BÀI 1. TỔNG QUAN VỀ ĐẶC TẢ CÁC YÊU CẦU CỦA HỆ THỐNG

- **Mục đích:** Cung cấp cho sinh viên kiến thức về các giai đoạn và các phương pháp khảo sát yêu cầu hệ thống, và các yêu chung đối với báo cáo khả thi.
- **Yêu cầu:** Sinh viên vận dụng được các kiến thức đã học, lập được kế hoạch phỏng vấn, lập phiếu điều tra, tạo được các báo cáo khả thi cho hệ thống sắp xây dựng.
- **Hình thức tổ chức dạy học**: Lý thuyết, tự học
- **Thời gian**: Lý thuyết(trên lớp: 2; online: 2) Tự học, tự nghiên cứu: 8

1. Hệ thống thông tin

Hệ thống thông tin là hệ thống cung cấp thông tin cho công tác quản lý của các tổ chức. Hệ thống bao gồm con người, thiết bị và quy trình thu thập, phân tích, đánh giá, phân phối kịp thời, chính xác những thông tin cần thiết giúp ban lãnh đạo đưa ra các quyết định đúng đắn cho tổ chức.

Thông tin quản lý là những dữ liệu được xử lý và sẵn sàng phục vụ công tác quản lý của tổ chức. Có 3 loại thông tin quản lý trong một tổ chức, đó là thông tin chiến lược, thông tin chiến thuật, và thông tin điều hành.

Thông tin chiến lược: là thông tin sử dụng cho chính sách dài hạn của tổ chức, chủ yếu phục vụ cho các nhà quản lý cao cấp khi dự đoán tương lai. Loại thông tin này đòi hỏi tính khái quát, tổng hợp cao. Dữ liệu để xử lý ra loại thông tin này thường là từ bên ngoài tổ chức. Đây là loại thông tin được cung cấp trong những trường hợp đặc biệt.

Thông tin chiến thuật: là thông tin sử dụng cho chính sách ngắn hạn, chủ yếu phục vụ cho các nhà quản lý phòng ban trong tổ chức. Loại thông tin này trong khi cần mang tính tổng hợp vẫn đòi hỏi phải có mức độ chi tiết nhất định dạng thống kê. Đây là loại thông tin cần được cung cấp định kỳ.

Thông tin điều hành: (thông tin tác nghiệp) sử dụng cho công tác điều hành tổ chức hàng ngày và chủ yếu phục vụ cho người giám sát hoạt động tác nghiệp của tổ chức.

Loại thông tin này cần chi tiết, được rút ra từ quá trình xử lý các dữ liệu trong tổ chức. Đây là loại thông tin cần được cung cấp thường xuyên.

Mặc dù một hệ thống thông tin không nhất thiết phải sử dụng công nghệ thông tin, nhưng với sự phát triển tối ưu của mình công nghệ thông tin đang ngày càng góp phần tạo ra năng suất xử lý, lưu trữ, phân phối thông tin ngày một cao, hiệu quả và thuận lợi nên các hệ thống thông tin quản lý hiện đại thường sử dụng công nghệ thông tin dưới dạng các phần mềm quản lý.

2. Vòng đời phát triển hệ thống thông tin

Hệ thống thông tin được phát triển thông qua việc phát triển các hệ thống phần mềm, đây không phải đơn giản chỉ là công việc lập trình mà nó luôn được xem như một tiến trình hoàn chỉnh với các thành phần chủ yếu bao gồm: mô hình vòng đời phát triển phần mềm, các công cụ hỗ trợ cho phát triển phần mềm và con người trong nhóm phát triển phần mềm.

2.1. Chu trình phát triển phần mềm

Chu trình phát triển phần mềm là một chuỗi các hoạt động của nhà phân tích, nhà thiết kế, nhà phát triển và người dùng 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: là người nghiên cứu các yêu cầu của khách hàng để định nghĩa phạm vi bài toán, nhận dạng nhu cầu của 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 quản lý của tổ chức này.
+ Nhà thiết kế: thiết kế hệ thống theo các hướng về cơ sở dữ liệu, tổ chức biểu diễn các forms và định dạng các forms báo cáo – quyết định các yêu cầu về phần cứng và phần mềm cho hệ thống.
+ Chuyên gia lĩnh vực: 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: là những người dựa trên các phân tích và thiết kế để viết chương trình cho hệ thống bằng ngôn ngữ lập trình đã được thống nhất.
+ Người dùng: là đối tượng phục vụ của hệ thống cần được phát triển.

Trong quá trình phát triển phần mềm, các nhà phân tích đóng vai trò rất quan trọng nên kết quả phân tích phải được thể hiện sao cho dễ hiểu đối với toàn thể đội ngũ phát triển phần mềm. Đây cũng là môt trong rất nhiều lý do khiến cho phương pháp hướng đối tượng được nhiều người hưởng ứng.

2.2. 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
+ Phân tích yêu cầu

+ Thiết kế hệ thống
+ Xây dựng phần mềm
+ Thử nghiệm hệ thống
+ Thực hiện, triển khai
+ Bảo trì, nâng cấp

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 không 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 hệ thống để thực hiện không*?"

Các hoạt động trong thời gian này thường bao gồm các việc: 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".

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.

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. 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.

Kết quả của giai đoạn phân tích là bản đặc tả yêu cầu.

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.

Công tác thiết kế xoay quanh câu hỏi chính: Hệ thống làm cách nào để thỏa mãn các yêu cầu đã được nêu trong bản 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 các bản báo cáo và những kết quả mà hệ thống mới phải sinh ra.
+ Thiết kế các form nhập liệu (vẽ trên giấy hoặc máy tính).
+ Nhận biết các thành phần dữ liệu và bảng để tạo cơ sở dữ liệu.

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

Kết quả giai đoạn thiết kế là bản đặc tả thiết kế. 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.

Xây dựng phần mềm

Đây là giai đoạn viết mã lệnh (code) thực sự để tạo nên 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 các thủ tục mà anh ta tạo ra được viết như thế nào và để thực hiện công việc gì?

Để đảm bảo chương trình được viết 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.

Thử nghiệm hệ thống

Sau khi các thủ tục đã được thử nghiệm riêng, cần phải thử nghiệm toàn bộ hệ thống.

Trong thử nghiệm hệ thống mọi thủ tục được tích hợp và chạy thử để kiểm tra xem các chi tiết ghi trong bản đặc tả yêu cầu và những mong chờ của người dùng có được thoả mãn không?

Dữ liệu thử cần được chọn lọc đặc biệt, kết quả cần được phân tích để phát hiện mọi lệch lạc so với sự mong đợi.

