Đề tài: Tìm hiểu mô hình phát triển dự án Agile và phương pháp Scrum. Phát triển module quản lý lỗi cho dự án “Xây dựng ứng dụng phát triển dự án theo phương pháp Scrum”

MỤC LỤC

[**1.** **ĐẶT VẤN ĐỀ** 3](#_Toc311467810)

[**2.** **CƠ SỞ LÝ THUYẾT** 3](#_Toc311467811)

[**2.1.** **Giới thiệu về mô hình Agile** 3](#_Toc311467812)

[***2.1.1.*** ***Tổng quan*** 3](#_Toc311467813)

[***2.1.2.*** ***Đặc điểm của Agile*** 4](#_Toc311467814)

[**2.2.** **Phương pháp Scrum** 7](#_Toc311467815)

[***2.2.1.*** ***Tổng quan về Scrum*** 7](#_Toc311467816)

[***2.2.2.*** ***Đặc điểm của Scrum*** 9](#_Toc311467817)

[***2.2.3.*** ***Các bước phát triển dự án với phương pháp Scrum:*** 11](#_Toc311467818)

[***2.2.4.*** ***So sánh Scrum với các phương pháp truyền thống*** 12](#_Toc311467819)

[**3.** **KHẢO SÁT CÁC ỨNG DỤNG THỰC TẾ** 13](#_Toc311467820)

[**3.1.** **Những tính năng cơ bản** 14](#_Toc311467821)

[**3.2.** **So sánh những tính năng cơ bản** 15](#_Toc311467822)

[**4.** **XÂY DỰNG MODULE QUẢN LÝ LỖI** 16](#_Toc311467823)

[**4.1.** **Phân tích yêu cầu** 16](#_Toc311467824)

[***4.1.1.*** ***Phân tích các yêu cầu chức năng*** 16](#_Toc311467825)

[***4.1.2.*** ***Phân tích các yêu cầu phi chức năng*** 21](#_Toc311467826)

[**4.2.** **Thiết kế** 22](#_Toc311467827)

[***4.2.1.*** ***Thiết kế cơ sở dữ liệu*** 22](#_Toc311467828)

[***4.2.2.*** ***Thiết kế kiến trúc*** 22](#_Toc311467829)

[***4.2.3.*** ***Mô tả các thành phần ứng dụng*** 23](#_Toc311467830)

[***4.2.4.*** ***Công nghệ sử dụng*** 29](#_Toc311467831)

[**5.** **KẾT QUẢ** 29](#_Toc311467832)

1. **ĐẶT VẤN ĐỀ**

Ngày nay, khi ngành công nghệ thông tin đang tiếp tục phát triển với tốc độ rất nhanh đồng thời những thành tựu mà ngành mang lại cho đời sống xã hội cũng hết sức to lớn. Chúng ta có thể bắt gặp ứng dụng công nghệ hiện đại trong lĩnh vực thông tin ở bất cứ đâu, từ những trung tâm máy tính hiện đại, các nhà máy hay khu công nghiệp, những ngành kinh tế đang sử dụng những sản phẩm phần mềm hay thậm chí những thiết bị gia dụng được tích hợp công cụ xử lí thông minh… Tất cả đều tạo nên một nền tảng vững mạnh bao gồm cả phần cứng và phần mềm.

Lĩnh vực công nghệ phần mềm đang cùng với ngành công nghệ thông tin nói chung trở thành lĩnh vực khoa học trẻ tuổi song đầy sức mạnh. Hiện nay, nhiều ứng dụng phần mềm ra đời đã trở thành những công cụ, tiện ích không thể thiếu trong đời sống hàng ngày của chúng ta. Để đáp ứng được nhu cầu về sản phẩm với số lượng lớn và chất lượng cao, những nhà phát triển luôn định hướng việc xây dựng những phần mềm vừa đảm bảo về tốc độ, tính linh hoạt, kịp thời … cũng như chất lượng phục vụ người dùng cao nhất. Nhằm đảm bảo những tiêu chí trên, rất nhiều mô hình, phương pháp phát triển phần mềm đã ra đời nhằm giải quyết nhu cầu đó. Những phương pháp, quy trình tổ chức phát triển phần mềm trước đây như mô hình thác nước, xoắn ốc hay tiến hóa… Tuy nhiên, những mô hình này còn nhiều hạn chế trong việc tổ chức, quản lí nên những mô hình mới ra đời đã sớm thay đổi cách nhìn của các nhà phát triển về quy trình phát triển phần mềm. Điển hình trong số đó là mô hình Agile – một mô hình không chỉ đảm bảo tính khoa học trong các giai đoạn phát triển mà còn có những ưu điểm phù hợp với những giai đoạn bảo trì sau này. Từ khi Agile ra đời, những phương pháp mới cũng được hình thành và triển khai thực tế như Scrum, XP, AUP… Từ đó, nhóm sinh viên xây dựng đề tài tiểu luận *“Tìm hiểu mô hình Agile trong vấn đề quản trị dự án công nghệ thông tin”*. Đồ án tập trung vào tìm hiểu mô hình phát triển phần mềm nổi tiếng Agile, phương pháp triển khai Scrum, khảo sát các công cụ quản trị dự án theo mô hình Agile nổi tiếng, đồng thời phát triển module “Quản lý lỗi” cho dự án “Xây dựng ứng dụng phát triển dự án theo phương pháp Scrum”.

Trong suốt quá trình thực hiện đề tài, em luôn cố gắng tìm hiểu, cập nhật thông tin mới nhất về vấn đề quan tâm và những yếu tố liên quan. Tuy nhiên, báo cáo đồ án còn nhiều thiếu xót cần khắc phục, chúng em sẽ tiếp tục triển khai và hoàn thành trong thời gian tới. Em xin chân thành gửi lời cảm ơn tới TS.Phạm Huy Hoàng đã tận tình hướng dẫn, giúp đỡ em hoàn thành đồ án này.

1. **CƠ SỞ LÝ THUYẾT**
   1. **Giới thiệu về mô hình Agile**
      1. ***Tổng quan***

**Agile là gì?**

Trước hết, Agile là một triết lí (philosophy) cho việc phát triển phần mềm. Nói cách khác, đó là một cách “tư duy” về các dự án phần mềm. Các triết lí của Agile được cụ thể hóa bởi một số phương pháp phát triển phần mềm (method), chẳng hạn như Extreme Programming (XP) hay Scrum, gọi tắt là các phương pháp Agile.  
Mỗi phương pháp Agile bao gồm một tập hợp các quy tắc (pratice), chẳng hạn quy tắc về sử dụng công cụ quản lí mã nguồn, quy tắc về các chuẩn lập trình hay quy tắc trình diễn sản phẩm hàng tuần cho khách hàng.

Triết lí Agile được đưa ra trong một bản tuyên ngôn (manifesto) gồm 4 điểm và được làm rõ hơn bởi 12 quy tắc.

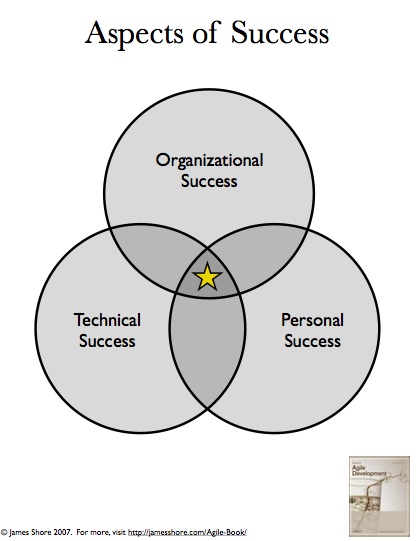
Dựa trên những triết lý đó, mô hình Agile được ra đời. Mô hình phát triển phần mềm Agile (Agile Software development) bao gồm một tập các cơ chế thực hiện các dự án công nghệ phần mềm dựa trên nguyên tắc ***“lặp và tăng trưởng”.*** Mục tiêu chính của mô hình nhằm giảm thiểu các rủi ro và thích nghi với các thay đổi liên tục bằng việc phát triển phần mềm trong các khung thời gian ngắn (***timeboxes)*** gọi là các bước tự lặp. Mỗi một bước lặp giống như một dự án phát triển phần mềm thu nhỏ, bao gồm đầy đủ các bước: lập kế hoạch (planning), phân tích yêu cầu (requirement analysis), thiết kế (design), lập trình (coding – programming), kiểm thử (testing), viết tài liệu (document), (triển khai (deployment)). Vào cuối mỗi bước lặp, bất kể kết quả như thế nào, nhóm phát triển đều phải ngồi lại và cùng đánh giá quá trình của bước lặp vừa rồi (retrospective).

