Phân tích và thiết kế hệ thống thông tin với UML

TS. Dương Kiều Hoa

Tôn Thất Hoà An

Chương 1: TỔNG QUAN VỀ PHÂN TÍCH THIẾT KẾ HỆ THỐNG

ଓ 🕮 ଌ

1- DẪN NHẬP:

1.1- Tính trực quan:

Chúng ta có thể thấy rằng: "Một số tập hợp dữ liệu phức tạp nhất định khi được trình bày bằng đồ thị sẽ truyền tải đến người đọc nhiều thông tin hơn so với các dữ liệu thô". Với phần mềm cũng vậy, khi ngành Công nghiệp của chúng ta ngày càng phát triển, các hệ thống sẽ trở nên phức tạp hơn. Khả năng nắm bắt và kiểm soát sự phức tạp đó của chúng ta đi kèm với khả năng trình bày hệ thống một cách toàn diện - một sự trình bày vượt ra ngoài giới hạn của những dòng lệnh thô. Sự thành công trên thị trường của những ngôn ngữ như Visual Basic và phần giao diện trực quan của C++, Java đã cho thấy sự trình bày trực quan mang tính cốt yếu đối với quá trình phát triển các hệ thống phức tạp.

1.2- Mô hình trừu tượng:

Trước đây, có một thời gian dài, ngành công nghiệp chúng ta đã phải nói tới một "Cuộc khủng hoảng phần mềm". Các cuộc tranh luận đều dựa trên thực tế là chẳng những nhiều đồ án phần mềm không thể sản sinh ra những hệ thống thoả mãn đòi hỏi và nhu cầu của khách hàng, mà còn vượt quá ngân sách và thời hạn. Các công nghệ mới như lập trình hướng đối tượng, lập trình trực quan cũng như các môi trường phát triển tiên tiến có giúp chúng ta nâng cao năng suất lao động, nhưng trong nhiều trường hợp, chúng chỉ hướng tới tầng thấp nhất của việc phát triển phần mềm: phần viết lệnh (coding). Một trong những vấn đề chính của ngành phát triển phần mềm thời nay là có nhiều đồ án bắt tay vào lập trình quá sớm và tập trung quá nhiều vào việc viết code. Lý do một phần là do ban quản trị thiếu hiểu biết về quy trình phát triển phần mềm và họ nảy lo âu khi thấy đội quân lập trình của họ không viết code. Và bản thân các lập trình viên cũng cảm thấy an tâm hơn khi họ ngồi viết code - vốn là tác vụ mà họ quen thuộc! – hơn là khi xây dựng các mô hình trừu tượng cho hệ thống mà họ phải tạo nên.

1.3- Mô hình hóa trực quan:

Mô hình hoá trực quan là một phương thức tư duy về vấn đề sử dụng các mô hình được tổ chức xoay quanh các khái niệm đời thực. Mô hình giúp chúng ta hiểu vấn đề, giao tiếp với mọi người có liên quan đến dự án (khách hàng, chuyên gia lĩnh vực thuộc đề án, nhà phân tích, nhà thiết kế, …). Mô hình rất hữu dụng trong việc mô hình hoá doanh nghiệp, soạn thảo tài liệu, thiết kế chương trình cũng như ngân hàng dữ liệu. Mô hình giúp hiểu các đòi hỏi của hệ thống tốt hơn, tạo các thiết kế rõ ràng hơn và xây dựng nên các hệ thống dễ bảo trì hơn.

Mô hình là kết quả của sự trừu tượng hóa nhằm miêu tả các thành phần cốt yếu của một vấn đề hay một cấu trúc phức tạp qua việc lọc bớt các chi tiết không quan trọng và làm cho vấn đề trở thành dễ hiểu hơn. Trừu tượng hóa là một năng lực căn bản của con người,

cho phép chúng ta giải quyết các vấn đề phức tạp. Các kỹ sư, nghệ sĩ và thợ thủ công đã xây dựng mô hình từ hàng ngàn năm nay để thử nghiệm thiết kế trước khi thực hiện. Phát triển phần mềm cũng không là ngoại lệ. Để xây dựng các hệ thống phức tạp, nhà phát triển phải trừu tượng hóa nhiều hướng nhìn khác nhau của hệ thống, sử dụng ký hiệu chính xác để xây dựng mô hình, kiểm tra xem mô hình có thỏa mãn các đòi hỏi của hệ thống, và dần dần bổ sung thêm chi tiết để chuyển các mô hình thành thực hiện.

Chúng ta xây dựng mô hình cho các hệ thống phức tạp bởi chúng ta không thể hiểu thấu đáo những hệ thống như thế trong trạng thái toàn vẹn của chúng. Khả năng thấu hiểu và nắm bắt tính phức tạp của con người là có hạn. Điều này ta có thể thấy rõ trong ví dụ của ngành xây dựng. Nếu bạn muốn tạo một túp lều ở góc vườn, bạn có thể bắt tay vào xây ngay. Nếu bạn xây một ngôi nhà, có lẽ bạn sẽ cần tới bản vẽ, nhưng nếu bạn muốn xây một toà nhà chọc trời thì chắc chắn bạn không thể không cần bản vẽ. Thế giới phần mềm của chúng ta cũng thế. Chỉ tập trung vào các dòng code hay thậm chí cả phân tích Forms trong Visual Basic chẳng cung cấp một cái nhìn toàn cục về việc phát triển đồ án. Xây dựng mô hình cho phép nhà thiết kế tập trung vào bức tranh lớn về sự tương tác giữa các thành phần trong đồ án, tránh bị sa lầy vào những chi tiết riêng biệt của từng thành phần.

Một môi trường kinh doanh mang tính cạnh tranh gay gắt và luôn luôn thay đổi dẫn đến tính phức tạp ngày càng tăng cao, và tính phức tạp này đặt ra những thách thức đặc trưng cho các nhà phát triển hệ thống. Mô hình giúp chúng ta tổ chức, trình bày trực quan, thấu hiểu và tạo nên các hệ thống phức tạp. Chúng giúp chúng ta đáp ứng các thách thức của việc phát triển phần mềm, hôm nay cũng như ngày mai.

2- MÔ TẢ CHU TRÌNH PHÁT TRIỂN PHẦN MỀM:

2.1- Software Development – một bài toán phức tạp:

Kinh nghiệm của nhiều nhà thiết kế và phát triển cho thấy phát triển phần mềm là một bài toán phức tạp. Xin nêu một số các lý do thường được kể đến:

- Những người phát triển phần mềm rất khó hiểu cho đúng những gì người dùng cần

- Yêu cầu của người dùng thường thay đổi trong thời gian phát triển.