Thực hiện, triển khai

Trong giai đoạn này, hệ thống vừa phát triển sẽ được triển khai 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 ra các tệp hướng dẫn sử dụng cũng như huấn luyện cho người dùng để đảm bảo hệ thống được sử dụng hữu hiệu nhất.

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 hệ thống được sử dụng có hiệu quả.

Hoạt động bảo trì hệ thống cho các hệ thống khác nhau có thể rất khác biệt tùy theo mức độ cần sửa đổi và nâng cấp của từng hệ thống.

Phát triển phần mềm

Kinh nghiệm của nhiều nhà thiết kế và phát triển cho thấy, phát triển phần mềm là một bài toán phức tạp với một số các lý do như sau:

+ Những nhà phát triển phần mềm thường gặp khó khăn trong việc hiểu cho đúng những gì người dùng cần.
+ Yêu cầu người dùng thường được miêu tả bằng văn bản, dài dòng, khó hiểu thậm chí nhiều khi còn bị mâu thuẫn và thường thay đổi trong thời gian phát triển.
+ 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.

+ Khó định lượng chính xác hiệu suất cũng như sự hài lòng của người dùng đối với sản phẩm.
+ Chọn lựa phần cứng và phần mềm thích hợp cho giải pháp là một trong những thách thức lớn đối với đội quân phát triển phần mềm.

Ngoài ra phần mềm 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 người dùng hay từ phía công nghệ. Ví dụ phần mềm đã được phát triển cho một ngân hàng, cần có khả năng tái sử dụng cho một ngân hàng 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.

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, đã được áp dụng để tạo nên hàng ngàn hệ thống trong suốt nhiều năm qua. Theo lối tiếp cận này, chúng ta quan tâm chủ yếu tới những thông tin mà hệ thống sẽ lưu trữ.

Đội ngũ phát triển tìm hiểu người dùng sẽ cần những thông tin nào, rồi thiết kế ngân hàng dữ liệu để chứa những thông tin đó, cung cấp các form để nhập thông tin và in các báo cáo để trình bày các thông tin.

Nói cách khác, chúng ta tập trung vào thông tin và không để ý đến những gì có thể xảy ra với những hệ thống đó và cách hoạt động của hệ thống ra sao.

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 thì 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 cấu trúc có ưu điểm là tư duy phân tích thiết kế rõ ràng, chương trình sáng sủa dễ hiểu. Tuy nhiên, phương pháp này có một số nhược điểm sau:

+ Không hỗ trợ việc sử dụng lại. Các chương trình hướng cấu trúc phụ thuộc chặt chẽ vào cấu trúc dữ liệu và bài toán cụ thể, do đó không thể dùng lại một modul nào đó trong phần mềm này cho phần mềm mới với các yêu cầu về dữ liệu khác.
+ Không phù hợp cho phát triển các phần mềm lớn. Nếu hệ thống thông tin lớn, thì việc phân ra thành các bài toán con, cũng như phân chia các bài toán con thành các modul và quản lý mối quan hệ giữa các modul đó không phải là dễ dàng.

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 của hệ thống.

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.

Lối tiếp cận hướng đối tượng là lối tư duy về vấn đề theo cách 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 có thể xây dựng ứng dụng bằng cách chắp các đối tượng đó lại với nhau.

3.3. Ưu điểm của mô hình hướng đối tượng

Tính tái sử dụng

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: ta có thể tạo các đối tượng một lần và dùng chúng nhiều lần sau đó.

Ví dụ ta có thể tái sử dụng các đối tượng căn bản trong các thiết kế hướng đối tượng cũng như mã lệnh 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 giảm thiểu các khó khăn trong việc bảo trì, giúp tăng tốc độ thiết kế và phát triển phần mềm, giúp tạo ra các thế hệ phần mềm có khả năng thích ứng và bền chắc.

Tính trực quan

Ta có thể thấy rằng, một tập hợp dữ liệu lớn phức tạp khi được trình bày bằng đồ thị sẽ truyền tải đến người đọc nhiều thông tin hơn so với việc trình bày bằng văn bản.

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. Với tính trực quan, dễ hiểu, phương pháp phân tích thiết kế hướng đối tượng giúp tăng cường khả năng nắm bắt, kiểm soát sự phức tạp đó với một sự trình bày vượt ra ngoài giới hạn của những dòng lệnh thô.

Mô hình hóa trực quan

Phương pháp phân tích thiết kế hướng đối tượng sử dụng mô hình hoá trực quan, đó 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 hoá trực quan giúp chúng ta hiểu rõ vấn đề và giao tiếp được với tất cả các thành viên có liên quan đến dự án như khách hàng, chuyên gia lĩnh vực thuộc đề án, nhà phân tích, nhà thiết kế…

Phù hợp với các hệ thống lớn

Phương pháp hướng đối tập trung vào việc xác định các đối tượng, dữ liệu và hành động gắn với đối tượng đó cũng như mối quan hệ giữa các đối tượng khác nhau.
Các đối tượng độc lập với nhau và chỉ hoạt động khi nhận được yêu cầu từ các đối tượng khác. Vì vậy, phương pháp này hỗ trợ việc phân tích, thiết kế và quản lý cho một hệ thống lớn có các hoạt động nghiệp vụ phức tạp bởi quá trình phân tích thiết kế không phụ thuộc vào số biến dữ liệu hay số lượng các thao tác cần thực hiện mà chỉ quan tâm đến các đối tượng tồn tại trong hệ thống đó.

4. Các giai đoạn phát triển phần mềm với mô hình hướng đối tượng

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

Là giai đoạn phát triển một mô hình chính xác và súc tích một vấn đề nào đó, 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 vẫn 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.
Ví dụ đối với một phòng bán ô tô, giai đoạn OOA sẽ nhận biết được các đối tượng như:

+ Khách hàng.
+ Người bán hàng.
+ Phiếu đặt hàng.
+ Hoá đơn thanh toán.
+ Xe ô tô.

Mối 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.

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.

4.2. 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.
+ Các 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.

4.3. 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.

