thực hiện bởi một số nhóm nhỏ làm việc song song. Vào cuối mỗi ngày, tất cả các nhóm đồng bộ hóa; nghĩa là, họ đã hoàn thành một phần các thành phần với nhau và kiểm tra và gỡ lỗi sản phẩm kết quả. Ổn định được thực hiện tại cuối mỗi bản dựng. Bất kỳ lỗi nào còn lại đã được phát hiện cho đến nay đều là lỗi, và bây giờ họ đóng băng công trình; nghĩa là, sẽ không có thay đổi nào được thực hiện đối với các cation cụ thể.

T

task 59:

-một task là một đơn vị thực thi (unit of execution) hoặc một đơn vị công việc (unit of work). Khái niệm task thường mang tính chất khá chung chung, vì đối tượng chính xác được hướng đến khi dùng khái niệm task thường là tiến trình (process), tiến trình nhẹ (light-weight process), luồng (thread), bước (step), yêu cầu (request) hoặc yêu cầu truy vấn (query).

test-driven development 59:

-là một phương thức làm việc, hay một quy trình viết mã hiện đại. Lập trình viên sẽ thực hiện thông qua các bước nhỏ (BabyStep) và tiến độ được đảm bảo liên tục bằng cách viết

và chạy các bài test tự động (automated tests). Quá trình lập trình trong TDD cực kỳ chú trọng vào các bước liên tục sau:

1. Viết 1 test cho hàm mới. Đảm bảo rằng test sẽ fail.
2. Chuyển qua viết code sơ khai nhất cho hàm đó để test có thể pass.
3. Tối ưu hóa đoạn code của hàm vừa viết sao cho đảm bảo test vẫn pass và tối ưu nhất cho việc lập trình kế tiếp
4. Lặp lại cho các hàm khác từ bước 1

test workflow 44

là 1 trong 5 quy trình làm việc chính của mô hình vòng đời phát triển lặp và tăng trưởng. Đây là quy trình diễn ra xuyên suốt nhưng chiếm nhiều thời gian nhất ở chu trình tăng trưởng cuối. Nhiệm vụ chính của test workflow là kiểm tra tất cả artifact đã hoạt động tốt chưa và không gây ảnh hưởng nghiêm trọng đến các artifact liên quan đến nó.

timeboxing 60

là 1 kĩ thuật quản lí thời gian đã được sử dụng trong quy trình phát triển linh hoạt : 1 lượng thời gian cụ thể sẽ được đặt cho 1 công việc nhất định và các thành viên trong nhóm sẽ hoàn thành công việc đó tốt nhất họ có thể trong suốt khoảng thời gian đó.

W

waterfall life-cycle model 41

là 1 mô trong các mô hình phát triển phần mềm cổ điển. Đây là mô hình với các vòng lặp phản hồi tượngt ung cho quá trình bảo trì sản phẩm. Ví dụ, nếu ơr giai đoạn thiết kế họ tìm thấy 1 lỗi gây ra ở giai đoạn yêu cầu. Thành viên trong giai đoạn thiết kế sẽ phản hồi lại cho thành viên ở khâu phân tích và từ đó lại khâu phân tích sẽ phản hồi lại cho khâu yêu cầu và chỉnh sửa ở đó. Sau đó, lại di chuyển xuống khâu phân tích, sửa lại tài liệu chi tiết từ nhưng chỉnh sửa ở khâu yêu cầu và sau đó sửa lại tài liệu thiết kế.

workflow (quy trình làm việc)44

là chuỗi các nhiệm vụ để tạo ra kết quả mong muốn. Nó có thể là chuỗi các bước theo tuần tự hoặc song song xảy ra nhiều bước đồng thời.

Các giai đoạn phát triển không kết thúc khi chuyển sang giai đoạn khác, nó kéo dài liên tục trong suốt vòng đời phát triển nên do đó dùng khái niệm workflow thay vì dùng từ phase

Ưu điểm, nhược điểm

[Ưu, nhược mô hình lặp và tăng trưởng?]

Ưu điểm:

- Mỗi giai đoạn khách hàng có được sản phẩm thực hiện 1 phần công việc theo yêu cầu. Ngay từ phân phối ban đầu khách hàng đã có thể thấy được sự lợi ích của sản phẩm
  - Giảm thiểu được chi phí bảo trì
  - Các phần được thực hiện nhanh, tính bằng tuần lễ
  - Giảm shock tâm lý khi dùng sản phẩm mới hoàn toàn, Dần dần giúp khách hàng có thời gian thích nghi với sản phẩm
  - Không đòi hỏi khách hàng có kinh phí lớn, có thể dừng phát triển bất cứ lúc nào

Nhược điểm:

- Khó khi kết hợp build vào cấu trúc hiện thời là làm sao không phá hủy cấu trúc đã xây dựng. như vậy, yêu cầu cấu trúc phải mở
- Mặc dù uyển chuyển hơn các mô hình như thác nước và bản mẫu vì dễ thay đổi theo ý kiến khách hàng, nhưng có thể làm suy biến thành mô hình bản mẫu và khó điều phối chung.