- Yêu cầu thường được miêu tả bằng văn bản, dài dòng, khó hiểu, nhiều khi thậm chí mâu thuẫn.

- Đội quân phát triển phần mềm, vốn là người "ngoài cuộc", rất khó nhận thức thấu đáo các mối quan hệ tiềm ẩn và phức tạp cần được thể hiện chính xác trong các ứng dụng lớn.

- Khả năng nắm bắt các dữ liệu phức tạp của con người (tại cùng một thời điểm) là có hạn.

- Khó định lượng chính xác hiệu suất của thành phẩm và thỏa mãn chính xác sự mong chờ từ phía người dùng.

- Chọn lựa phần cứng và phần mềm thích hợp cho giải pháp là một trong những thách thức lớn đối với Designer.

Phần mềm ngoài ra cần có khả năng ***thích ứng*** và ***mở rộng***. Phần mềm được thiết kế tốt là phần mềm đứng vững trước những biến đổi trong môi trường, dù từ phía cộng đồng người dùng hay từ phía công nghệ. Ví dụ phần mềm đã được phát triển cho một nhà băng cần có khả năng tái sử dụng cho một nhà băng khác với rất ít sửa đổi hoặc hoàn toàn không cần sửa đổi. Phần mềm thoả mãn các yêu cầu đó được coi là phần mềm có khả năng thích ứng.

Một phần mềm có khả năng mở rộng là phần mềm được thiết kế sao cho dễ phát triển theo yêu cầu của người dùng mà không cần sửa chữa nhiều.

Chính vì vậy, một số các khiếm khuyết thường gặp trong phát triển phần mềm là:

- Hiểu không đúng những gì người dùng cần

- Không thể thích ứng cho phù hợp với những thay đổi về yêu cầu đối với hệ thống

- Các Module không khớp với nhau

- Phần mềm khó bảo trì và nâng cấp, mở rộng

- Phát hiện trễ các lỗ hổng của dự án

- Chất lượng phần mềm kém

- Hiệu năng của phần mềm thấp

- Các thành viên trong nhóm không biết được ai đã thay đổi cái gì, khi nào, ở đâu, tại sao phải thay đổi.

2.2- Chu Trình Phát Triển Phần Mềm (Software Development Life Cycle):

Vì phát triển phần mềm là một bài toán khó, nên có lẽ trước hết ta cần điểm qua một số các công việc căn bản của quá trình này. Thường người ta hay tập hợp chúng theo tiến trình thời gian một cách tương đối, xoay quanh chu trình của một phần mềm, dẫn tới kết qủa khái niệm Chu Trình Phát Triển Phần Mềm (Software Development Life Cycle - SDLC) như sau:

Chu Trình Phát Triển Phần Mềm là một chuỗi các hoạt động của nhà phân tích (Analyst), nhà thiết kế (Designer), người phát triển (Developer) và người dùng (User) để phát triển và thực hiện một hệ thống thông tin. Những hoạt động này được thực hiện trong nhiều giai đoạn khác nhau.

Nhà phân tích (Analyst): là người nghiên cứu yêu cầu của khách hàng/người dùng để định nghĩa một phạm vi bài toán, nhận dạng nhu cầu của một tổ chức, xác định xem nhân lực, phương pháp và công nghệ máy tính có thể làm sao để cải thiện một cách tốt nhất công tác của tổ chức này.

Nhà thiết kế (Designer): thiết kế hệ thống theo hướng cấu trúc của database, screens, forms và reports – quyết định các yêu cầu về phần cứng và phần mềm cho hệ thống cần được phát triển.

C**huyên gia lĩnh vực (Domain Experts):** là những người hiểu thực chất vấn đề cùng tất cả những sự phức tạp của hệ thống cần tin học hoá. Họ không nhất thiết phải là nhà lập trình, nhưng họ có thể giúp nhà lập trình hiểu yêu cầu đặt ra đối với hệ thống cần phát triển. Quá trình phát triển phần mềm sẽ có rất nhiều thuận lợi nếu đội ngũ làm phần mềm có được sự trợ giúp của họ.

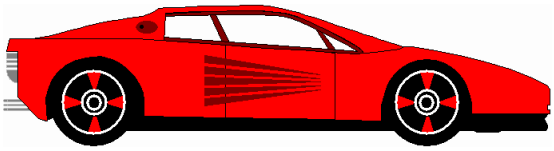
Lập trình viên (Programmer): là những người dựa trên các phân tích và thiết kế để viết chương trình (coding) cho hệ thống bằng ngôn ngữ lập trình đã được thống nhất.

Người dùng (User): là đối tượng phục vụ của hệ thống cần được phát triển.

Để cho rõ hơn, xin lấy ví dụ về một vấn đề đơn giản sau:

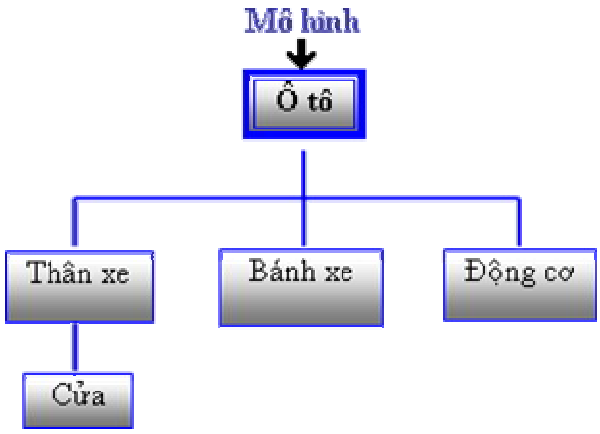
Người bình thường chúng ta khi nhìn một chiếc xe ô tô thường sẽ có một bức tranh từ bên ngoài như sau:

Vấn đề



Hình 1.1: Nhìn vấn đề ô tô của người bình thường

Chuyên gia lĩnh vực sẽ giúp nhà phân tích "trình bày lại" vấn đề như sau:



Hình 1.2: Nhìn vấn đề ô tô của chuyên gia phân tích

Chính vì sự trợ giúp của chuyên gia lĩnh vực có thể đóng vai trò rất quan trọng nên trong những giai đoạn đầu của quá trình phát triển phần mềm, kết quả phân tích nên được thể hiện sao cho dễ hiểu đối với các chuyên gia lĩnh vực. Đây cũng là môt trong rất nhiều lý do khiến cho phương pháp hướng đối tượng được nhiều người hưởng ứng.