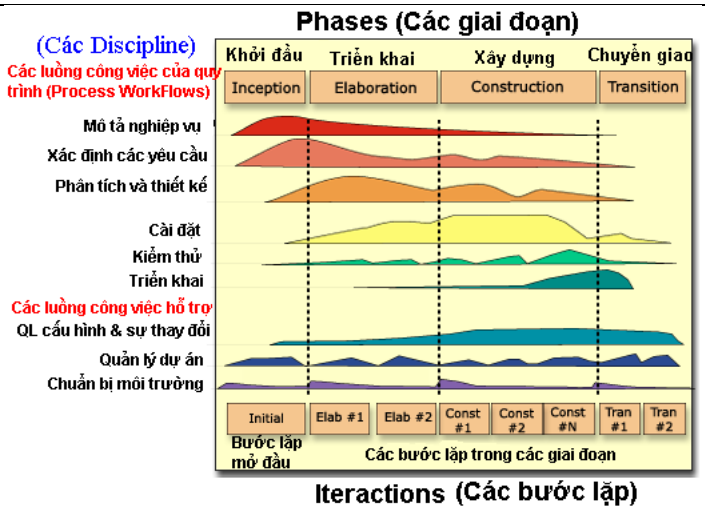
Một vài ngôn ngữ hướng đối tượng thường được nhắc tới là C++ và Java. Kết quả chung cuộc của giai đoạn này là một loạt các code chạy được, nó chỉ được đưa vào sử dụng sau khi đã trải qua nhiều vòng quay của nhiều bước thử nghiệm khác nhau.

4.4. Quy trình phát triển phần mềm hướng đối tượng Rational Unified Process (RUP)

Quy trình RUP (Rational Unified Process): Là quy trình công nghệ phần mềm được phát triển bởi hãng Rational.

Hỗ trợ các hoạt động giữa các nhóm, phân chia công việc cho từng thành viên trong nhóm trong từng giai đoạn khác nhau của quá trình phát triển phần mềm.

Sử dụng hệ thống ký hiệu trực quan của ngôn ngữ UML và được phát triển song song với UML.


Phases (Các giai đoạn)
(Các Discipline)
Các luồng công việc của quy trình (Process WorkFlows)
Khởi đầu
Triển khai
Xây dựng
Chuyển giao
Inception
Elaboration
Construction
Transition
Mô tả nghiệp vụ
Xác định các yêu cầu
Phân tích và thiết kế
Cài đặt
Kiểm thử
Triển khai
Các luồng công việc hỗ trợ
QL cấu hình & sự thay đổi
Quản lý dự án
Chuẩn bị môi trường
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #N
Tran #1
Tran #2
Bước lặp mở đầu
Các bước lặp trong các giai đoạn
Iteractions (Các bước lặp)

5. Ngôn ngữ mô hình hoá thống nhất - UML

5.1. Giới thiệu UML

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, xuất hiện 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 do đó 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.

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 (Unifield Modeling Language)

UML là một ngôn ngữ mô hình hoá thống nhất, bao gồm những ký hiệu hình học, được sử dụng để thể hiện và mô tả các thiết kế của một hệ thống. Nó là một ngôn ngữ để đặc tả, trực quan hoá, xây dựng và làm tài liệu cho nhiều khía cạnh khác nhau của một hệ thống phần mềm.

UML cũng 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.

Đặc điểm của ngôn ngữ UML

UML là ngôn ngữ mô hình hoá nên nó thoả mãn các tính chất sau:

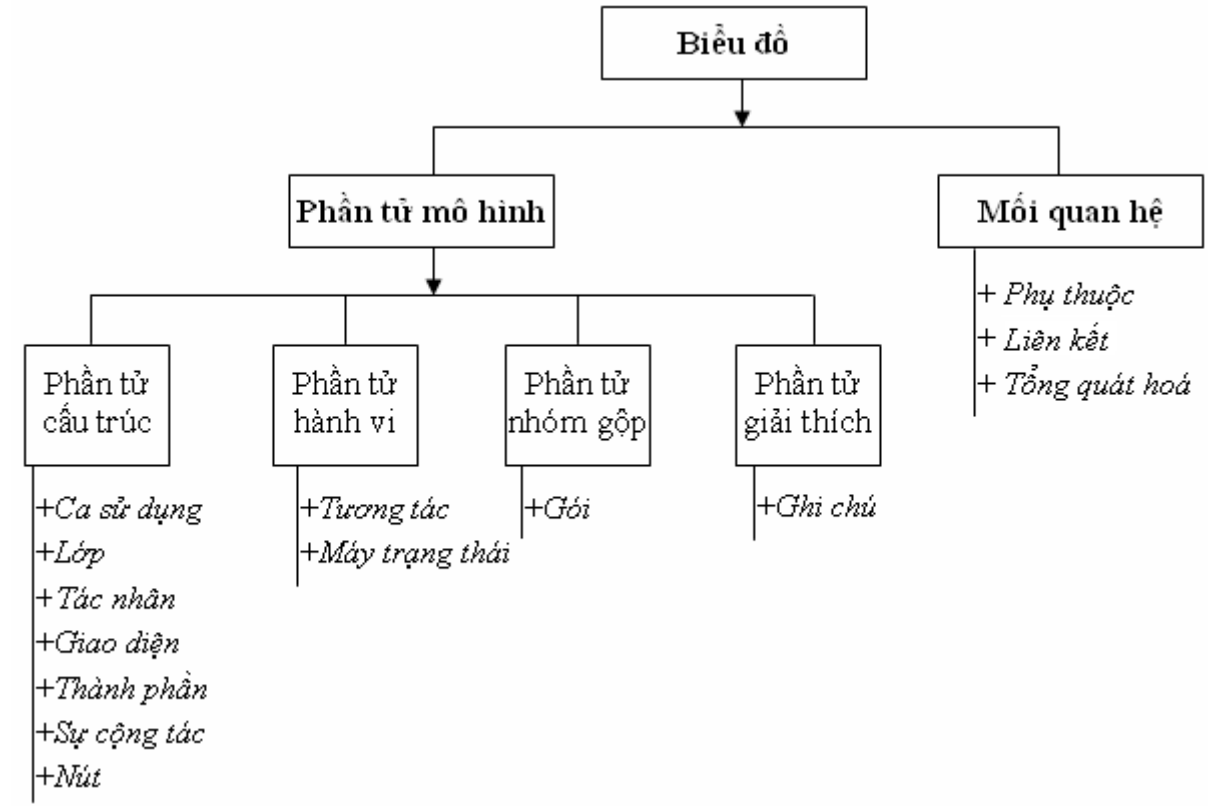
+ Chính xác: mô tả đúng hệ thống cần xây dựng.
+ Đồng nhất: các khung nhìn khác nhau không mâu thuẫn với nhau.
+ Có thể hiểu được: cho cả những người xây dựng lẫn những người sử dụng
+ Dễ chỉnh sửa và dễ dàng liên kết với các mô hình khác.