**Vì sao chúng ta cần Agile?**

Theo quan niệm truyền thống, một dự án phần mềm được coi là thành công khi sản phẩm được giao đúng hạn, trong ngân sách cho phép và làm đúng yêu cầu của khách hàng. Trên thực tế, nhiều dự án thỏa mãn tất cả các tiêu chí này nhưng rút cuộc vẫn bị coi là thất bại bởi phần mềm làm ra không được người dùng ưa thích, hoặc không mang lại nhiều lợi ích cho các cá nhân, tổ chức sử dụng chúng.

Ngoài các yếu tố truyền thống nói trên, một dự án phần mềm chỉ được coi là thành công khi thỏa mãn ba tiêu chí: ***Thành công ở mức cá nhân, thành công về mặt kĩ thuật và thành công ở mức công ty***. Thành công ở mức cá nhân giúp kích thích các thành viên trong nhóm. Thành công về kĩ thuật đảm bảo khả năng bảo trì và tiến hóa của sản phẩm. Vị trí của nhóm sẽ được bảo đảm nhờ các thành công mà nhóm mang lại cho công ty. Các phương pháp Agile giúp cho dự án phần mềm đạt được ba thành công này.

* + 1. ***Đặc điểm của Agile***
       1. Những tiêu chí thành công

[](http://farm4.static.flickr.com/3566/3421845742_45cf0615b5_o.jpg)

*Ba tiêu chí thành công của một dự án phần mềm*

**Thành công ở mức công ty**

Agile nhắm đến việc tạo ra giá trị cho công ty trong khi làm giảm chi phí. Đồng thời, Agile giúp sớm xác định các kì vọng đối với sản phẩm đang được phát triển. Nhờ đó, những dự án không mang lại giá trị như mong đợi sẽ được phát hiện sớm, tiết kiệm chi phí cho công ty.

Theo phương pháp Agile, các chuyên gia về nghiệp vụ (business) sẽ làm việc trực tiếp cùng với đội dự án. Các chức năng quan trọng nhất của sản phẩm được tập trung phát triển trước và được đưa vào vận hành sớm nhất có thể. Các phiên bản mới với các tính năng mới sẽ lần lượt được đưa thêm vào.

Agile giúp tăng cường khả năng giao tiếp giữa các thành viên trong nhóm. Chất lượng mã nguồn được cải tiến liên tục. Tiến độ dự án cũng được xem xét và đánh giá một cách thường xuyên.

**Thành công về mặt kĩ thuật**

Trong Extreme Programming, một phương pháp tuân theo triết lí Agile, các lập trình viên làm việc cùng nhau. Nhờ vậy, các chi tiết quan trọng sẽ không bị bỏ sót, mỗi đoạn code sẽ được kiểm tra bởi ít nhất hai người. Các lập trình viên liên tục tích hợp những đoạn code vừa viết vào hệ thống, cho phép một phiên bản mới của phần mềm được “ra lò” bất cứ khi nào nó góp thêm một giá trị đáng kể. Hơn nữa, toàn bộ đội dự án tập trung hoàn thành một chức năng trước khi chuyển sang chức năng tiếp theo. Bởi vậy, tiến độ công việc được kiểm soát tốt hơn và dự án có thể dễ dàng “chuyển hướng” khi có những thay đổi từ phía khách hàng.

Ngoài ra, Extreme Programming cũng đề xuất những quy tắc giúp tạo ra các thiết kế và các đọan mã tốt. Chẳng hạn, quy tắc “phát triển dựa trên kiểm thử” (test-driven development) trợ giúp lập trình viên viết các chương trình thực hiện đúng chức năng mong muốn.

**Thành công về mặt cá nhân**

Mỗi thành viên trong dự án Agile, dù ở bất kì cương vị nào, cũng đều cảm nhận được một cách rõ ràng sự thành công của bản thân.  
Các lập trình viên nhận thấy trình độ kĩ thuật cũng như tầm ảnh hưởng của mình đối với dự án được nâng cao, chẳng hạn trong việc ước lượng và lập kế hoạch. Quyền tự chủ của đội dự án cũng được tăng cường.

Các tester nhận thấy họ có ảnh hưởng lớn đến chất lượng sản phẩm, đồng thời giảm được các công việc lặp lại một cách nhàm chán.

Nhà quản lí dự án hài lòng vì kiểm soát được tiến độ công việc, dự án thực hiện đúng các cam kết và làm thỏa mãn khách hàng.

Khách hàng, người sử dụng, các chuyên gia nghiệp vụ cảm thấy hài lòng vì điều kiển được hướng đi của dự án và các ý kiến được lắng nghe.

Các nhà lãnh đạo cao cấp sẽ cảm thấy hài lòng vì dự án mang lại lợi nhuận lớn cho công ty.

* + - 1. Các quy tắc của Agile

**Bản tuyên ngôn 4 điểm cùng 12 quy tắc của Agile:**

Tuyên ngôn 4 điểm của Agile là:

* Cá nhân và các tương tác quan trọng hơn quy trình và công cụ.
* Tập trung làm cho phần mềm chạy được thay vì viết tài liệu.
* Cộng tác trực tiếp với khách hàng thay vì dựa trên những bản hợp đồng.
* Phản ứng với các thay đổi thay vì tuân theo một kế hoạch định sẵn.

Bản tuyên ngôn được cụ thể hóa bằng 12 nguyên tắc sau:

* Ưu tiên cao nhất của dự án là thỏa mãn khách hàng bằng việc bàn giao sản phẩm sớm và liên tục.
* Hoan nghênh các thay đổi từ phía khách hàng, kể cả các thay đổi vào giai đoạn cuối.
* Bàn giao sản phẩm theo chu kì từ vài tuần đến vài tháng. Chu kì ngắn tốt hơn chu kì dài.
* Các nhân viên hiểu nghiệp vụ và các lập trình viên phải làm việc cùng nhau hàng ngày.
* Tổ chức dự án xoay quanh những cá nhân tích cực. Hỗ trợ và tin tưởng họ.
* Phương pháp giao tiếp tốt nhất trong đội dự án là gặp mặt trực tiếp.
* Các chức năng đã họat động là thước đo chính cho tiến độ dự án.
* Khuyến khích phát triển bền vững: Lập trình viên, người dùng, nhà quản lí…phải có khả năng tham gia dự án một cách liên tục.
* Liên tục cải tiến chất lượng thiết kế và mã nguồn.
* Tính đơn giản giữ vai trò cốt yếu. Làm càng ít càng tốt.
* Những yêu cầu và thiết kế tốt nhất được nảy nở từ những nhóm làm việc tự chủ.
* Sau những khoảng thời gian nhất định, đội dự án xem xét cách thức cải tiến hiệu quả công việc.
  + - 1. Phạm vi áp dụng mô hình Agile

Mặc dù Agile là một phương pháp tiên tiến nhưng không phải phải dự án nào cũng nên áp dụng Agile. Nếu mục tiêu của dự án là tăng năng suất lao động, làm cho các lập trình viên làm việc nhanh hơn thì Agile có thể không phù hợp bởi các quy tắc của nó không nhằm tăng năng suất của đội dự án.

Những dự án nên áp dụng Agile cần có những đặc điểm sau đây:

* + Mức độ rủi ro thấp.
  + Thành viên nhóm có kinh nghiệm.
  + Yêu cầu thay đổi thường xuyên.
  + Kích thước nhóm nhỏ. Các thành viên làm việc cùng một địa điểm.
  + Văn hóa công ty ưa thích sự “không trật tự” (thrive on chaos).

Trái lại, những điều kiện sau đây sẽ trở thành vật cản cho việc áp dụng Agile:

* + Kích thước nhóm lớn ( hơn 20 thành viên bao gồm lập trình viên, tester,…).
  + Các thành viên phân tán về mặt địa lí (ví dụ các dự án outsource).
  + Văn hóa làm việc theo mệnh lệnh.
  1. **Phương pháp Scrum**
     1. ***Tổng quan về Scrum***

Scrum là một quy trình phát triển phần mềm theo mô hình linh hoạt (agile). Agile cung cấp rất nhiều phương pháp luận, quy trình và các thực nghiệm để cho việc phát triển phần mềm trở nên nhanh chóng và dễ dàng. Hiện nay tại Việt Nam, quy trình này đang được thử nghiệm tại các đội phát triển phần mềm của một số công ty lớn. Scrum theo mô hình này.