2.3- Các giai đoạn của Chu Trình Phát Triển Phần Mềm:

Chu trình của một phần mềm có thể được chia thành các giai đoạn như sau:

- Nghiên cứu sơ bộ (Preliminary Investigation hay còn gọi là Feasibility Study)
- Phân tích yêu cầu (Analysis)
- Thiết kế hệ thống (Design of the System)
- Xây dựng phần mềm (Software Construction)
- Thử nghiệm hệ thống (System Testing)
- Thực hiện, triển khai (System Implementation)
- Bảo trì, nâng cấp (System Maintenance)

a) Nghiên cứu sơ bộ:

Câu hỏi quan trọng nhất khi phát triển một hệ thống hoàn toàn không phải câu hỏi mang tính phương pháp luận. Mà cũng chẳng phải câu hỏi về kỹ thuật. Nó là một câu hỏi dường như có vẻ đơn giản, nhưng thật ra đặc biệt khó trả lời: "Đây có đúng là một hệ thống để thực hiện không?" Đáng buồn là chính câu hỏi này trong thực tế thường chẳng hề được đặt ra và lại càng không được trả lời. Mặc dù việc lầm lẫn về phương pháp hay quyết định sai lầm về kỹ thuật cũng có thể dẫn tới thất bại, nhưng thường thì dự án có thể được cứu vãn nếu có đầy đủ tài nguyên cùng sự cố gắng quên mình của các nhân viên tài

giỏi. Nhưng sẽ chẳng một ai và một điều gì cứu vãn cho một hệ thống phần mềm hoàn toàn chẳng được cần tới hoặc cố gắng tự động hóa một quy trình lầm lạc.

Trước khi bắt tay vào một dự án, bạn phải có một ý tưởng cho nó. Ý tưởng này đi song song với việc nắm bắt các yêu cầu và xuất hiện trong giai đoạn khởi đầu. Nó hoàn tất một phát biểu: "Hệ thống mà chúng ta mong muốn sẽ làm được những việc như sau ....". Trong suốt giai đoạn này, chúng ta tạo nên một bức tranh về ý tưởng đó, rất nhiều giả thuyết sẽ được công nhận hay loại bỏ. Các hoạt động trong thời gian này thường bao gồm thu thập các ý tưởng, nhận biết rủi ro, nhận biết các giao diện bên ngoài, nhận biết các các chức năng chính mà hệ thống cần cung cấp, và có thể tạo một vài nguyên mẫu dùng để "minh chứng các khái niệm của hệ thống". Ý tưởng có thể đến từ nhiều nguồn khác nhau: khách hàng, chuyên gia lĩnh vực, các nhà phát triển khác, chuyên gia về kỹ nghệ, các bản nghiên cứu tính khả thi cũng như việc xem xét các hệ thống khác đang tồn tại. Một khía cạnh cần nhắc tới là code viết trong thời kỳ này thường sẽ bị "bỏ đi", bởi chúng được viết nhằm mục đích thẩm tra hay trợ giúp các giả thuyết khác nhau, chứ chưa phải thứ code được viết theo kết quả phân tích và thiết kế thấu đáo.

Trong giai đoạn nghiên cứu sơ bộ, nhóm phát triển hệ thống cần xem xét các yêu cầu của doanh nghiệp (cần dùng hệ thống), những nguồn tài nguyên có thể sử dụng, công nghệ cũng như cộng đồng người dùng cùng các ý tưởng của họ đối với hệ thống mới. Có thể thực hiện thảo luận, nghiên cứu, xem xét khía cạnh thương mại, phân tích khả năng lời-lỗ, phân tích các trường hợp sử dụng và tạo các nguyên mẫu để xây dựng nên một khái niệm cho hệ thống đích cùng với các mục đích, quyền ưu tiên và phạm vi của nó.

Thường trong giai đoạn này người ta cũng tiến hành tạo một phiên bản thô của lịch trình và kế hoạch sử dụng tài nguyên.

Một giai đoạn nghiên cứu sơ bộ thích đáng sẽ lập nên tập hợp các yêu cầu (dù ở mức độ khái quát cao) đối với một hệ thống khả thi và được mong muốn, kể cả về phương diện kỹ thuật lẫn xã hội. Một giai đoạn nghiên cứu sơ bộ không được thực hiện thoả đáng sẽ dẫn tới các hệ thống không được mong muốn, đắt tiền, bất khả thi và được định nghĩa lầm lạc – những hệ thống thường chẳng được hoàn tất hay sử dụng.

Kết quả của giai đoạn nghiên cứu sơ bộ là Báo Cáo Kết Quả Nghiên Cứu Tính Khả Thi. Khi hệ thống tương lai được chấp nhận dựa trên bản báo cáo này cũng là lúc giai đoạn Phân tích bắt đầu.

b) Phân tích yêu cầu

Sau khi đã xem xét về tính khả thi của hệ thống cũng như tạo lập một bức tranh sơ bộ của dự án, chúng ta bước sang giai đoạn thường được coi là quan trọng nhất trong các công việc lập trình: hiểu hệ thống cần xây dựng. Người thực hiện công việc này là nhà phân tích.

Quá trình phân tích nhìn chung là hệ quả của việc trả lời câu hỏi "Hệ thống cần phải làm gì?". Quá trình phân tích bao gồm việc nghiên cứu chi tiết hệ thống doanh nghiệp hiện

thời, tìm cho ra nguyên lý hoạt động của nó và những vị trí có thể được nâng cao, cải thiện. Bên cạnh đó là việc nghiên cứu xem xét các chức năng mà hệ thống cần cung cấp và các mối quan hệ của chúng, bên trong cũng như với phía ngoài hệ thống. Trong toàn bộ giai đoạn này, nhà phân tích và người dùng cần cộng tác mật thiết với nhau để xác định các yêu cầu đối với hệ thống, tức là các tính năng mới cần phải được đưa vào hệ thống.

Những mục tiêu cụ thể của giai đoạn phân tích là:

- Xác định hệ thống cần phải làm gì.

- Nghiên cứu thấu đáo tất cả các chức năng cần cung cấp và những yếu tố liên quan

- Xây dựng một mô hình nêu bật bản chất vấn đề từ một hướng nhìn có thực (trong đời sống thực).

- Trao định nghĩa vấn đề cho chuyên gia lĩnh vực để nhận sự đánh giá, góp ý.

- Kết quả của giai đoạn phân tích là bản Đặc Tả Yêu Cầu (Requirements Specifications).

c) Thiết kế hệ thống

Sau giai đoạn phân tích, khi các yêu cầu cụ thể đối với hệ thống đã được xác định, giai đoạn tiếp theo là thiết kế cho các yêu cầu mới. Công tác thiết kế xoay quanh câu hỏi chính: Hệ thống làm cách nào để thỏa mãn các yêu cầu đã được nêu trong Đặc Tả Yêu Cầu?