5.2. Các thành phần của ngôn ngữ UML

Một số những thành phần chủ yếu của ngôn ngữ UML:

+ ***Biểu đồ (diagram):*** là các hình vẽ miêu tả nội dung trong một khung nhìn của hệ thống. UML có tất cả 9 loại biểu đồ khác nhau, được sử dụng trong những sự kết hợp khác nhau để cung cấp tất cả các khung nhìn của một hệ thống.
+ ***Phần tử mô hình hóa (model element):*** các khái niệm được sử dụng trong các biểu đồ được gọi là các phần tử mô hình. Mỗi phần tử mô hình có một sự miêu tả trực quan, một ký hiệu hình học với một ý nghĩa xác định.
  Ví dụ như lớp, đối tượng, thông điệp cũng như các mối quan hệ giữa các khái niệm này. Một phần tử mô hình có thể được sử dụng trong nhiều biểu đồ khác nhau, nhưng nó luôn có duy nhất một ý nghĩa và một ký hiệu hình học.
+ ***Cơ chế chung:*** cơ chế chung cung cấp thêm những lời nhận xét, bổ sung thêm các thông tin cũng như các quy tắc ngữ pháp về một phần tử mô hình; chúng còn cung cấp thêm các cơ chế để có thể mở rộng ngôn ngữ UML cho phù hợp với một phương pháp xác định.

Các thành phần của ngôn ngữ UML được mô tả trực quan bởi hình vẽ sau:

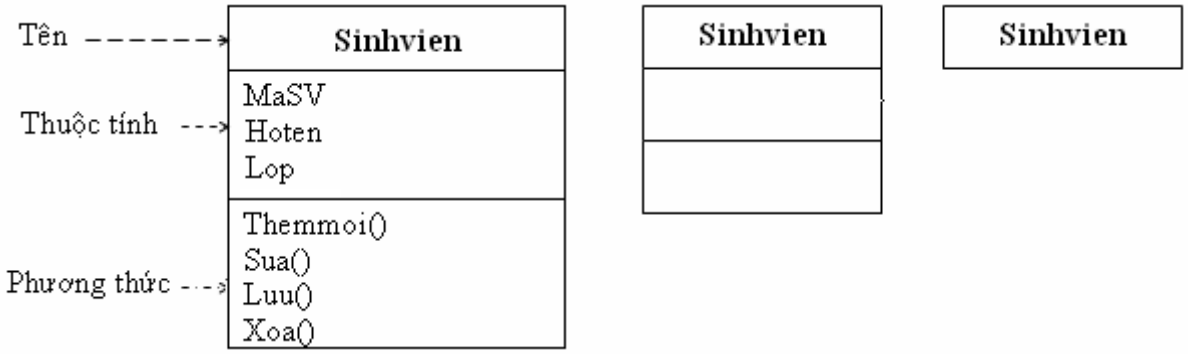


5.3. Phần tử cấu trúc

Lớp

Một lớp mô tả một tập hợp các đối tượng có chung các thuộc tính, các phương thức, các mối quan hệ và ngữ nghĩa.

Lớp được biểu diễn bằng một hình chữ nhật, bên trong có tên, các thuộc tính và các phương thức. Ngoài ra khi muốn vẽ đơn giản thì có thể bỏ trống hoặc bỏ qua các thuộc tính hoặc các phương thức.



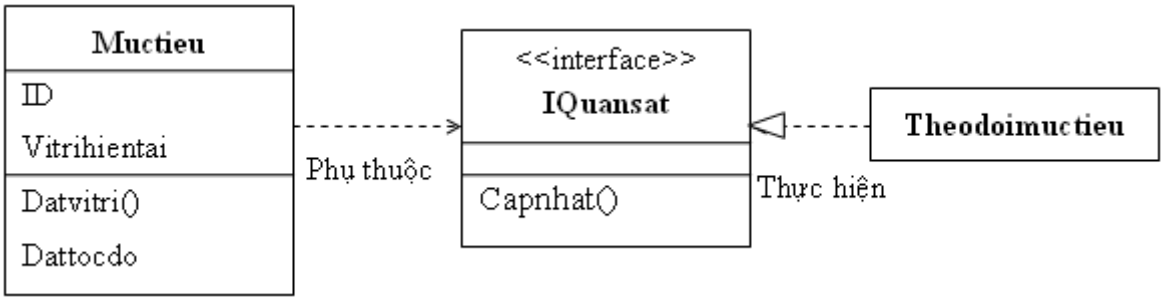
Chú ý: Một lớp có trách nhiệm thực hiện một hay nhiều giao diện.

Giao diện

Giao diện là một tập hợp các tác vụ, đặc tả một dịch vụ của một lớp hoặc của một thành phần. Giao diện không có cấu trúc nghĩa là không có thuộc tính, và các thao tác của nó không được cài đặt nghĩa là không có phương thức. Các cài đặt này thuộc trách nhiệm của lớp hay thành phần cung cấp giao diện đó.

Một lớp hay một thành phần có thể cung cấp nhiều giao diện, mỗi giao diện đó thể hiện một vai trò, ví dụ một NHANVIEN có thể có nhiều hành vi giao tiếp khác nhau, tuỳ theo vai trò như là giám đốc, kế toán... Mặt khác một giao diện lại có thể được thực hiện bởi nhiều lớp hay thành phần khác nhau.

Giao diện được biểu diễn bằng một hình chữ nhật chia 3 ngăn như lớp, cho phép diễn tả nội dung bên trong của nó. Tên giao diện thường có thêm tiền tố I ở trước. Giao diện được nối với lớp hay thành phần cung cấp nó bằng một đường mũi tên nét đứt có đầu tam giác (mối liên quan thực hiện), và được nối với lớp hay thành phần yêu cầu nó bằng một mũi tên nét đứt đầu mở (mối liên quan phụ thuộc).

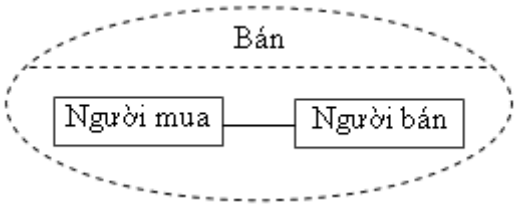


Sự cộng tác

Sự cộng tác xác định các hoạt động bên trong hệ thống, nó diễn tả một tập hợp các phần tử tương tác cùng nhau thông qua các đường liên lạc, để hoàn thành một nhiệm vụ hay một tập hợp các nhiệm vụ chung nào đó.