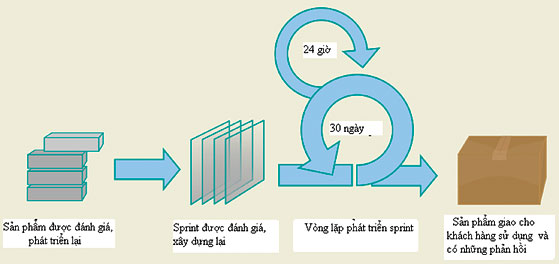
Scrum chia dự án thành các vòng lặp phát triển gọi là các ***sprint***. Mỗi sprint thường mất 2- 4 tuần (30 ngày) để hoàn thành. Nhằm đảm bảo cho những dự án có nhiều sự thay đổi và yêu cầu tốc độ cao.

Một sprint hoàn thành một số chức năng, mục đích nào đó trong toàn bộ hệ thống. Các tác vụ trong sprint được chia ra thành các danh mục, đội làm việc sẽ phát triển và đánh giá lại sao cho đạt được mục đích ban đầu trong khoảng thời gian đề ra.

Thành phần chính quan trọng của scrum là các role (vai trò) và các cuộc trao đổi đánh giá. Có các nhiệm vụ chính sau đây:

* ***Product Owner***: là người làm những công việc bắt đầu cho dự án, tạo ra các yêu cầu trong quá trình phát triển dự án. Phân tích mục tiêu, giải phóng các kế hoạch.
* ***Scrum Master***: họ phải đảm bảo các sprint được hoàn thành đúng mục đích, bảo vệ đội làm việc và loại bỏ các trở ngại.
* ***Đội làm việc ở scrum***: thường từ 5-9 người, tùy theo quy mô dự án nó có thể có rất nhiều đội, nhiều người tham gia. Sẽ không có những lập trình viên (programmer), người thiết kế (designer), kiểm thử viên (tester)… thường thấy ở các dự án phần mềm truyền thống. Các đội làm việc sẽ tiến hành cài đặt các chức năng được mô tả trong bản yêu cầu. Họ tự quản lý, tổ chức và điều chỉnh đội làm việc của mình sao cho hiệu quả lớn nhất. Tất cả các thành viên có ảnh hưởng như nhau đến sự thành công hoặc thất bại của toàn bộ hệ thống hoặc các hệ thống nhỏ hơn trong đó.

Có 2 pha là lập kế hoạch và kết thúc sẽ xác định các tiến trình cần thiết gồm các dữ liệu đầu vào đầu ra thật đầy đủ. Có một số vòng lặp phát triển trong pha kế hoạch. Kế hoạch lập ra ban đầu chỉ là tương đối và sẽ có sự điều chỉnh.



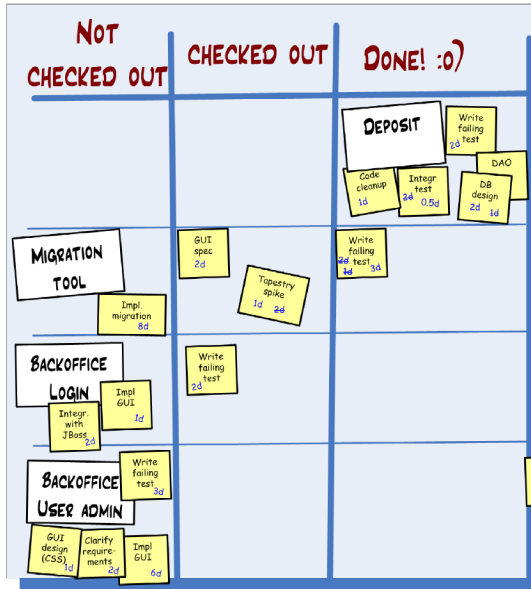
*Tổng quan về phương pháp Scrum*

|  |  |
| --- | --- |
| |  | | --- | |  | |

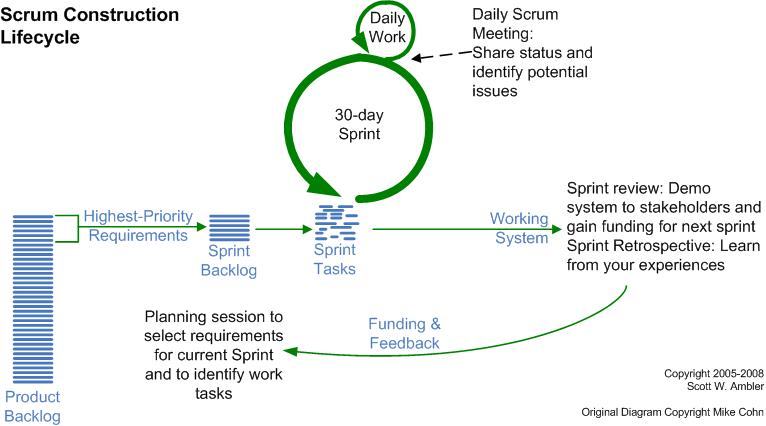
* + 1. ***Đặc điểm của Scrum***

**Các khái niệm của Scrum**

* ***Product backlogs***: đây có thể coi là trái tim của phương pháp Scrum. Mọi thứ đều bắt đầu từ product backlogs, đây có thể là tập hợp các yêu cầu của khách hàng, các đặc điểm của phần mềm …Nói cách khác đây là những gì mà khách hàng muốn và thường được mô tả bởi tài liệu (hay đôi khi là các thuật ngữ) của khách hàng.
* ***Sprint***: một bước lặp trong dự án.
* ***Scrum master***: người đảm nhận nhiệm vụ hoàn thành các sprint theo đúng các mục đích (sprint goals). Đồng thời, giải quyết các vấn đề mà đội phát triển gặp phải.
* ***Sprint planning***: đây là một buổi họp với mục tiêu cung cấp đầy đủ thông tin cho nhóm phát triển có thể làm công việc một cách trơn tru. Những output cuối buổi họp cần có:
  + Mục tiêu cuối cùng của bước lặp (sprint goal).
  + Danh sách các thành viên tham gia trong bước lặp.
  + Sprint backlog.
  + Ngày trình bày bản phiên bản thử nghiệm (sprint demo date).
  + Lên kế hoạch các buổi họp tiến độ - Daily scrum.
* ***Sprint backlogs***: danh sách các công việc, các yêu cầu cần phải hoàn thành trong bước lặp này.
* ***Daily scrums***:các buổi họp trong mỗi bước lặp với mục đích báo cáo tiến độ.
* ***Sprint demos:*** buổi trình bày những vấn đề đã làm được của bước lặp.
* ***Sprint retrospectives***: sau mỗi một bước lặp (sprint), cả nhóm sẽ ngồi lại cùng nhau và tổng kết những vấn đề còn thiếu sót, tồn đọng trong bước lặp vừa rồi để đưa ra phương án cải thiện cho sprint sắp tới.
* ***Kanban:*** đây là một phương pháp giúp quản lí các công việc của cá nhân và toàn đội thông qua một chiếc bảng (billboard). Mỗi công việc được ghi lên tấm sticky note hoặc card và dán lên bảng. Bảng thường có 3 cột cơ bản là: “Not checkout”, “Checkout – Inproccess”, “Done” thể hiện cho 3 quá trình cơ bản của một công việc (một số bảng có thêm cột “Tested”).



**Quy trình phát triển một dự án phần mềm theo phương pháp Scrum:**



*Vòng đời phát triển phần mềm theo mô hình Scrum*

Ở trên là ví dụ về một vòng đời phát triển phần mềm theo phương pháp Scrum với độ dài một sprint là 30 ngày – 4 tuần. Ta có thể thấy các bước của một sprint bao gồm:

* Chọn các Sprint Backlogs và Planning
* Phân chia các sprint backlogs thành các công việc cụ thể (tasks)
* Phân tích, thiết kế
* Lập trình
* Kiểm thử
* Triển khai sản phẩm
* Rút kinh nghiệm qua sprint vừa rồi.
  + 1. ***Các bước phát triển dự án với phương pháp Scrum:***

Có nhiều cách để triển khai việc phát triển phần mềm theo phương pháp Scrum, tuy nhiên có 10 bước cơ bản sau đây:

* Bước 1: Thu nhập các đặc điểm của sản phẩm (backlog) trong đơn đặt hàng. Đây là bước quan trọng nhất. Lập nên các đội làm việc, có thể tách thành các đội nếu cần thiết và thảo luận với nhau về nghiệp vụ cần làm. Sau đó bổ nhiệm một người vào vị trí Product owner, người này có khả năng trao đổi, bao quát công việc tốt, biết sắp xếp ưu tiên đúng thứ tự các nhiệm vụ. Sau đó tự tổ chức lại đội làm việc, đề xuất ra vị trí Scrum master và thảo luận chi tiết các yêu cầu, sắp xếp chúng theo thứ tự ưu tiên.
* Bước 2: Ước lượng đầy các yêu cầu về sản phẩm đầu ra. Có ước lượng ở mức độ cao, chia sản phẩm thành số lượng các danh mục backlog. Tuy nhiên số lượng sẽ không chính xác được, về sau chúng sẽ được bổ sung. Tiếp đến là ước lượng chi tiết từng backlog, ước lượng số lượng các đội làm việc.
* Bước 3: Lên kế hoạch phát triển các vòng lặp sprint. Sử dụng các cuộc trao đổi kế hoạch phát triển sprint với tất cả các thành viên. Xác định khoảng thời gian sẽ phát triển một sprint (thường là 30 ngày), mục tiêu của sprint là gì, sẽ đạt được gì, phân tích các yêu cầu của sprint một cách rõ ràng.
* Bước 4: Lên kế hoạch phát triển các nhiệm vụ của sprint. Tất cả mọi người sẽ xác định ngân sách của sprint đó, chia các đặc điểm thành các tác vụ nhỏ hơn, ước lượng số thời gian sẽ làm từng task (giờ), hoàn tất các yêu cầu và nhận dạng task quan trọng.
* Bước 5: Tạo ra không gian làm việc cộng tác cho tất cả mọi người. Thường sử dụng bảng trắng để vẽ nên những vấn đề cần thiết cho tất cả mọi người cùng đánh giá.
* Bước 6: Các thành viên bắt tay xây dựng từng sprint. Lập trình, kiểm thử và điều chỉnh thời gian để có hiệu quả tốt nhất. Đôi khi có thể hủy bỏ một sprint và quay lại với việc lập kế hoạch khác.
* Bước 7: Mọi người báo cáo kết quả để tiếp tục làm việc. Các báo cáo tập trung vào các vấn đề: đạt được những gì so với lần trao đổi trước; sẽ hoàn thành những gì trong lần trao đổi tiếp theo; có những trở ngại gì trong quá trình làm việc v.v.
* Bước 8: Tổng hợp kết quả trên biểu đồ. Đây là bức tranh tổng quát về những việc đã làm được, những việc chưa làm được, thời gian ước lượng còn lại và có thể điều chỉnh lại.
* Bước 9: Xem xét để hoàn tất. Khi các thành viên nói công việc đã hoàn thành có nghĩa là mọi thay đổi sẽ bị từ chối, đẩy lại cho vòng lặp sau.
* Bước 10: Đánh giá, phản ánh và lặp lại. Có các cuộc họp đánh giá lại sprint của các thành viên. Sẽ trình bày những gì đạt được, phản hồi của khách hàng, xét thời hạn của sprint. Nhìn lại biểu đồ ở bước 8 để xác định lại toàn bộ hệ thống và tiếp nhận những đóng góp, bổ sung để đưa tiếp vào các vòng lặp sprint tiếp theo
  + 1. ***So sánh Scrum với các phương pháp truyền thống***

Với các phương pháp truyền thống, việc lập kế hoạch dự án (xác định những việc cần làm và thời gian kế thúc) dựa trên kinh nghiệm chứ không phải là môi trường làm việc trực tiếp. Và so với kế hoạch đó khi bắt tay vào xây dựng thì thường có độ trễ nhất định.

Như trong mô hình thác nước chia dự án phần mềm gồm các giai đoạn: đặc tả yêu cầu, thiết kế hệ thống, cài đặt (lập trình), kiểm thử và bảo trì. Quy trình này dễ quản lý nhưng lại kém linh hoạt và không hiệu quả bởi nếu có sự thay đổi ở các giai đoạn sau sẽ ảnh hưởng rất lớn đến các giai đoạn trước.

Quy trình xoắn ốc chia dự án thành các giai đoạn: lập kế hoạch, phân tích rủi ro, giao tiếp khách hàng, đánh giá lại, sản xuất và phân phối. Tuy nhiên, quy trình này vẫn chưa được sử dụng rộng rãi.

Các quy trình phần mềm truyền thống thường có quá khá nhiều giai đoạn, nhiều thành phần, yếu tố trong suốt thời gian phát triển sản phẩm. Phương pháp scrum giúp cho dự án phần mềm tránh khỏi điều này.

Điểm mạnh nhất của phương pháp Scrum nói riêng cũng nhưng các phương pháp dựa trên mô hình Agile nói chung là sự linh hoạt, dự án không được cố định từ đầu về thời gian hoàn thành hay những yêu cầu mà nó sẽ được xác định khi phát triển thực tế.

Phương pháp Scrum phân phối sản phẩm mềm dẻo: nội dung sản phẩm chuyển giao được xác định linh hoạt theo môi trường sử dụng thực tế.

Thời gian biểu của Scrum linh hoạt: có thể muộn hoặc sớm hơn so với kế hoạch ban đầu.

Dự án phát triển theo phương pháp Scrum có chất lượng sản phẩm tốt và giảm rủi ro sản xuất, chi phí thấp. Khả năng trao đổi giữa khách hàng và nhà phát triển, giữa những thành viên trong đội được đặt lên mức cao.

Tốc độ phát triển nhanh, tiết kiệm thời gian. Việc chuẩn bị hành động cho những thay đổi trong quá trình phát triển tốt hơn vì hầu như hàng ngày luôn có những buổi họp đánh giá lại ở những vòng lặp phát triển.

Các bugs (lỗi) và các vấn đề được phát hiện sớm hơn rất nhiều so với các phương pháp truyền thống bởi vì khách hàng được tham gia đánh giá rất nhiều và đầu ra của sản phẩm rất nhanh. Và khi đi sai hướng, có thể hủy ngay sprint đó để quay lại với bản kế hoạch.

|  |  |  |  |
| --- | --- | --- | --- |
| **Đặc điểm** | **Waterfall** | **Spiral** | **Scrum** |
| Xác định các giai đoạn phát triển | Bắt buộc | Bắt buộc | Chỉ có giai đoạn lập kế hoạch và kết thúc |
| Sản phẩm cuối cùng | Được xác định trong quá trình lập kế hoạch | Được xác định trong quá trình lập kế hoạch | Xác định trong quá trình xây dựng dự án |
| Chi phí sản phẩm | Được xác định trong quá trình lập kế hoạch | Thay đổi cục bộ | Xác định trong quá trình xây dựng dự án |
| Ngày hoàn thành sản phẩm | Được xác định trong quá trình lập kế hoạch | Thay đổi cục bộ | Xác định trong quá trình xây dựng dự án |
| Đáp ứng với môi trường sử dụng | Trong kế hoạch ban đầu | Trong kế hoạch ban đầu | Xuyên suốt từ kế hoạch đến xây dựng và kết thúc |
| Kinh nghiệm trao đổi | Đào tạo trước cho đến khi bắt tay làm dự án | Đào tạo trước cho đến khi bắt tay làm dự án | Thực hiện trong quá trình làm dự án |
| Khả năng thành công | Thấp | Trung bình thấp | Cao |

*Bảng so sánh các đặc điểm của Scrum với nhưng phương pháp khác*

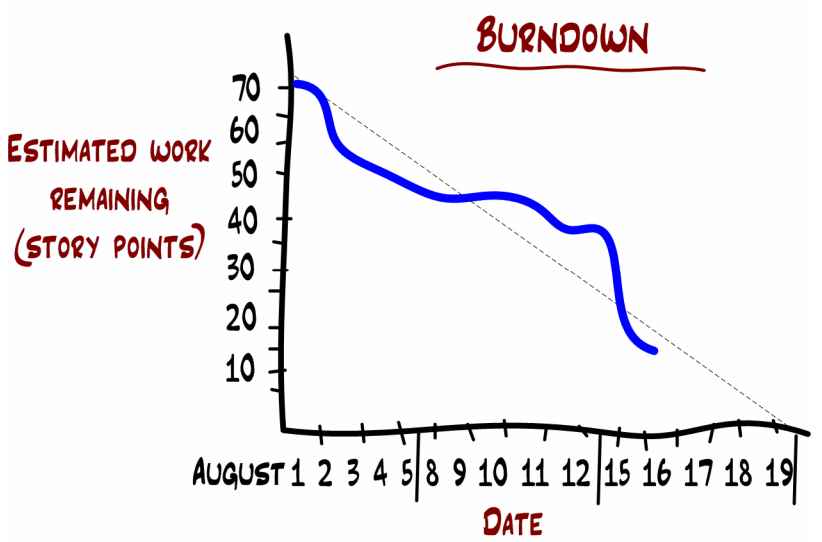
1. **KHẢO SÁT CÁC ỨNG DỤNG THỰC TẾ**

Nhờ các điểm tiên tiến của mô hình Agile mà hiện nay những phương pháp phát triển phần mềm dựa trên mô hình này đã và đang được áp dụng rộng rãi. Để đảm bảo cho việc phát triển phát triển dự án theo mô hình Agile được nhanh chóng và đơn giản, rất nhiều các phần mềm hỗ trợ đã ra đời. Trong đồ án này, em tập trung vào việc khảo sát các phần mềm hỗ trợ phương pháp Scrum. Các phần mềm khảo sát bao gồm: ***“Agilo”*** và ***“Plugin GreenHooper cho Jira”.***

* 1. **Những tính năng cơ bản**

Dựa trên những đặc điểm chung của mô hình Agile và các đặc điểm riêng của phương pháp Scrum. Một phần mềm hỗ trợ quản trị dự án phát triển theo phương pháp Scrum cần có những tính năng cơ bản sau đây.