Một số các công việc thường được thực hiện trong giai đoạn thiết kế:

- Nhận biết form nhập liệu tùy theo các thành phần dữ liệu cần nhập.

- Nhận biết reports và những output mà hệ thống mới phải sản sinh

- Thiết kế forms (vẽ trên giấy hay máy tính, sử dụng công cụ thiết kế)

- Nhận biết các thành phần dữ liệu và bảng để tạo database

- Ước tính các thủ tục giải thích quá trình xử lý từ input đến output.

Kết quả giai đoạn thiết kế là Đặc Tả Thiết Kế (Design Specifications). Bản Đặc Tả Thiết Kế Chi Tiết sẽ được chuyển sang cho các lập trình viên để thực hiện giai đoạn xây dựng phần mềm.

d) Xây dựng phần mềm

Đây là giai đoạn viết lệnh (code) thực sự, tạo hệ thống. Từng người viết code thực hiện những yêu cầu đã được nhà thiết kế định sẵn. Cũng chính người viết code chịu trách nhiệm viết tài liệu liên quan đến chương trình, giải thích thủ tục (procedure) mà anh ta tạo nên được viết như thế nào và lý do cho việc này.

Để đảm bảo chương trình được viết nên phải thoả mãn mọi yêu cầu có ghi trước trong bản Đặc Tả Thiết Kế Chi Tiết, người viết code cũng đồng thời phải tiến hành thử nghiệm phần chương trình của mình. Phần thử nghiệm trong giai đoạn này có thể được chia thành hai bước chính:

Thử nghiệm đơn vị:

Người viết code chạy thử các phần chương trình của mình với dữ liệu giả (test/dummy data). Việc này được thực hiện theo một kế hoạch thử, cũng do chính người viết code soạn ra. Mục đích chính trong giai đoạn thử này là xem chương trình có cho ra những kết quả mong đợi. Giai đoạn thử nghiệm đơn vị nhiều khi được gọi là "Thử hộp trắng" (White Box Testing)

Thử nghiệm đơn vị độc lập:

Công việc này do một thành viên khác trong nhóm đảm trách. Cần chọn người không có liên quan trực tiếp đến việc viết code của đơn vị chương trình cần thử nghiệm để đảm bảo tính "độc lập". Công việc thử đợt này cũng được thực hiện dựa trên kế hoạch thử do người viết code soạn nên.

e) Thử nghiệm hệ thống

Sau khi các thủ tục đã được thử nghiệm riêng, cần phải thử nghiệm toàn bộ hệ thống. Mọi thủ tục được tích hợp và chạy thử, kiểm tra xem mọi chi tiết ghi trong Đặc Tả Yêu Cầu và những mong chờ của người dùng có được thoả mãn. Dữ liệu thử cần được chọn lọc đặc biệt, kết quả cần được phân tích để phát hiện mọi lệch lạc so với mong chờ.

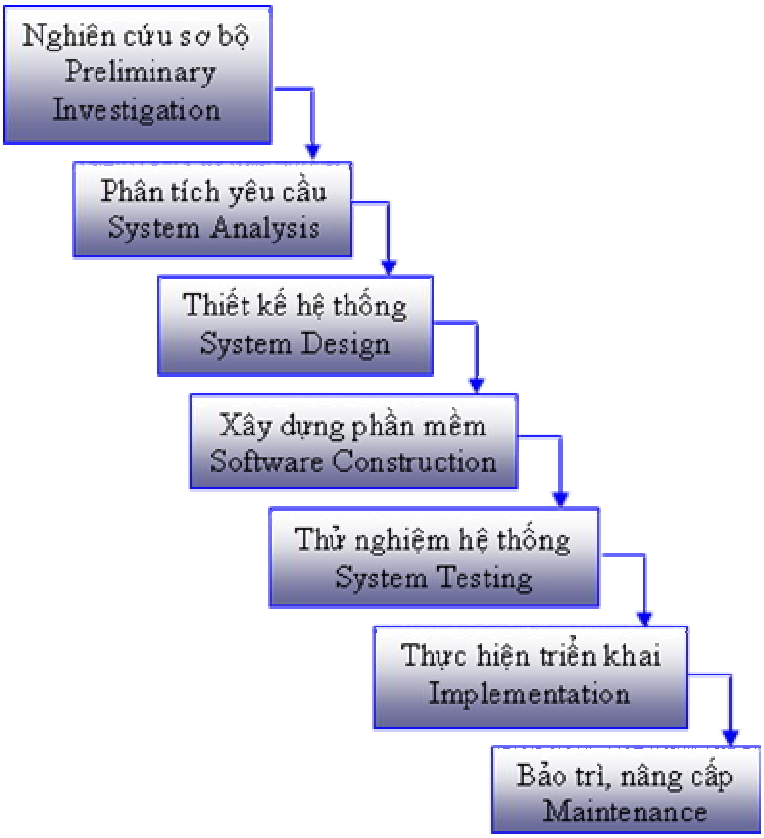
f) Thực hiện, triển khai

Trong giai đoạn này, hệ thống vừa phát triển sẽ được triển khai sao cho phía người dùng. Trước khi để người dùng thật sự bắt tay vào sử dụng hệ thống, nhóm các nhà phát triển cần tạo các file dữ liệu cần thiết cũng như huấn luyện cho người dùng, để đảm bảo hệ thống được sử dụng hữu hiệu nhất.

g) Bảo trì, nâng cấp

Tùy theo các biến đổi trong môi trường sử dụng, hệ thống có thể trở nên lỗi thời hay cần phải được sửa đổi nâng cấp để sử dụng có hiệu quả. Hoạt động bảo trì hệ thống có thể rất khác biệt tùy theo mức độ sửa đổi và nâng cấp cần thiết.

Sơ đồ tổng quát các giai đoạn của Chu Trình Phát Triển Phần Mềm:



Hình 1.3: Sơ đồ tổng quát các giai đoạn của Chu Trình Phát Triển Phần Mềm

3- PHƯƠNG PHÁP HƯỚNG CHỨC NĂNG VÀ PHƯƠNG PHÁP HƯỚNG ĐỐI TƯỢNG:

3.1- Phương pháp hướng chức năng:

Đây là lối tiếp cận truyền thống của ngành Công nghệ phần mềm. Theo lối tiếp cận này, chúng ta quan tâm chủ yếu tới những thông tin mà hệ thống sẽ giữ gìn. Chúng ta hỏi người dùng xem họ sẽ cần những thông tin nào, rồi chúng ta thiết kế ngân hàng dữ liệu để chứa những thông tin đó, cung cấp Forms để nhập thông tin và in báo cáo để trình bày các thông tin. Nói một cách khác, chúng ta tập trung vào thông tin và không mấy để ý đến những gì có thể xảy ra với những hệ thống đó và cách hoạt động (ứng xử) của hệ thống là ra sao. Đây là lối tiệm cận xoay quanh dữ liệu và đã được áp dụng để tạo nên hàng ngàn hệ thống trong suốt nhiều năm trời.