Đây chỉ là một sự diễn tả về cấu trúc, nên nó chưa đi sâu vào các chi tiết của sự tương tác (như biểu đồ tương tác). Mặt khác các chi tiết về cấu trúc, như là tên và chi tiết đầy đủ về lớp của các cá thể tham gia hợp tác cũng có thể được bỏ qua, nghĩa là chỉ giữ lại các sắc thái để giải thích cho một sự hợp tác cụ thể.

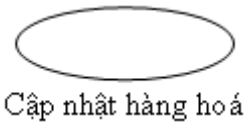
Một sự cộng tác được biểu diễn bằng một hình ellip đứt, trong đó có viết tên sự cộng tác. Ngoài ra có thể thêm một ngăn trong hình ellip để vẽ cấu trúc nội tại của nó bao gồm các vai trò và các cầu nối giữa chúng.



Ca sử dụng

Ca sử dụng mô tả một tập hợp các hành động mà hệ thống thực hiện để cho ra một kết quả có thể quan sát được đối với một tác nhân.

Ca sử dụng được mô tả bằng một đường ellip nét liền, phía dưới có tên ca sử dụng.



Actor (tác nhân)

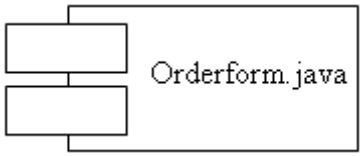
Actor là đại diện cho người hoặc một hệ thống khác, tương tác với hệ thống đang được xét thông qua các ca sử dụng. Actor được chia làm 2 loại: actor kích hoạt hệ thống và actor tham gia sử dụng hệ thống.
Actor được biểu diễn bằng một biểu tượng hình người, phía dưới có tên actor.

Thành phần

Thành phần là một bộ phận vật lý có thể thay thế được của một hệ thống, phù hợp với những điều kiện cụ thể và cung cấp các phương thức thực hiện một tập các giao diện.
Ta có 3 loại thành phần:

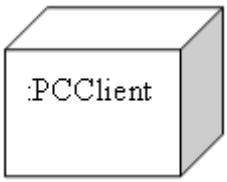
+ *Thành phần bố trí*: đó là các thành phần cần và đủ để tạo nên một hệ thống khả thi, như là các thư viện động (DLL) và các mã khả thi.
+ *Thành phần sản phẩm làm việc*: đó là các thành phần tồn tích từ quá trình phát triển hệ thống, như các tệp mã nguồn, các tệp dữ liệu... Các thành phần này không trực tiếp tham gia vào hệ thống thực thi nhưng không có chúng thì không tạo được hệ thống thực thi.
+ *Thành phần thực hiện*: đó là các thành phần được tạo nên như là một kết quả của một hệ thực hiện, chẳng hạn một đối tượng COM...

Thành phần được mô tả bằng một hình chữ nhật với hai hình chữ nhật con trên biên trái, bên trong chứa tên của thành phần.



Nút

Nút là một tài nguyên có thể thực thi được chương trình, nút thường có một bộ nhớ và một bộ xử lý. Các đối tượng và các thành phần lúc chạy có thể trú ngụ trên các nút. Ví dụ nút có thể là các máy khách và máy chủ.
Nút được biểu diễn bằng một hình hộp chữ nhật bên trong chứa tên của nó.



5.4. Phần tử hành vi

Tương tác – Thông điệp

Tương tác là một hành vi bao gồm một tập các thông báo được trao đổi giữa một tập các đối tượng trong một khung cảnh cụ thể, nhằm thực hiện một mục tiêu xác định.

Khi một đối tượng gửi một thông điệp cho một đối tượng khác, thì đối tượng này trong khi thực hiện các hành động đáp ứng thông điệp trên lại có thể gửi thông điệp cho đối tượng khác, cứ như thế tạo thành một luồng kích hoạt lan dần.

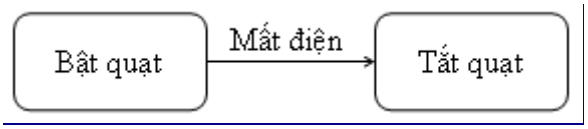
+ Thông điệp được sử dụng trong biểu đồ tuần tự.

Máy trạng thái

Máy trạng thái gồm một số các phần tử biểu diễn các trạng thái và các chuyển dịch từ một trạng thái này sang một trạng thái khác.

Các trạng thái được dùng trong biểu đồ trạng thái.

Một trạng thái được ký hiệu bằng một hình chữ nhật góc tròn, trong đó có tên trạng thái và các trạng thái con của nó (nếu có).

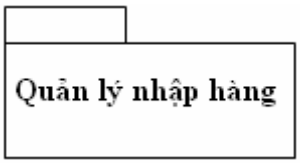


Phần tử nhóm gộp - Gói

Gói là một cơ chế để gom nhóm nhiều phần tử vào với nhau. Các phần tử được đóng gói ở đây có thể là các lớp, các giao diện, các thành phần, các ca sử dụng, các nút... và cũng có thể là các gói khác.

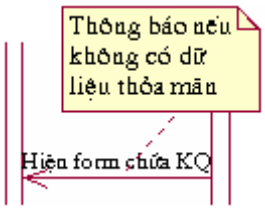
Gói có thể được sử dụng ở mọi giai đoạn phát triển của hệ thống. Khi các phần tử quá lớn hoặc quá phức tạp và ta muốn diễn tả lại nó sơ lược hơn, nhưng dễ nắm bắt hơn, dễ phân công nghiên cứu và dễ triển khai hơn thì ta dùng gói.

Gói được biểu diễn bằng một hình chữ nhật có quai, bên trong ghi tên gói. Các thành phần của gói có thể được liệt kê trong hình chữ nhật hoặc bị bỏ qua.



Phần tử giải thích

Chú thích được sử dụng trong mô hình UML để mô tả, giải thích và đánh dấu một phần tử bất kỳ trong một mô hình. Phần tử giải thích thể hiện ra như một ghi chú và được ký hiệu bằng một hình chữ nhật có góc gấp cùng với một lời bình luận bằng văn bản hay đồ thị ở bên trong .



5.5.Mối quan hệ

Mối quan hệ phụ thuộc

Thường dùng để diễn đạt một đối tượng (bên phụ thuộc) chịu ảnh hưởng của mọi thay đổi trong một đối tượng khác (bên độc lập.

Mối quan hệ phụ thuộc được biểu diễn bằng một mũi tên nét đứt có chiều từ bên phụ thuộc sang bên độc lập.