* **Quản lí thành viên:** Bao gồm các chức năng thêm mới, chỉnh sửa thông tin của các thành viên trong nhóm phát triển, thường thì mỗi thành viên sẽ được cung cấp một username và password để truy cập. Những tính năng cơ bản gồm:
  + - * Tạo mới và chỉnh sửa thông tin thành viên.
      * Phân quyền cho các thành viên.
      * Gửi mail từ hệ thống tới các thành viên.
* **Quản lí các sản phẩm (product)**: Bao gồm các chức năng thêm mới, chỉnh sửa thông tin của các dự án, một số phần mềm còn có tên gọi khác là project. Hoặc có cả hai project, product. Với những phần mềm như vậy, product có thể coi là các version của một dự án, tuy nhiên rất ít những phần mềm theo hướng làm này.
* **Quản lí các yêu cầu (backlogs – user stories):** Khi đã tồn tại một product, tính năng này giúp người quản trị hay những thành viên có quyền thêm mới và chỉnh sửa các backlogs. Những tính năng cơ bản gồm:
  + - * Tạo mới một backlogs.
      * Chỉnh sửa các thông tin của một backlogs (các thông tin quan trọng như: thời gian cần thiết để hoàn thành, thời gian bắt đầu, người thực hiện).
      * Phân loại các backlog: một số phần mềm hỗ trợ phân loại các backlog từ thấp tới cao giúp người quản trị dễ dàng tạo các phân nhóm cho backlog hay chia nhỏ các backlog.
      * Quản lí tiến độ của một backlog (đã hoàn thành những việc gì, đến đâu…)
* **Quản lí các bước lặp (sprint):**
  + - * Tạo sprint trong các product.
      * Chỉnh sửa các thông tin của sprint.
      * Đưa các product backlog thành các sprint backlog.
* **Quản lí quy trình của một sprint:**
  + - * Sprint planning
      * Daily scrum
      * Sprint demo
      * Sprint deploy
      * Sprint retrospective
* **Hỗ trợ kanban**: bảng các công việc.
* **Báo cáo tiến độ:** Trong quản trị dự án, việc cập nhật tiến độ là một yếu tố rất quan trọng. Những tính năng cơ bản bao gồm:
  + - * Burndown chart: đây là bảng báo cáo (hoặc đồ thị) gồm các thông tin trong một sprint:
        + Số lượng công việc cần phải hoàn thành.
        + Số lượng công việc cần hoàn thành trong một ngày.
        + Số lượng công việc mà cả nhóm đã làm.



Burndown chart

* + - * + Workload chart: đồ thị này báo cáo tỉ lệ công việc (tính theo thời gian, hoặc số lượng) của các thành viên nhằm giúp người quản trị phân chia công việc chính xác và cân bằng hơn.
        + Working chart: đây là đồ thị thông báo số lượng công việc mà các thành viên đã hoàn thành trong sprint hiện tại nhằm đưa ra một cái nhìn tổng quan cho người quản trị.
  1. **So sánh những tính năng cơ bản**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tính năng** | | **Agilo** | **Jira** | **Ghi chú** |
| ***Quản lí thành viên*** | Thêm mới và chỉnh sửa thành viên | **x** | **x** |  |
| Hỗ trợ gửi email | **x** | **x** |  |
| Phân quyền cho các thành viên | **x** | **x** |  |
| Thêm mới và chỉnh sửa thành viên | **x** | **x** |  |
| ***Quản lí sản phẩm - dự án*** |  | **x** | **x** |  |
| ***Quản lí yêu cầu*** | Tạo mới và chỉnh sửa các backlog | **x** | **x** |  |
| Phân loại backlog | **x** | **x** | \* Agilo phân các backlog ra làm user story,  technical story và impediment.  \* Jira linh hoạt hơn, cho phép người dùng  tạo các phân loại và đánh nhãn cho các backlog (bug, enhancement ...). |
| Phân nhóm backlog | **x** | **x** | \* Jira cho phép người dùng đặt các backlog vào các nhóm theo các phân cấp như chương epic, hồi, giúp quản lí các backlog dễ dàng hơn. |
| Tạo thêm các task, subtask trong mỗi backlog |  | **x** |  |
| Quản lí tiến độ của một backlog, task, subtask | **x** | **x** |  |
| ***Quản lí bước lặp*** | Tạo và chỉnh sửa các sprint | **x** | **x** |  |
| Quản lí qui trình của sprint | **x** | **X** | \* Agilo và Jira đều giao diện thân thiện, hỗ trợ kéo thả các product backlog vào sprint. Tuy nhiên jira giúp người dùng tính toáncác thông số như team velocity, số lượng backlog có thể hoàn thành trong sprint (dựa vàothời lượng của sprint, thời gian đánh giá trướccho mỗi backlog, độ ưu tiên cho backlog. |
| Hỗ trợ kanban | **x** | **x** |  |
| ***Báo cáo*** | Burndown chart | **x** | **x** |  |
|  | Workload chart |  | **x** |  |
|  | Working chart |  | **x** |  |

* Nhận xét:

Cả Agilo và Jira đều đảm bảo gần như đầy đủ các yêu cầu cơ bản của một phần mềm hỗ trợ phát triển dự án theo phương pháp Scrum. Tuy nhiên, về tính tiện dụng, giao diện phần mềm cũng như các tính năng khác hỗ trợ người dùng thì Jira hơn hẳn Agilo.

1. **XÂY DỰNG MODULE QUẢN LÝ LỖI**
   1. **Phân tích yêu cầu**
      1. ***Phân tích các yêu cầu chức năng***

Tương nhự như các yều cầu chung của phần mềm hỗ trợ phát triển dự án Scrum. Module quản lý lỗi có các yêu cầu nghiệp vụ sau:

* Cho phép tạo mới/chỉnh sửa/xóa các thông tin của lỗi (issue).
* Người dùng có thể phân loại các issue nhằm tiện lợi hơn cho việc quản trị dự án cũng như tìm kiếm.
* Gán (assign) lỗi cho các thành viên trong dự án.
* Tìm kiếm các issue theo dự án (project/product), theo các danh mục mà người dùng có thể định nghĩa trước hoặc theo các sprint (milestone) của dự án.

Dưới đây là các biểu đồ ca sử dụng (use-case) và biểu đồ tương tác giữa các thành phần trong ứng dụng:

1. Các đối tượng (actor) của ứng dụng bao gồm ba đối tượng chính: Product owner, Scrum master và team member. Các đối tượng này đều kế thừa từ đối tượng tổng quát User.



*Các đối tượng chính trong ứng dụng*

Đối tượng User cần có các thông tin cơ bản như: tên truy cập, mật khẩu, email, tên.

1. Các use-case cơ bản

* Đăng ký/ Đăng nhập
* Quản lý lỗi (issue)
* Quản lý người dùng



*Các Use-case cơ bản*

**Quản lý lỗi:**



*Các tính năng của module quản lý lỗi*

* Thêm mới/chỉnh sửa/xóa issue



*Biểu đồ tương tác tính năng thêm mới issue*



*Biểu đồ tương tác tính năng chỉnh sửa issue*

* Phân loại issue
* Assign issue cho team member
* Thêm các task cho issue
  + 1. ***Phân tích các yêu cầu phi chức năng***

Bên cạnh những yêu cầu chức năng đã nêu trên, ứng dụng cần có các yêu cầu sau đây:

* Tái sử dụng mã nguồn: trong việc xây dựng phần mềm, tái sử dụng mã nguồn là một yêu cầu hết sức quan trọng. Việc thiết kế ứng dụng có khả năng tái sử dụng cao giúp cho việc phát triển các tính năng mới được đơn giản hóa, giảm thiểu thời gian cũng như chí.
* Dễ mở rộng: Ứng dụng phải được thiết kế sao cho việc thêm mới các module cũng như các tính năng có thể thực hiện hoàn toàn đơn giản.
* Tối ưu hiệu năng: bên cạnh việc tái sử dụng mã nguồn, khả năng mở rộng cao, ứng dụng còn phải có hiệu năng tốt. Việc này giúp giảm sự lãng phí về tài nguyên hệ thống cũng như tăng độ tin cậy của người dùng đối với ứng dụng.
* Tính mềm dẻo cao: Ứng dụng có thể tích hợp với nhiều nguồn dữ liệu từ các cơ sở dữ liệu khác nhau (MS SQL, Oracle, MySQL…) đồng thời ứng dụng có thể sử dụng các ngôn ngữ khác nhau để xây dựng tầng hiển thị. Tuy nhiên, trong đồ án này, em sẽ chỉ sử dụng ngôn ngữ C# và cơ sở dữ liệu Microsoft SQL.
  1. **Thiết kế**
     1. ***Thiết kế cơ sở dữ liệu***
     2. ***Thiết kế kiến trúc***

Ứng dụng được phân tầng theo 3 tầng chính, mỗi tầng đảm nhận một tác vụ riêng. Việc phân chia này giúp ứng dụng có thể triển khai trên các mô hình phân tán khi cần mở rộng hoặc tăng hiệu năng.