Lối tiếp cận xoay quanh dữ liệu là phương pháp tốt cho việc thiết kế ngân hàng dữ liệu và nắm bắt thông tin, nhưng nếu áp dụng cho việc thiết kế ứng dụng lại có thể khiến phát sinh nhiều khó khăn. Một trong những thách thức lớn là yêu cầu đối với các hệ thống thường xuyên thay đổi. Một hệ thống xoay quanh dữ liệu có thể dễ dàng xử lý việc thay

đổi ngân hàng dữ liệu, nhưng lại khó thực thi những thay đổi trong nguyên tắc nghiệp vụ hay cách hoạt động của hệ thống.

Phương pháp hướng đối tượng đã được phát triển để trả lời cho vấn đề đó. Với lối tiếp cận hướng đối tượng, chúng ta tập trung vào cả hai mặt của vấn đề : thông tin và cách hoạt động.

3.2- Phương pháp hướng đối tượng:

Hướng đối tượng là thuật ngữ thông dụng hiện thời của ngành công nghiệp phần mềm. Các công ty đang nhanh chóng tìm cách áp dụng và tích hợp công nghệ mới này vào các ứng dụng của họ. Thật sự là đa phần các ứng dụng hiện thời đều mang tính hướng đối tượng. Nhưng hướng đối tượng có nghĩa là gì?

Lối tiếp cận hướng đối tượng là một lối tư duy về vấn đề theo lối ánh xạ các thành phần trong bài toán vào các đối tượng ngoài đời thực. Với lối tiếp cận này, chúng ta chia ứng dụng thành các thành phần nhỏ, gọi là các đối tượng, chúng tương đối độc lập với nhau. Sau đó ta có thể xây dựng ứng dụng bằng cách chắp các đối tượng đó lại với nhau. Hãy nghĩ đến trò chơi xây lâu đài bằng các mẫu gỗ. Bước đầu tiên là tạo hay mua một vài loại mẫu gỗ căn bản, từ đó tạo nên các khối xây dựng căn bản của mình. Một khi đã có các khối xây dựng đó, bạn có thể chắp ráp chúng lại với nhau để tạo lâu đài. Tương tự như vậy một khi đã xây dựng một số đối tượng căn bản trong thế giới máy tính, bạn có thể chắp chúng lại với nhau để tạo ứng dụng của mình.

Xin lấy một ví dụ đơn giản: vấn đề rút tiền mặt tại nhà băng. Các "mẫu gỗ" thành phần ở đây sẽ là ánh xạ của các đối tượng ngoài đời thực như tài khoản, nhân viên, khách hàng, …Và ứng dụng sẽ được sẽ được nhận diện cũng như giải đáp xoay quanh các đối tượng đó.

4- ƯU ĐIỂM CỦA MÔ HÌNH HƯỚNG ĐỐI TƯỢNG:

4.1- Tính tái sử dụng (Reusable)

Phương pháp phân tích và thiết kế hướng đối tượng thực hiện theo các thuật ngữ và khái niệm của phạm vi lĩnh vực ứng dụng (tức là của doanh nghiệp hay đơn vị mà hệ thống tương lai cần phục vụ), nên nó tạo sự tiếp cận tương ứng giữa hệ thống và vấn đề thực ngoài đời. Trong ví dụ bán xe ô tô, mọi giai đoạn phân tích thiết kế và thực hiện đều xoay quanh các khái niệm như khách hàng, nhân viên bán hàng, xe ô tô, … Vì quá trình phát triển phần mềm đồng thời là quá trình cộng tác của khách hàng/người dùng, nhà phân tích, nhà thiết kế, nhà phát triển, chuyên gia lĩnh vực, chuyên gia kỹ thuật, ... nên lối tiếp cận này khiến cho việc giao tiếp giữa họ với nhau được dễ dàng hơn.

Một trong những ưu điểm quan trọng bậc nhất của phương pháp phân tích và thiết kế hướng đối tượng là tính tái sử dụng: bạn có thể tạo các thành phần (đối tượng) một lần và dùng chúng nhiều lần sau đó. Giống như việc bạn có thể tái sử dụng các khối xây dựng (hay bản sao của nó) trong một toà lâu đài, một ngôi nhà ở, một con tàu vũ trụ, bạn cũng

có thể tái sử dụng các thành phần (đối tượng) căn bản trong các thiết kế hướng đối tượng cũng như code của một hệ thống kế toán, hệ thống kiểm kê, hoặc một hệ thống đặt hàng.

Vì các đối tượng đã được thử nghiệm kỹ càng trong lần dùng trước đó, nên khả năng tái sử dụng đối tượng có tác dụng giảm thiểu lỗi và các khó khăn trong việc bảo trì, giúp tăng tốc độ thiết kế và phát triển phần mềm.

Phương pháp hướng đối tượng giúp chúng ta xử lý các vấn đề phức tạp trong phát triển phần mềm và tạo ra các thế hệ phần mềm có khả năng thích ứng và bền chắc.

4.2- Các giai đoạn của chu trình phát triển phần mềm với mô hình hướng đối tượng:

Phân tích hướng đối tượng (Object Oriented Analysis - OOA):

Là giai đoạn phát triển một mô hình chính xác và súc tích của vấn đề, có thành phần là các đối tượng và khái niệm đời thực, dễ hiểu đối với người sử dụng.

Trong giai đoạn OOA, vấn đề được trình bày bằng các thuật ngữ tương ứng với các đối tượng có thực. Thêm vào đó, hệ thống cần phải được định nghĩa sao cho người không chuyên Tin học có thể dễ dàng hiểu được.

Dựa trên một vấn đề có sẵn, nhà phân tích cần ánh xạ các đối tượng hay thực thể có thực như khách hàng, ô tô, người bán hàng, … vào thiết kế để tạo ra được bản thiết kế gần cận với tình huống thực. Mô hình thiết kế sẽ chứa các thực thể trong một vấn đề có thực và giữ nguyên các mẫu hình về cấu trúc, quan hệ cũng như hành vi của chúng. Nói một cách khác, sử dụng phương pháp hướng đối tượng chúng ta có thể mô hình hóa các thực thể thuộc một vấn đề có thực mà vẫn giữ được cấu trúc, quan hệ cũng như hành vi của chúng.