[Ưu, nhược điểm mô hình xây và sửa? (code-and-fix life-cycle model)]

Ưu điểm:

- Có thể ngay lập tức bắt đầu phát triển, sửa chữa các vấn đề khi chúng xảy ra,thấy kết quả ngay lập tức, cho đến khi dự án hoàn thành.
- Phù hợp với những dự án nhỏ và ngắn.
- Tiết kiệm thời gian và là một mô hình thuận tiện cho các dự án ngân sách thấp

Nhược điểm:

- Nếu phát hiện ra các vấn đề lớn về kiến trúc trong quá trình này, thường phải viết lại các phần lớn của ứng dụng.
- Vì nó không được thiết kế cho các dự án phần mềm quy mô vừa và lớn, nó có thể được coi là phương pháp phát triển kém hiệu quả nhất trong nhiều trường hợp
- Rủi ro rất lớn đối với sự hài lòng của khách hàng và thậm chí là thất bại do không có sự hiểu biết cụ thể về các yêu cầu
- Tuy nhiên chi phí sửa chữa cao khi gặp phải lỗi.

[Ưu, nhược mô hình tháp nước? (waterfall life-cycle model)]

Ưu:

- Dễ pân công công việc, phân bố chi phí, giám sát công việc.
- Kiến trúc hệ thống hàng đợi ổn định.

Nhược:

- Mối quan hệ giữa các giai đoạn không được thể hiện
- Hệ thống phải được kết thúc ở từng giai đoạn do vậy rất khó thực hiện được đầy đủ những yêu cầu của khách hàng... vì trong mô hình này rất khó khăn trong việc thay đổi các pha đã được thực hiện. Giả sử, pha phân tích và xác định yêu cầu đã hoàn tất và chuyển sang pha kế tiếp, nhưng lúc này lại có sự thay đổi yêu cầu của người sử dụng; thì chỉ còn cách là phải thực hiện lại từ đầu.
- Chỉ tiếp xúc với khách hằng ở pha đầu tiên nên phần mềm ko đáp ứng được hết các yêu cầu của khách hang.
- Chi phí phát triển dự án tương đối lớn.
- Khả năng thất bại cao.

[Ưu, nhược mô hình bảo mẫu nhanh? (rapid-prototyping life-cycle model)]

Ưu:

- Giảm thời gian phát triển.
- Tăng khả năng tái sử dụng của các thành phần.
- Đưa ra đánh giá ban đầu nhanh chóng.
- Khuyến khích khách hàng đưa ra phản hồi.

Nhược:

- Thành viên trong nhóm phát triển sử dụng bản mẫu nhanh k tham khảo ý kiến khách hàng => có thể gây nên sai sót, không đúng yêu cầu khách hàng, gây tổn thất.
- nhóm thiết kế sử dụng đặc tả để thiết kế mà không cập nhật ý kiến khách hàng. => có thể gây nên sai sót, gây tổn thất.

[Ưu, nhược phương pháp họp đứng (stand-up meeting)?]

- Ưu điểm:
  + Cũng giống timeboxing, stand-up meeting giúp kết nối và đáp ứng yêu cầu của khách hàng nhanh nhất có thể
- Nhược điểm:
  + Giống timeboxing, chỉ phù hợp với các dự án nhỏ

[Ưu, nhược kỹ thuật timeboxing?]

Ưu:

+ Thúc đẩy năng suất và hiệu quả của quá trình phát triển phần mềm
+ Giảm thời gian phát triển phần mềm
+ Lập trình viên tập trung cao, ý thức về khoảng thời gian mình có
+ Giúp quản lí tốt rủi ro và độ phức tạp

Nhược:

+ Nếu không thể hoàn thành toàn bộ công việc trong khoảng thời gian được định sẵn thì công việc sẽ được cắt giảm bớt do timeboxing chỉ yêu cầu thời gian cố định chứ không có chỉ tiêu cố định nên chỉ thích hợp với các dự án nhỏ

[Ưu, nhược mô hình vòng đời xoắn ốc? (spiral life-cycle model)]

- Ý tưởng giảm thiểu rủi ro thông qua việc sử dụng các nguyên mẫu và các phương thức khác là ý tưởng nằm trong mô hình vòng đời xoắn ốc. Trước khi bắt đầu mỗi giai đoạn, một nỗ lực được thực hiện để giảm thiểu (kiểm soát) những rủi ro. Nếu không thể giảm thiểu tất cả các rủi ro đáng kể ở giai đoạn đó, thì dự án sẽ chấm dứt ngay lập tức

* Ưu điểm:
  - Phân tích rủi ro dự án được đầy lên làm một phần thiết yếu trong quy trình xoắn ốc để tăng độ tin cậy của dự án
  - Xây dựng dự án có sự kết hợp các mô hình khác vào phát triển (Thác nước, mô hình mẫu…)
  - Cho phép thay đổi tuỳ theo yêu cầu cho mỗi vòng xoắn ốc
  - Nó được xem như là một mô hình tổng hợp của các mô hình khác. Không chỉ áp dụng cho phần mềm mà còn phải cho cả phần cứng
  - Một rủi ro nào đó không được giải quyết thì chấm dứt dự án
  - Các vòng tròn được lặp để đáp ưng được những thay đổi của người dùng
  - Kiểm soát rủi ro ở từng giai đoạn phát triển.
  - Đánh giá tri phí chính xác hơn các phương pháp khác

* Nhược điểm:
  - phức tạp và không thích hợp với các dự án nhỏ và ít rủi ro.
  - Cần có kỹ năng tốt về phân tích rủi ro.
  - Yêu cầu thay đổi thường xuyên dẫn đến lặp vô hạn
  - Chưa được dùng rộng dãi như mô hình thác nước hay là mẫu.
  - Đòi hỏi năng lực quản lý

**[Ưu, nhược điểm mô hình vòng đời lặp đi lặp lại và gia tăng?
(iterative-and-incremental life-cycle model)]**

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