*Tổng quan kiến trúc ứng dụng*

**Tầng trình diễn (Presentation Tier)**

Giao diện tương tác với người dùng cuối qua ứng dụng web (web application) hoặc các ứng dụng trên nền tảng khác.

Tầng trình diễn của ứng dụng giao tiếp với tầng nghiệp vụ trung gian thông qua các dịch vụ mà tầng nghiệp vụ trung gian cung cấp.

**Tầng dịch vụ (Service Tier)**

Cung cấp các dịch vụ của hệ thống tới tầng trình diễn và môi trường bên ngoài (các nhà phát triển ứng dụng, người sử dụng…).

Nhận và xử lý các yêu cầu từ tầng trình diễn. Cung cấp các dịch vụ cho các tầng phía trên sử dụng. Ngoài ra tầng nghiệp vụ trung gian đảm nhận vai trò là cầu nối giữa việc truy xuất dữ liệu từ các tầng phía trên với tầng tài nguyên.

Tầng này đảm nhận các chức năng xử lý nghiệp vụ của hệ thống.

**Tầng tài nguyên (Data Access tier)**

Đây là tầng sẽ thao tác trực tiếp với cơ sở dữ liệu. Bao gồm các nguồn cung ứng dữ liệu, thông tin, tri thức của hệ thống cho tầng nghiệp vụ trung gian.

* + 1. ***Mô tả các thành phần ứng dụng***
       1. Tầng trình diễn (Presentation Tier)
       2. Tầng dịch vụ (Service Tier)

Tầng này đảm nhận nhiệm vụ xử lý các nghiệp vụ chức năng của hệ thống. Bên cạnh đó cung cấp các dịch vụ cho tầng trình diễn và các ứng dụng khác sử dụng (**Web service**)thuộc tầng môi giới dịch vụ.

1. **Business Entities**

Ánh xạ cơ sở dữ liệu thành các các đối tượng, thực thể của thế giới thực theo nguyên lý của lập trình hướng đối tượng (object-oriented). Business Entities sẽ đóng gói và cung cấp dữ liệu dưới dạng các thực thể (objects) cho tất cả các tầng trong toàn bộ hệ thống thông qua các thuộc tính (properties). Business entities không truy cập vào dữ liệu một cách trực tiếp mà thông qua tầng tài nguyên để mô phỏng dữ liệu.

**Business Entities** còn làm nhiệm vụ chuẩn hóa(validate) dữ liệu.

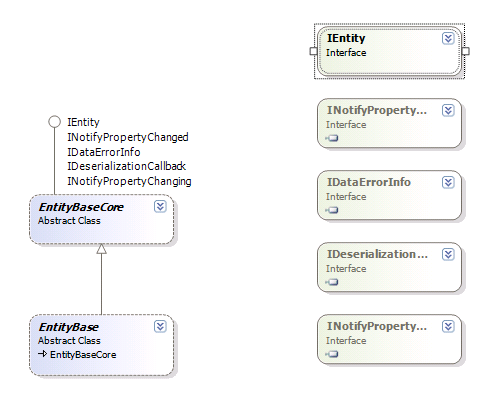
Các mẫu thiết kế áp dụng

* + - Abstract Factory Pattern
    - Memento Pattern
    - State Pattern
    - Table Module

Thiết kế mà hệ thống sử dụng để xây dựng **business entities** là **Table Module**. Cụ thể hơn, các bảng ở trong cơ sở dữ liệu sẽ được mô phỏng thành các lớp, các cột trong cơ sở dữ liệu được mô phỏng thành các **attribute, property**. Bên cạnh đó, để tiện lợi hơn trong việc truy xuất dữ liệu các quan hệ giữa các bảng (**relationship)** cũng được mô phỏng lại thành các **property (**quan hệ 1-n) hoặc **collection** (quan hệ m-n).

Các lớp **Entity** đều được kế thừa từ **EntityBase**. **EntityBase** kế thừa các giao diện:

* + - **IEntity:** những thuộc tính, phương thức chung cho các base. (Các lớp **base** khác cũng sẽ phải kế thừa **IEntity** này).
    - **INotifyPropertyChanged**: thông báo cho client rằng dữ liệu đã thay đổi.
    - **InotifyPropertyChanging:**thông báo cho client rằng dữ liệu đang bị thay đổi.
    - **IDataErrorInfo**: cung cấp các thông tin về thông báo lỗi.
    - **IDeserializationCallback:** thông báo khi dữ liệu đã hoàn thành việc sắp xếp.



*Biểu đồ lớp của* ***EntityBase***

Mỗi một thực thể (entity) đều có một vòng đời và các trạng thái nhất định. Lớp **EntityBase** bao gồm những hành vi đã được đóng gói để quản lý các trạng thái này. Dựa vào các trạng thái của các thực thể, hệ thống dễ dàng quản lý các giao dịch (**transaction)**, giải quyết các xung đột, tương tranh khi có các yêu cầu đọc, sửa đổi, hay xóa dữ liệu. Các trạng thái gồm có:

* + - **Unchanged:** Khi dữ liệu được đọc ra từ cơ sở dữ liệu và chưa có thay đổi gì.
    - **Added:** Khi có yêu cầu thêm dữ liệu vào cơ sỡ dữ liệu.
    - **Changed:** Khi có yêu cầu thay đổi dữ liệu trong cơ sở dữ liệu.
    - **Deleted:** Khi có yêu cầu xóa dữ liệu trong cơ sở dữ liệu

Khi việc thao tác với dữ liệu không thành công, thực thể (**Entity)** sẽ quay về trạng thái ngay trước đó. Để làm việc này, hệ thống sử dụng mẫu thiết kế **Memento Pattern.** Các thực thể đóng vai trò quản lý các trạng thái thay vì sử dụng 1 lớp khác như trong mẫu thiết kế chuẩn.

Để dễ dàng mở rộng hơn, hệ thống còn có một lớp ảo **EntityFactory** cung cấp các phương thức tiện ích để khởi tạo các đối tượng (tạo mới-**Create<T>(), CoreCreate(Type type)**, các tiện ích để định vị kiểu thực thể **EntityLocator …)**. Nhà phát triển có thể viết các **Factory** cụ thể của từng đối tượng để khởi tạo đối tượng theo các yêu cầu cụ thể. **EntityFactory** được thiết kế dựa chủ yếu vào mẫu **Abstract Factory Pattern** và **Dependency Injection.**

Trong hệ thống **Business Entities** có tên là **Agile.Entities.**

1. **Business Services**

Thành phần này sẽ sử dụng các đối tượng mà Business Entities đã mô tả để thực hiện các nghiệp vụ logic trên các đối tượng đó. Các nghiệp vụ logic được thực hiện như các dịch vụ, cung cấp cho tầng phía trên.

Mẫu thiết kế áp dụng

* + - Remote Façade

Thay vì việc đóng gói các phương thức cơ bản của các đối tượng nghiệp vụ vào Business Entities. Thiết kế sẽ tách riêng phần hành vi thành Business Services bao gồm các lớp logic nghiệp vụ (Business Logic Object – Domain Model), các lớp Service sẽ sử dụng các logic nghiệp vụ và đóng gói thành các dịch vụ cụ thể (Remote Facade) cung cấp cho các tầng cao hơn sử dụng. Với việc hướng dịch vụ(Service-oriented) như vậy, ta có thể thể dễ dàng hơn trong việc cung cấp các dịch vụ cho các lớp khác cũng như tái sử dụng lại các dịch vụ mà không cần quan tâm đến dữ liệu có những gì. Việc tái sử dụng mã nguồn cũng sẽ đơn giản hơn rất nhiều.