Có nhiều biểu hiện khác nhau của sự phụ thuộc, ta thường dùng từ khoá để chỉ rõ sự khác biệt đó, ví dụ: **<<use>>, <<permit>>**, **<<instance of>>**, **<<include>>**, **<<extend>>…**

Mối quan hệ liên kết

Giữa các cá thể của hai lớp có thể tồn tại những sự ghép cặp, phản ánh một mối liên hệ nào đó trên thực tế. Quan hệ liên kết mô tả các mối liên hệ đó.

Quan hệ liên kết được ký hiệu bằng đường nét liền nối hai lớp, có thể chứa tên của liên kết và tại mỗi đầu của liên kết có thể có thêm một cơ số cho biết số cá thể tối thiểu và tối đa của đầu đó tham gia liên kết với một cá thể ở đầu bên kia.

Mối quan hệ tổng quát hoá

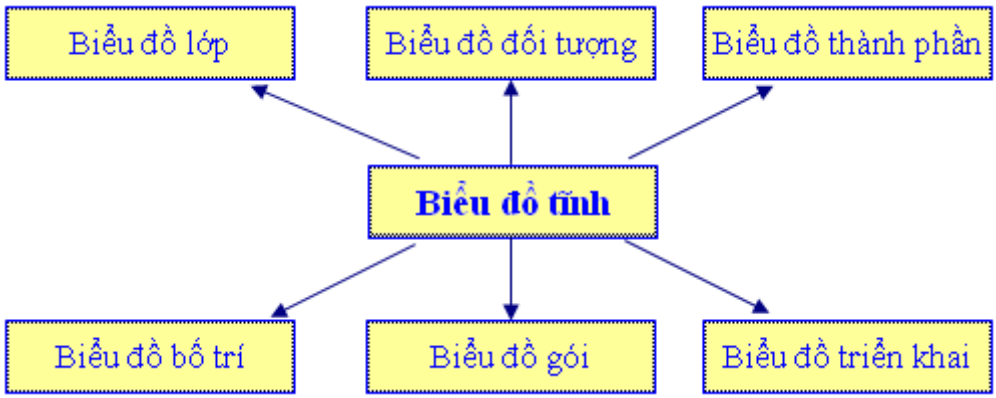
Tổng quát hoá là sự rút ra các đặc điểm chung của nhiều lớp con để tạo thành một lớp giản lược hơn gọi là lớp cha. Mối quan hệ tổng quát hoá được ký hiệu bởi một đường nét liền với một mũi tên hình tam giác rỗng chỉ về phía cha.

5.6. Biểu đồ trong UML

Biểu đồ là các hình vẽ mô tả trực quan mối quan hệ giữa các phần tử mô hình hóa để minh họa một thành phần cụ thể hay một khía cạnh cụ thể của hệ thống.

Một mô hình hệ thống thường có nhiều loại biểu đồ, các biểu đồ được chia thành hai loại là biểu đồ tĩnh và biểu đồ động.

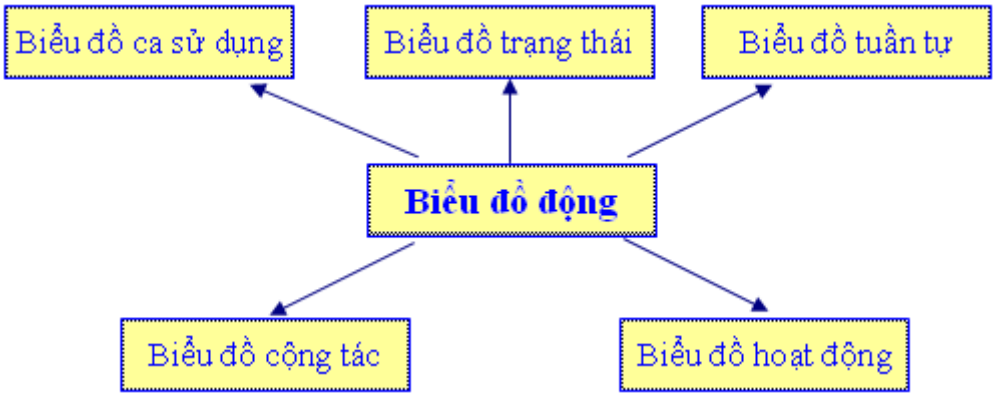
Biểu đồ tĩnh: Mô tả cấu trúc nội tại của một đối tượng và các mối quan hệ của chúng.



Biểu đồ động: Biểu đồ động mô tả sự thay đổi của các đối tượng, xảy ra qua nhiều giai đoạn khác nhau trong thời gian hệ thống hoạt động.

Biểu đồ động đặc biệt quan trọng trong trường hợp giá trị của các đối tượng được thay đổi theo thời gian chạy. Ví dụ, một bảng điểm của một sinh viên sẽ được bổ sung và hoàn chỉnh tăng dần theo thời gian.

Các mô hình động cũng là yếu tố hết sức cần thiết để miêu tả ứng xử của một đối tượng khi đưa ra các yêu cầu hoặc thực thi các tác vụ.



Biểu đồ ca sử dụng: biểu diễn sơ đồ chức năng của hệ thống. Từ tập yêu cầu của hệ thống, biểu đồ use case sẽ phải chỉ ra hệ thống cần thực hiện điều gì để thoả mãn các yêu cầu của người dùng hệ thống đó.
Biểu đồ lớp: chỉ ra các lớp đối tượng trong hệ thống, các thuộc tính và phương thức của từng lớp và các mối quan hệ giữa những lớp đó.
Biểu đồ trạng thái: tương ứng với mỗi lớp ta sẽ chỉ ra các trạng thái mà đối tượng của lớp đó có thể có và sự chuyển tiếp giữa những trạng thái đó.
Các biểu đồ tương tác: biểu diễn mối liên hệ giữa các đối tượng trong hệ thống và mối liên hệ giữa các đối tượng với các tác nhân bên ngoài. Có hai loại biểu đồ tương tác:

+ Biểu đồ tuần tự: biểu diễn mối quan hệ giữa các đối tượng với nhau và mối quan hệ giữa các đối tượng với tác nhân theo thứ tự thời gian.
+ Biểu đồ cộng tác: cũng biểu diễn mối quan hệ giữa các đối tượng với nhau và mối quan hệ giữa các đối tượng và tác nhân nhưng nhấn mạnh đến vai trò của các đối tượng trong tương tác.

Biểu đồ hoạt động: biểu diễn các hoạt động của hệ thống.
Biểu đồ thành phần: định nghĩa các thành phần của hệ thống và mối liên hệ giữa các thành phần đó.
Biểu đồ triển khai: mô tả hệ thống sẽ được triển khai như thế nào, thành phần nào được cài đặt ở đâu, các liên kết vật lý hoặc giao thức truyền thông nào được sử dụng.