Đối với ví dụ một phòng bán ô tô, giai đoạn OOA sẽ nhận biết được các thực thể như:

- Khách hàng
- Người bán hàng
- Phiếu đặt hàng
- Phiếu (hoá đơn) thanh toán
- Xe ô tô

Tương tác và quan hệ giữa các đối tượng trên là:

- Người bán hàng dẫn khách hàng tham quan phòng trưng bày xe.
- Khách hàng chọn một chiếc xe

- Khách hàng viết phiếu đặt xe

- Khách hàng trả tiền xe

- Xe ô tô được giao đến cho khách hàng

Đối với ví dụ nhà băng lẻ, giai đoạn OOA sẽ nhận biết được các thực thể như:

- Loại tài khoản: ATM (rút tiền tự động), Savings (tiết kiệm), Current (bình thường), Fixed (đầu tư), ...

- Khách hàng

- Nhân viên

- Phòng máy tính.

Tương tác và quan hệ giữa các đối tượng trên:

- Một khách hàng mới mở một tài khoản tiết kiệm

- Chuyển tiền từ tài khoản tiết kiệm sang tài khoản đầu tư

- Chuyển tiền từ tài khoản tiết kiệm sang tài khoản ATM

Xin chú ý là ở đây, như đã nói, ta chú ý đến cả ***hai*** khía cạnh: thông tin và cách hoạt động của hệ thống (tức là những gì có thể xảy ra với những thông tin đó).

Lối phân tích bằng kiểu ánh xạ "đời thực" vào máy tính như thế thật sự là ưu điểm lớn của phương pháp hướng đối tượng.

Thiết kế hướng đối tượng (Object Oriented Design - OOD):

Là giai đoạn tổ chức chương trình thành các tập hợp đối tượng cộng tác, mỗi đối tượng trong đó là thực thể của một lớp. Các lớp là thành viên của một cây cấu trúc với mối quan hệ thừa kế.

Mục đích của giai đoạn OOD là tạo thiết kế dựa trên kết quả của giai đoạn OOA, dựa trên những quy định phi chức năng, những yêu cầu về môi trường, những yêu cầu về khả năng thực thi, ... OOD tập trung vào việc cải thiện kết quả của OOA, tối ưu hóa giải pháp đã được cung cấp trong khi vẫn đảm bảo thoả mãn tất cả các yêu cầu đã được xác lập.

Trong giai đoạn OOD, nhà thiết kế định nghĩa các chức năng, thủ tục (operations), thuộc tính (attributes) cũng như mối quan hệ của một hay nhiều lớp (class) và quyết định chúng cần phải được điều chỉnh sao cho phù hợp với môi trường phát triển. Đây cũng là giai đoạn để thiết kế ngân hàng dữ liệu và áp dụng các kỹ thuật tiêu chuẩn hóa.

Về cuối giai đoạn OOD, nhà thiết kế đưa ra một loạt các biểu đồ (diagram) khác nhau. Các biểu đồ này có thể được chia thành hai nhóm chính là Tĩnh và động. Các biểu đồ tĩnh biểu thị các lớp và đối tượng, trong khi biểu đồ động biểu thị tương tác giữa các lớp và phương thức hoạt động chính xác của chúng. Các lớp đó sau này có thể được nhóm thành các gói (Packages) tức là các đơn vị thành phần nhỏ hơn của ứng dụng.

Lập trình hướng đối tượng (Object Oriented Programming - OOP):

Giai đoạn xây dựng phần mềm có thể được thực hiện sử dụng kỹ thuật lập trình hướng đối tượng. Đó là phương thức thực hiện thiết kế hướng đối tượng qua việc sử dụng một ngôn ngữ lập trình có hỗ trợ các tính năng hướng đối tượng. Một vài ngôn ngữ hướng đối tượng thường được nhắc tới là C++ và Java. Kết quả chung cuộc của giai đoạn này là một loạt các code chạy được, nó chỉ được đưa vào sử dụng sau khi đã trải qua nhiều vòng quay của nhiều bước thử nghiệm khác nhau.

PHẦN CÂU HỎI

Hỏi: Một số tập hợp dữ liệu phức tạp nhất định khi được trình bày bằng đồ thị sẽ truyền tải đến người đọc nhiều thông tin hơn so với các dữ liệu thô?

Đáp: Đúng

Hỏi: Mô hình giúp chúng ta tổ chức, trình bày trực quan, thấu hiểu và tạo nên các hệ thống phức tạp.

Đáp: Đúng

Hỏi: Ưu điểm lớn nhất của mô hình hướng đối tượng là tính tái sử dụng (Reusable)?

Đáp: Đúng.

Chương 2: NGÔN NGỮ MÔ HÌNH HOÁ THỐNG NHẤT LÀ GÌ

ଓ 🕮 ଌ

1- GIỚI THIỆU UML:

1.1- Mô hình hóa hệ thống phần mềm:

Như đã trình bày ở phần trước, mục tiêu của giai đoạn phân tích hệ thống là sản xuất ra một mô hình tổng thể của hệ thống cần xây dựng. Mô hình này cần phải được trình bày theo hướng nhìn (View) của khách hàng hay người sử dụng và làm sao để họ hiểu được. Mô hình này cũng có thể được sử dụng để xác định các yêu cầu của người dùng đối với hệ thống và qua đó giúp chúng ta đánh giá tính khả thi của dự án.

Tầm quan trọng của mô hình đã được lĩnh hội một cách thấu đáo trong hầu như tất cả các ngành khoa học kỹ thuật từ nhiều thế kỷ nay. Bất kỳ ở đâu, khi muốn xây dựng một vật thể nào đó, đầu tiên người ta đã tạo nên các bản vẽ để quyết định cả ngoại hình lẫn phương thức hoạt động của nó. Chẳng hạn các bản vẽ kỹ thuật thường gặp là một dạng mô hình quen thuộc. Mô hình nhìn chung là một cách mô tả của một vật thể nào đó. Vật đó có thể tồn tại trong một số giai đoạn nhất định, dù đó là giai đoạn thiết kế hay giai đoạn xây dựng hoặc chỉ là một kế hoạch. Nhà thiết kế cần phải tạo ra các mô hình mô tả tất cả các khía cạnh khác nhau của sản phẩm. Ngoài ra, một mô hình có thể được chia thành nhiều hướng nhìn, mỗi hướng nhìn trong số chúng sẽ mô tả một khía cạnh riêng biệt của sản phẩm hay hệ thống cần được xây dựng. Một mô hình cũng có thể được xây dựng trong nhiều giai đoạn và ở mỗi giai đoạn, mô hình sẽ được bổ sung thêm một số chi tiết nhất định.