Các dịch vụ đều được kế thừa từ lớp ảo **ServiceBase.** Lớp này bao gồm các thành phần:

* + - Các phương thứcảo **Get,** **Insert, Update, Delete:** làm các nhiệm vụ đọc, thêm mới, sửa đổi, xóa các bản ghi trong cơ sở dữ liệu (sử dụng các phương thức là **Data Layer** cung cấp).
    - Các phương thức **DeepLoad, DeepSave:** nạp dữ liệu từ các bảng liên quan (quan hệ 1-1, 1-n) vào trong đối tượng hiện tại (sử dụng các phương thức mà **Data Layer** cung cấp)
    - **ProcessorList**: tập hợp các bộ xử lý nghiệp vụ (sẽ trình bày ở Business process).
    - **Execute**:yêu cầu các bộ xử lý trong **ProcessorList** làm nhiệm vụ của nó.

Các phương thức hiện nay của hệ thống giao tiếp **với Data Access Logic** đều được sử dụng **TransactionManager** để quản trị các giao dịch (chi tiết ở **Data Access Logic**).

* + - 1. Tầng truy cập dữ liệu (Data Access Tier)

Tầng này bao gồm các nguồn cung ứng dữ liệu, thông tin, tri thức của hệ thống cho tầng nghiệp vụ trung gian. **Data Access Logic Layer** sẽ tương tác trực tiếp với cơ sở dữ liệu (ở hệ thống là **OTS.Data.SqlClient** - do hệ thống sử dụng cơ sở dữ liệu SQL Server). **Data Provider Layer** sẽ sử dụng những đối tượng mà **Data Access Logic Layer** cung cấp, cùng với các đóng gói chúng thành các phương thức, dịch vụ và cung cấp cho các tầng cao hơn. Như vậy, khi ta thay đổi cơ sở dữ liệu (từ SQL Server sang MySQL, Oracle …) ta chỉ cần thay đổi lớp (layer) này.

1. **Data Provider Layer**

**Data Provider Layer** làm nhiệm vụ cung ứng các giao diện lập trình (API) cho các tầng cao hơn (cụ thể là tầng trung gian) sử dụng. Nói cách khác, đây chính là cầu nối giữa tầng trung gian và các lớp truy cập cơ sở dữ liệu.

Cơ sở dữ liệu có thể dùng ở nhiều loại khác nhau (SQL server, MySQL, Oracle…) vì vậy, nếu không có **Data Provider Layer** làm cầu nối, mỗi khi thay đổi loại cơ sở dữ liệu ta sẽ gần như phải viết mới toàn bộ các thành phần tương tác trực tiếp với cơ sở dữ liệu (**Data Access Components)**. Việc này rất tốn kém thời gian. Thay vì đó, ta định nghĩa các nguồn cung ứng dữ liệu là các lớp ảo (abstract) và yêu cầu các thành phần tương tác trực tiếp với cơ sở dữ liệu hiện thực hóa chúng. Các tầng cao hơn cũng không cần quan tâm đến cơ sở dữ liệu là loại gì và cũng không cần quan tâm đến các thành phần tương tác trực tiếp với cơ sở dữ liệu làm những gì mà chỉ cẩn sử dụng các giao diện lập trình (interface), hoặc các thành phần ảo mà **Data Provider Layer** cung cấp.

Các mẫu thiết kế áp dụng:

* + - Repository Provider Pattern
    - Singleton Pattern
    - Data Transfer Object

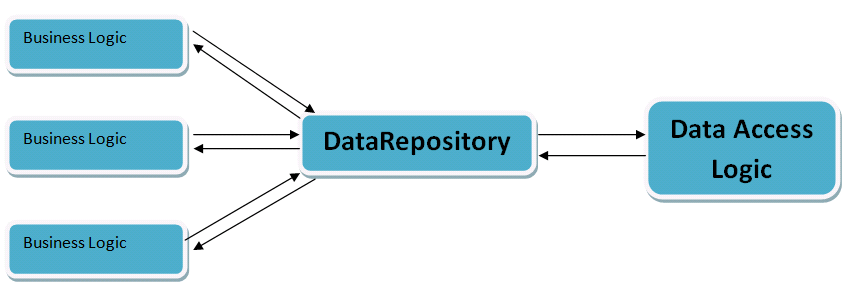
Các thành phần khác khi sử dụng **Data Provider Layer** có thể tạo một tham chiếu đến các lớp ảo của **Data Provider Layer** và sử dụng các phương thức ảo của chúng. Nhờ tính đa hình của lập trình hướng đối tượng, người phát triển có thể dễ dàng chuyển đổi cơ sở dữ liệu cũng như loại cơ sở dữ liệu khác. Tuy nhiên cách làm trên cũng có một số điểm yếu như sau:

* + - Rất dễ bị trùng lặp mã nguồn, tính tái sử dụng thấp.
    - Khó khăn khi muốn tập trung dữ liệu để thực hiện các chính sách (Ví dụ: caching).

Trong hệ thống, tầng nghiệp vụ trung gian và dữ liệu giao tiếp với nhau qua việc sử dụng **DataRepository. DataRepository** sẽ chỉ có một thể hiện duy nhất (**singleton pattern). DataRepository** sẽ đóng các chuỗi kết nối (connection string), các kết nối và cấu hình đến các cơ sở dữ liệu khác nhau cũng như các loại cơ sở dữ liệu khác nhau(SQL server, Oracle…). Giống như là kho chứa các nguồn cung ứng dữ liệu tới tất cả các bảng trong cơ sở dữ liệu hoặc các nguồn cung ứng dữ liệu khác, **DataProvider** cung cấp các dịch vụ tương tác với cơ sở dữ liệu cho tầng trung gian thông qua các phương thức đã được đóng gói (**Remote Façade)** mà tầng trung gian không cần quan tâm đến dữ liệu được truy cập như thế nào . Như vậy, tất cả các truy cập đến cơ sở dữ liệu sẽ đều đi qua **DataRepository**,sẽ dễ dàng hơn trong việc:

Giảm bớt lượng mã dư thừa trong quá trình xây dựng các thành phần truy cập cơ sở dữ liêu.

Tập trung các truy cập vào **DataRepository.** Do vậy, việc áp dụng các chính sách như đảm bảo tính nhất quán, caching, ghi nhật ký (logging)… cũng dễ dàng hơn rất nhiều.

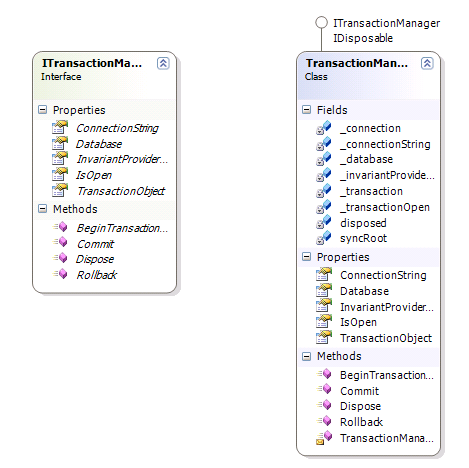


*Một ví dụ về việc giao tiếp giữa các logic nghiệp vụ và tầng truy cập cơ sở dữ liệu*

Như vậy, mỗi khi một thành phần nào đó ở các tầng trên cần đọc dữ liệu sẽ truy cập qua **DataRepository**.

Số lượng truy cập vào **DataRepository** và **Data Access Logic** sẽ tăng lên đáng kể. Kéo theo đó là các tốn kém và rủi ro khi giao tiếp. Để giải quyết việc này, sau khi dữ liệu được lấy ra từ cơ sở dữ liệu, dữ liệu sẽ được lưu vào những file dữ liệu tạm (file nhị phân, file xml) và sẽ lấy ra để đáp ứng khi có yêu cầu đọc (**Data Transfer Object)**. Dữ liệu sẽ được cập nhật khi dữ liệu trong cơ sở dữ liệu thay đổi. Các **Data Transfer Object** đã được gói cùng với các **Business Entities.**

Khi các phương thức truy nhập vào cơ sở dữ liệu để thực thi các nhiệm vụ như đọc, thêm mới, thay đổi xóa dữ liệu rất dễ xảy ra xung đột giữa các yêu cầu khác nhau từ clients, các tiến trình…. Vì vậy, việc quản lý các giao dịch là một yếu tố quan trọng cần phải xét đến trong thiết kế. Để làm việc này, hệ thống đã đóng gói các phương thức, tiện ích thành lớp (class) **TransactionManager.**



*Biểu đồ lớp của* ***TransactionManager***

* + - Một số phương thức và chức năng:
      * **BeginTransaction:** Khi có 1 yêu cầu truy cập vào cơ sở dữ liệu, phương thức khởi tạo và phương thức này sẽ được gọi để bắt đầu một giao dịch mới. (Ngoài ra còn 1 số trường hợp mượn các giao dịch khác).
      * **Commit:** Gửi giao dịch đến nguồn dữ liệu.
      * **Rollback:** Khi việc Commit xảy ra lỗi hay phát sinh ngoại lệ, giao dịch sẽ bị hủy bỏ và dữ liệu sẽ quay trở về trạng thái trước đó.