5.7.Khung nhìn trong UML

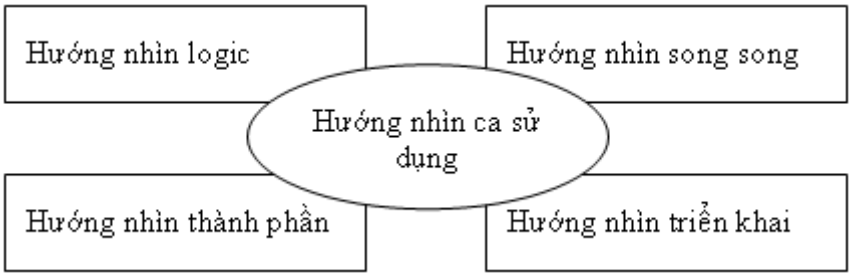
Mô hình hóa một hệ thống phức tạp là một việc làm khó khăn. Lý tưởng nhất là toàn bộ hệ thống được miêu tả chỉ trong một bản vẽ, với các định nghĩa rõ ràng và mạch lạc toàn bộ hệ thống.
Mặc dù vậy, thường thì đây là việc bất khả thi. Một bản vẽ không thể nắm bắt tất cả các thông tin cần thiết để miêu tả một hệ thống, mà một hệ thống cần phải được miêu tả với một loạt các khía cạnh khác nhau:

+ Về mặt chức năng: cấu trúc tĩnh của hệ thống cũng như các tương tác động
+ Về mặt phi chức năng: yêu cầu về thời gian, về độ đáng tin cậy, về quá trình thực thi...
+ Về khía cạnh tổ chức: tổ chức ánh xạ hệ thống vào các đoạn mã lệnh…

Vì vậy một hệ thống thường được miêu tả trong một loạt các khung nhìn khác nhau, mỗi khung nhìn bao gồm một số các biểu đồ khác nhau và thể hiện một khía cạnh riêng của hệ thống.

Một biểu đồ trong một khung nhìn cụ thể nào đó cần phải đủ độ đơn giản để tạo điều kiện giao tiếp dễ dàng với các biểu đồ khác cũng như các khung nhìn khác, làm sao cho bức tranh toàn cảnh của hệ thống được miêu tả bằng sự kết hợp tất cả các thông tin từ tất cả các khung nhìn.

UML có tất cả 5 khung nhìn sau:



Khung nhìn ca sử dụng - Use case View

Khung nhìn ca sử dụng là khung nhìn từ bên ngoài vào hệ thống của các nhà sử dụng, nhà thiết kế, nhà phát triển và người thử nghiệm nhằm làm rõ các chức năng mà hệ thống sẽ phải đáp ứng cho người dùng.

Với UML sắc thái tĩnh của khung nhìn ca sử dụng được diễn tả thông qua biểu đồ ca sử dụng, còn sắc thái động được diễn tả thông qua biểu đồ tương tác, biểu đồ trạng thái, biểu đồ hoạt động.

Khung nhìn ca sử dụng mang tính trung tâm, bởi nó thúc đẩy sự phát triển các khung nhìn khác. Mục tiêu chung của hệ thống là cung cấp các chức năng miêu tả trong khung nhìn này vì thế nó có ảnh hưởng đến tất cả các khung nhìn khác.

Khung nhìn logic - Logical View

Khung nhìn logic là khung nhìn vào bên trong hệ thống, cho thấy các nhiệm vụ của hệ thống được thiết kế ra sao (thành các lớp, các giao diện, các hợp tác như thế nào). Đây là khung nhìn của những nhà thiết kế hệ thống.

Với UML thì sắc thái tĩnh của khung nhìn logic được diễn tả thông qua biểu đồ lớp. Còn sắc thái động được diễn tả thông qua biểu đồ tương tác, biểu đồ trạng thái, biểu đồ hoạt động.

Khung nhìn thành phần - Component View

Là khung nhìn đối với việc cài đặt hệ thống vật lý của phần mềm, nó bao gồm các thành phần và các tệp tương đối độc lập, có thể được lắp ráp lại với nhau theo nhiều cách để tạo ra hệ thống chạy được.

Với UML thì sắc thái tĩnh của khung nhìn thành phần được diễn tả thông qua biểu đồ thành phần. Còn sắc thái động được diễn tả thông qua biểu đồ tương tác, biểu đồ máy trạng thái, biểu đồ hoạt động...

Trong khung nhìn thành phần cũng có thể bổ sung các thông tin về các thành phần như: vị trí của tài nguyên, các bản báo cáo về tiến trình của công việc …

Khung nhìn song song - Concurrency View

Khung nhìn song song nhắm tới sự phân chia hệ thống thành các qui trình (process) và các bộ xử lý (processor). Khía cạnh này, vốn là một thuộc tính phi chức năng của hệ thống, cho phép chúng ta sử dụng một cách hữu hiệu các nguồn tài nguyên, đồng thời cho ta thấy được sự hoạt động song song hay đồng bộ của hệ thống.
Bên cạnh việc chia hệ thống thành các chương trình con có thể được thực thi song song, khung nhìn này cũng phải quan tâm đến vấn đề giao tiếp và đồng bộ hóa các chương trình con đó.
Khung nhìn song song giành cho nhà phát triển và người tích hợp hệ thống, nó bao gồm các biểu đồ động (trạng thái, trình tự, tương tác và hoạt động) cùng các biểu đồ thực thi (biểu đồ thành phần và biểu đồ triển khai).

Khung nhìn triển khai - Deployment View

Khung nhìn triển khai chỉ cho chúng ta sơ đồ triển khai về mặt vật lý của hệ thống, ví dụ sơ đồ liên kết giữa các máy tính cũng như các máy móc khác.
Khung nhìn triển khai giành cho các nhà phát triển, nhà tích hợp cũng như người thử nghiệm hệ thống và được thể hiện bằng các biểu đồ triển khai.
Khung nhìn này cũng bao gồm sự ánh xạ các thành phần của hệ thống vào cấu trúc vật lý, ví dụ như chương trình nào hay đối tượng nào sẽ được thực thi trên máy tính nào...

5.8. UML và các giai đoạn của chu trình phát triển phần mềm

Giai đoạn nghiên cứu sơ bộ

UML đưa ra khái niệm ca sử dụng để nắm bắt các yêu cầu của khách hàng và sử dụng biểu đồ ca sử dụng để nêu bật mối quan hệ cũng như sự giao tiếp với hệ thống.
Giai đoạn này là một bước quan trọng và sự thành công của nó sẽ quyết định sự thành công của dự án, bởi vì một hệ thống dù có được xây dựng tốt nhưng không đáp ứng được những nhu cầu của khách hàng thì hệ thống cũng sẽ thất bại.