Mô hình thường được mô tả trong ngôn ngữ trực quan, điều đó có nghĩa là đa phần các thông tin được thể hiện bằng các ký hiệu đồ họa và các kết nối giữa chúng, chỉ khi cần thiết một số thông tin mới được biểu diễn ở dạng văn bản; Theo đúng như câu ngạn ngữ "Một bức tranh nói nhiều hơn cả ngàn từ". Tạo mô hình cho các hệ thống phần mềm trước khi thực sự xây dựng nên chúng, đã trở thành một chuẩn mực trong việc phát triển phần mềm và được chấp nhận trong cộng đồng làm phần mềm giống như trong bất kỳ một ngành khoa học kỹ thuật nào khác. Việc biểu diễn mô hình phải thoã mãn các yếu tố sau:

- Chính xác (accurate): Mô tả đúng hệ thống cần xây dựng.

- Đồng nhất (consistent): Các view khác nhau không được mâu thuẩn với nhau.

- Có thể hiểu được (understandable): Cho những người xây dựng lẫn sử dụng

- Dễ thay đổi (changeable)

- Dễ dàng liên lạc với các mô hình khác.

Có thể nói thêm rằng mô hình là một sự đơn giản hoá hiện thực. Mô hình được xây dựng nên để chúng ta dễ dàng hiểu và hiểu tốt hơn hệ thống cần xây dựng. Tạo mô hình sẽ giúp cho chúng ta hiểu thấu đáo một hệ thống phức tạp trong sự toàn thể của nó.

Nói tóm lại, mô hình hóa một hệ thống nhằm mục đích:

- Hình dung một hệ thống theo thực tế hay theo mong muốn của chúng ta .

- Chỉ rõ cấu trúc hoặc ứng xử của hệ thống.

- Tạo một khuôn mẫu hướng dẫn nhà phát triển trong suốt quá trình xây dựng hệ thống.

- Ghi lại các quyết định của nhà phát triển để sử dụng sau này.

1.2- Trước khi UML ra đời:

Đầu những năm 1980, ngành công nghệ phần mềm chỉ có duy nhất một ngôn ngữ hướng đối tượng là Simula. Sang nửa sau của thập kỷ 1980, các ngôn ngữ hướng đối tượng như Smalltalk và C++ xuất hiện. Cùng với chúng, nảy sinh nhu cầu mô hình hoá các hệ thống phần mềm theo hướng đối tượng. Và một vài trong số những ngôn ngữ mô hình hoá xuất hiện những năm đầu thập kỷ 90 được nhiều người dùng là:

- Grady Booch’s Booch Modeling Methodology

- James Rambaugh’s Object Modeling Technique – OMT

- Ivar Jacobson’s OOSE Methodology

- Hewlett- Packard’s Fusion

- Coad and Yordon’s OOA and OOD

Mỗi phương pháp luận và ngôn ngữ trên đều có hệ thống ký hiệu riêng, phương pháp xử lý riêng và công cụ hỗ trợ riêng, khiến nảy ra cuộc tranh luận phương pháp nào là tốt nhất. Đây là cuộc tranh luận khó có câu trả lời, bởi tất cả các phương pháp trên đều có những điểm mạnh và điểm yếu riêng. Vì thế, các nhà phát triển phần mềm nhiều kinh nghiệm thường sử dụng phối hợp các điểm mạnh của mỗi phương pháp cho ứng dụng của mình. Trong thực tế, sự khác biệt giữa các phương pháp đó hầu như không đáng kể và theo cùng tiến trình thời gian, tất cả những phương pháp trên đã tiệm cận lại và bổ sung lẫn cho nhau. Chính hiện thực này đã được những người tiên phong trong lĩnh vực mô hình hoá hướng đối tượng nhận ra và họ quyết định ngồi lại cùng nhau để tích hợp những điểm mạnh của mỗi phương pháp và đưa ra một mô hình thống nhất cho lĩnh vực công nghệ phần mềm.

1.3- Sự ra đời của UML:

Trong bối cảnh trên, người ta nhận thấy cần thiết phải cung cấp một phương pháp tiệm cận được chuẩn hoá và thống nhất cho việc mô hình hoá hướng đối tượng. Yêu cầu cụ thể là đưa ra một tập hợp chuẩn hoá các ký hiệu (Notation) và các biểu đồ (Diagram) để nắm bắt các quyết định về mặt thiết kế một cách rõ ràng, rành mạch. Đã có ba công trình tiên phong nhắm tới mục tiêu đó, chúng được thực hiện dưới sự lãnh đạo của James Rumbaugh, Grady Booch và Ivar Jacobson. Chính những cố gắng này dẫn đến kết quả là xây dựng được một Ngôn Ngữ Mô Hình Hoá Thống Nhất (Unifield Modeling Language – UML).

UML là một ngôn ngữ mô hình hoá thống nhất có phần chính bao gồm những ký hiệu hình học, được các phương pháp hướng đối tượng sử dụng để thể hiện và miêu tả các thiết kế của một hệ thống. Nó là một ngôn ngữ để đặc tả, trực quan hoá, xây dựng và làm sưu liệu cho nhiều khía cạnh khác nhau của một hệ thống có nồng độ phần mềm cao. UML có thể được sử dụng làm công cụ giao tiếp giữa người dùng, nhà phân tích, nhà thiết kế và nhà phát triển phần mềm.

Trong quá trình phát triển có nhiều công ty đã hỗ trợ và khuyến khích phát triển UML có thể kể tới như : Hewlett Packard, Microsoft, Oracle, IBM, Unisys.

1.4- UML (Unifield Modeling Language):

Ngôn ngữ mô hình hóa thống nhất (Unifield Modeling Language – UML) là một ngôn ngữ để biểu diễn mô hình theo hướng đối tượng được xây dựng bởi ba tác giả trên với chủ đích là:

- Mô hình hoá các hệ thống sử dụng các khái niệm hướng đối tượng.

- Thiết lập một kết nối từ nhận thức của con người đến các sự kiện cần mô hình hoá.

- Giải quyết vấn đề về mức độ thừa kế trong các hệ thống phức tạp, có nhiều ràng buộc khác nhau.

- Tạo một ngôn ngữ mô hình hoá có thể sử dụng được bởi người và máy.