Trong hệ thống, **Data Provider Layer** có tên là **Agile.Data.**

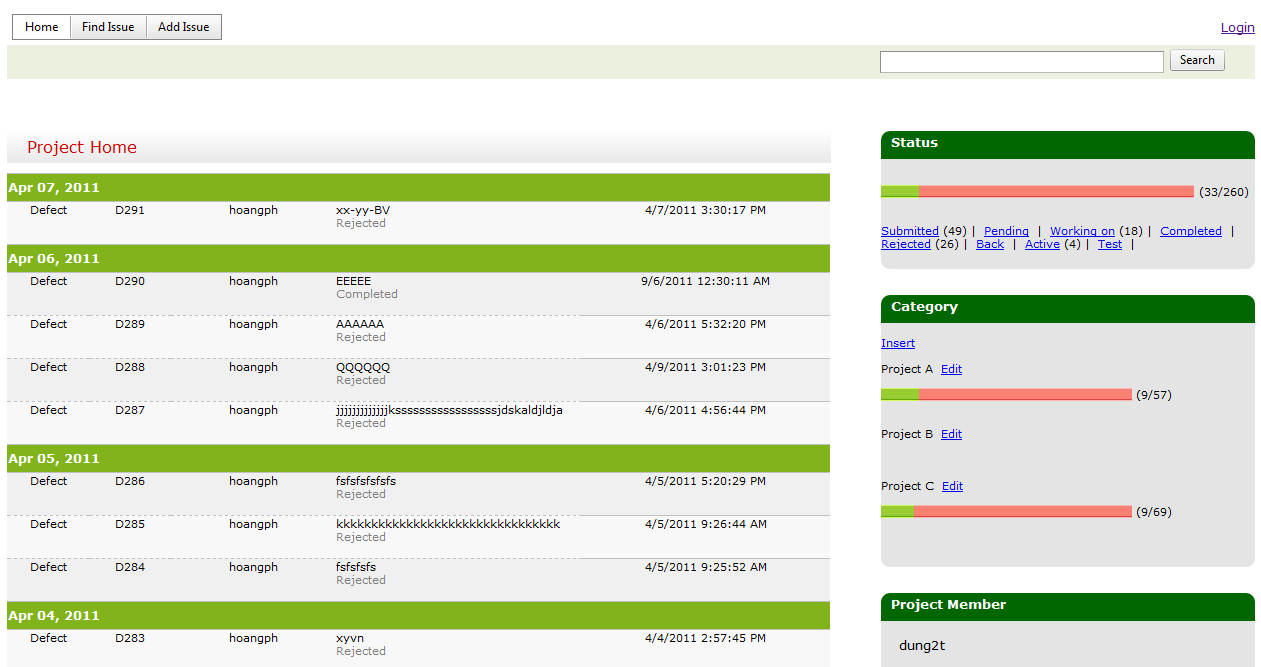
1. Data Access Logic Layer

**Data Access Logic Layer** thao tác trực tiếp với cơ sở dữ liệu. Thực thi các câu lệnh truy vấn cơ sở dữ liệu và trả lại kết quả khi có yêu cầu. Các lớp trong **Data Access Logic Layer** sẽ hiện thực hóa các lớp ảo của **Data Provider Layer** với từng loại cơ sở dữ liệu cụ thể (SQL server, Oracle…).

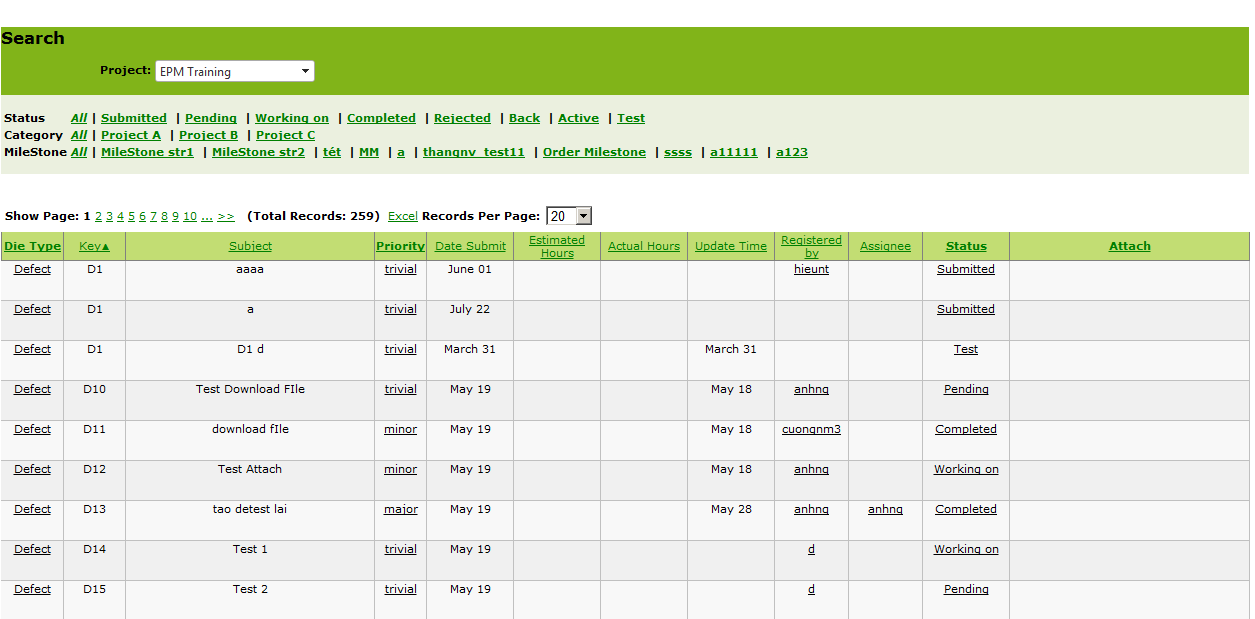
Trong hệ thống, **Data Access Logic** có tên là **Agile.Data.SqlClient**

* + 1. ***Công nghệ sử dụng***
* Microsoft .NET framework
* Microsoft SQL Database
* Nettiers: N-tier application framework
* jQuery.

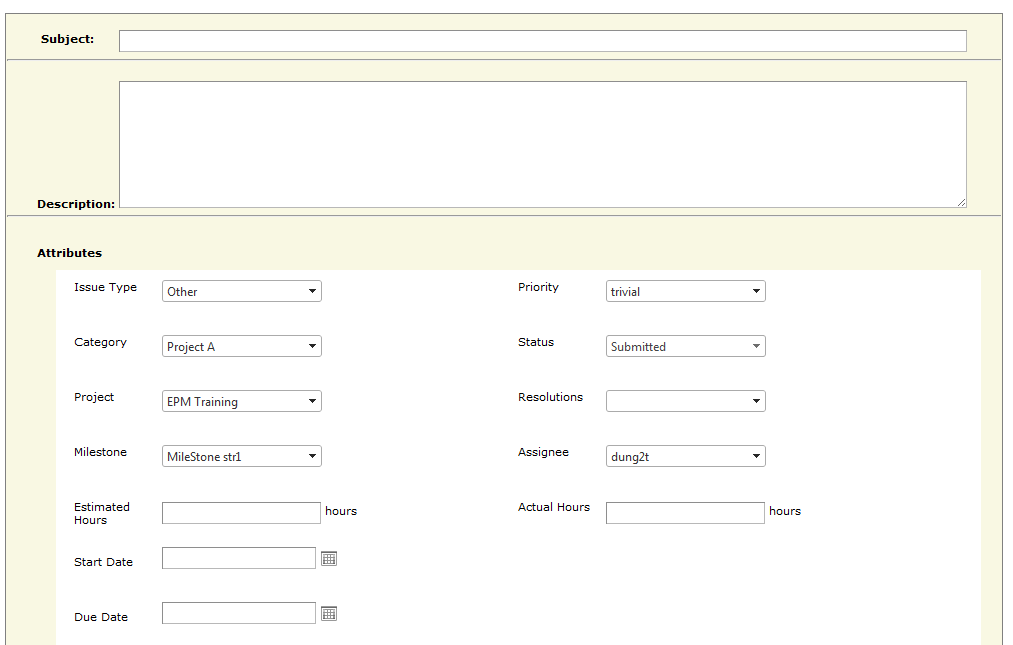
1. **KẾT QUẢ THỰC HIỆN**

****

*Trang chủ*

**

*Trang tìm kiếm issue*

**

*Trang thêm mới/Chỉnh sửa issue*