1.5- Phương pháp và các ngôn ngữ mô hình hoá:

Phương pháp hay phương thức (method) là một cách trực tiếp cấu trúc hoá sự suy nghĩ và hành động của con người. Phương pháp cho người sử dụng biết phải làm gì, làm như thế nào, khi nào và tại sao (mục đích của hành động). Phương pháp chứa các mô hình (model), các mô hình được dùng để mô tả những gì sử dụng cho việc truyền đạt kết quả trong quá trình sử dụng phương pháp. Điểm khác nhau chính giữa một phương pháp và một ngôn ngữ mô hình hoá (modeling language) là ngôn ngữ mô hình hoá không có một tiến trình (process) hay các câu lệnh (instruction) mô tả những công việc người sử dụng cần làm.

Một mô hình được biểu diễn theo một ngôn ngữ mô hình hoá. Ngôn ngữ mô hình hoá bao gồm các ký hiệu – những biểu tượng được dùng trong mô hình – và một tập các quy tắc chỉ cách sử dụng chúng. Các quy tắc này bao gồm:

- Syntactic (Cú pháp): cho biết hình dạng các biểu tượng và cách kết hợp chúng trong ngôn ngữ.

- Semantic (Ngữ nghĩa): cho biết ý nghĩa của mỗi biểu tượng, chúng được hiểu thế nào khi nằm trong hoặc không nằm trong ngữ cảnh của các biểu tượng khác.

- Pragmatic : định nghĩa ý nghĩa của biểu tượng để sao cho mục đích của mô hình được thể hiện và mọi người có thể hiểu được.

2- UML TRONG PHÂN TÍCH THIẾT KẾ HỆ THỐNG:

UML có thể được sử dụng trong nhiều giai đoạn, từ phát triển, thiết kế cho tới thực hiện và bảo trì. Vì mục đích chính của ngôn ngữ này là dùng các biểu đồ hướng đối tượng để mô tả hệ thống nên miền ứng dụng của UML bao gồm nhiều loại hệ thống khác nhau như:

- **Hệ thống thống tin** (Information System): Cất giữ, lấy, biến đổi biểu diễn thông tin cho người sử dụng. Xử lý những khoảng dữ liệu lớn có các quan hệ phức tạp , mà chúng được lưu trữ trong các cơ sở dữ liệu quan hệ hay hướng đối tượng .

- **Hệ thống kỹ thuật** (Technical System): Xử lý và điều khiển các thiết bị kỹ thuật như viễn thông, hệ thống quân sự, hay các quá trình công nghiệp. Đây là loại thiết bị phải xử lý các giao tiếp đặc biệt , không có phần mềm chuẩn và thường là các hệ thống thời gian thực (real time).

- **Hệ thống nhúng** (Embeded System): Thực hiện trên phần cứng gắn vào các thiết bị như điện thoại di động, điều khiển xe hơi, … Điều này được thực hiện bằng việc lập trình mức thấp với hỗ trợ thời gian thực. Những hệ thống này thường không có các thiết bị như màn hình đĩa cứng, …

- **Hệ thống phân bố** (Distributed System): Được phân bố trên một số máy cho phép truyền dữ liệu từ nơi này đến nơi khác một cách dễ dàng. Chúng đòi hỏi các cơ chế liên lạc đồng bộ để đảm bảo toàn vẹn dữ liệu và thường được xây dựng trên một số các kỹ thuật đối tượng như CORBA, COM/DCOM, hay Java Beans/RMI.

- **Hệ thống Giao dịch** (Business System): Mô tả mục đích, tài nguyên (con người, máy tính, …), các quy tắc (luật pháp, chiến thuật kinh doanh, cơ chế, …), và công việc hoạt động kinh doanh.

- **Phần mềm hệ thống** (System Software): Định nghĩa cơ sở hạ tầng kỹ thuật cho phần mềm khác sử dụng, chẳng hạn như hệ điều hành, cơ sở dữ liệu, giao diện người sử dụng.

3- UML VÀ CÁC GIAI ĐOẠN PHÁT TRIỂN HỆ THỐNG

Preliminary Investigation: use cases thể hiện các yêu cầu của người dùng. Phần miêu tả use case xác định các yêu cầu, phần diagram thể hiện mối quan hệ và giao tiếp với hệ thống.

Analysis: Mục đích chính của giai đoạn này là trừu tượng hóa và tìm hiểu các cơ cấu có trong phạm vi bài toán. Class diagrams trên bình diện trừu tượng hóa các thực thể ngoài đời thực được sử dụng để làm rõ sự tồn tại cũng như mối quan hệ của chúng. Chỉ những lớp (class) nằm trong phạm vi bài toán mới đáng quan tâm.

Design: Kết quả phần analysis được phát triển thành giải pháp kỹ thuật. Các lớp được mô hình hóa chi tiết để cung cấp hạ tầng kỹ thuật như giao diện, nền tảng cho database, … Kết quả phần Design là các đặc tả chi tiết cho giai đoạn xây dựng phần mềm.

Development: Mô hình Design được chuyển thành code. Programmer sử dụng các UML diagrams trong giai đoạn Design để hiểu vấn đề và tạo code.

Testing: Sử dụng các UML diagrams trong các giai đoạn trước. Có 4 hình thức kiểm tra hệ thống:

- *Unit testing* (class diagrams & class specifications) : kiểm tra từng đơn thể, được dùng để kiểm tra các lớp hay các nhóm đơn thể.

- *Integration testing* (integration diagrams & collaboration diagrams) : kiểm tra tích hợp là kiểm tra kết hợp các component với các lớp để xem chúng hoạt động với nhau có đúng không.

- *System testing* (use-case diagrams) : kiểm tra xem hệ thống có đáp ứng được chức năng mà người sử dụng yêu cầu hay không.

- *Acceptance testing*: Kiểm tra tính chấp nhận được của hệ thống, thường được thực hiện bởi khách hàng, việc kiểm tra này thực hiện tương tự như kiểm tra hệ thống.

PHẦN CÂU HỎI

Hỏi: UML (Unifield Modeling Language) là gì?

Đáp: Ngôn ngữ mô hình hóa thống nhất – UML là một ngôn ngữ để biểu diễn mô hình theo hướng đối tượng.

Hỏi: Điểm khác nhau cơ bản giữa phương pháp (method) và một ngôn ngữ mô hình hoá (modeling language) là gì?

Đáp: Điểm khác nhau cơ bản giữa một phương pháp và một ngôn ngữ mô hình hoá là ngôn ngữ mô hình hoá không có một tiến trình (process) hay các câu lệnh (instruction) mô tả những công việc người sử dụng cần làm mà nó bao gồm các ký hiệu – những biểu tượng được dùng trong mô hình – và một tập các quy tắc chỉ cách sử dụng chúng.