Giai đoạn phân tích

Sau khi đã biết được người dùng muốn gì, chúng ta tập trung mô tả lại hệ thống thông qua các lớp, mối quan hệ và sự tương tác giữa chúng trong biểu đồ lớp. Mục đích chính là hiểu hệ thống hoạt động như thế nào.
Trong giai đoạn phân tích, chỉ các lớp mô tả các khái niệm đời thực mới được mô hình hóa, còn các lớp kỹ thuật định nghĩa chi tiết các giải pháp trong hệ thống phần mềm như các lớp cho giao diện người dùng, cho ngân hàng dữ liệu, cho sự giao tiếp... chưa phải là mối quan tâm của giai đoạn này.

Giai đoạn thiết kế

Trong giai đoạn này, kết quả của giai đoạn phân tích sẽ được mở rộng thành một giải pháp kỹ thuật. Các lớp mới thuộc về kỹ thuật sẽ được bổ sung để tạo thành một hạ tầng cơ sở kỹ thuật như: các lớp giao diện (giao diện người dùng, giao diện với các hệ thống khác, giao diện với các thiết bị ngoại vi...), các lớp điều khiển lưu trữ dữ liệu...
Giai đoạn thiết kế tập trung mô tả cấu trúc bên trong của hệ thống, sự tương tác của tập hợp các đối tượng để đạt được những chức năng mà hệ thống cần có.

Giai đoạn lập trình

Trong giai đoạn lập trình, các lớp của giai đoạn thiết kế sẽ được biến thành những dòng mã lệnh cụ thể trong một ngôn ngữ lập trình hướng đối tượng cụ thể.

Khi tạo ra các mô hình phân tích và thiết kế trong UML, tốt nhất nên cố gắng né tránh việc ngay lập tức biến đổi các mô hình này thành các dòng code vì nó có thể sẽ thành một trở ngại cho việc tạo ra các mô hình chính xác và đơn giản.

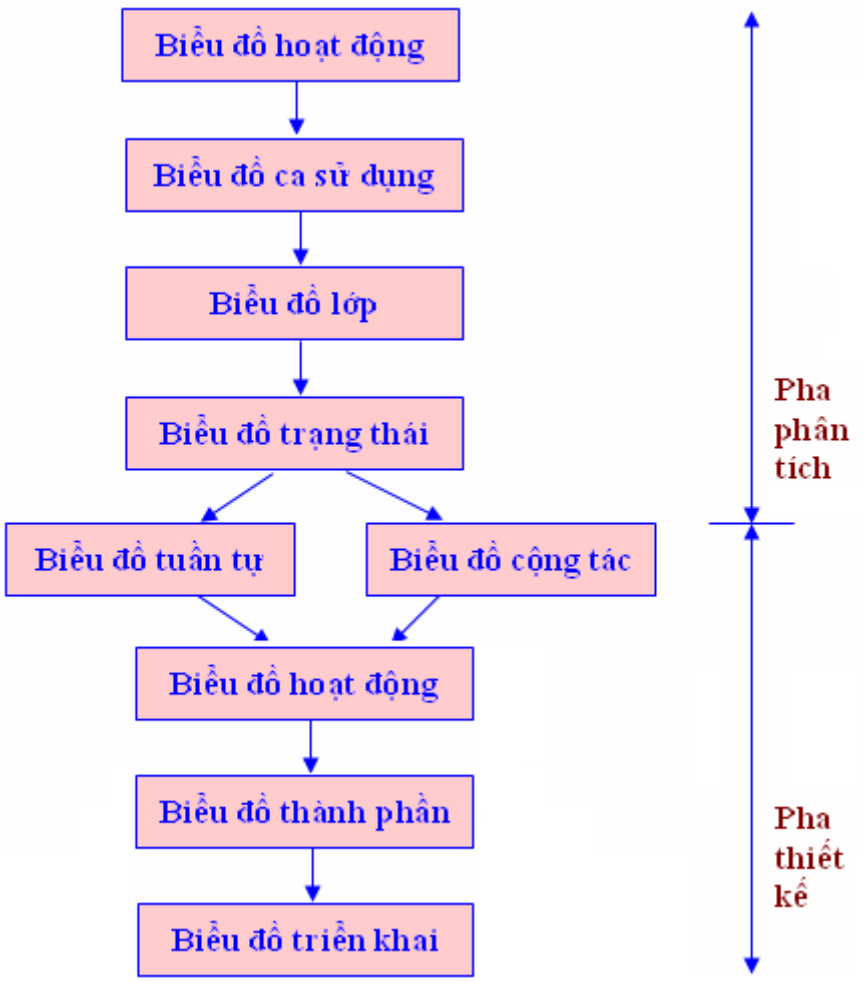
Giai đoạn thử nghiệm

Như đã trình bày trong phần chu trình phát triển phần mềm, một hệ thống phần mềm thường được thử nghiệm qua nhiều giai đoạn và với nhiều nhóm thử nghiệm khác nhau thông qua nhiều loại biểu đồ UML khác nhau làm nền tảng cho công việc của mình:

+ Thử nghiệm đơn vị: sử dụng biểu đồ lớp và đặc tả lớp.
+ Thử nghiệm tích hợp: sử dụng biểu đồ thành phần và biểu đồ cộng tác.
+ Thử nghiệm hệ thống: sử dụng biểu đồ ca sử dụng để đảm bảo hệ thống có phương thức hoạt động đúng như đã được định nghĩa từ ban đầu trong các biểu đồ này.

Các bước phát triển trong trong giai đoạn PTTK hệ thống hướng đối tượng

Tổng quát hoá ta có các bước trong giai đoạn phát triển một hệ thống hướng đối tượng như sau:



6. Công cụ CASE

CASE (Computer Aided Software Engineering) là Công cụ máy tính hỗ trợ kỹ thuật phần mềm.

Các công cụ CASE là các phần mềm hỗ trợ việc phân tích và thiết kế hệ thống.

Có bốn lý do để sử dụng các công cụ CASE là:

- Để tăng hiệu suất phân tích
- Để thuận tiện trong giao tiếp giữa những người phân tích và những người dùng
- Cung cấp sự liên tục giữa các giai đoạn trong vòng đời phát triển hệ thống.
- Để đánh giá tác động của sự bảo trì hệ thống.

CASE tools có thể được chia thành một số loại

- Công cụ Upper CASE dùng để hỗ trợ hoạt động phân tích và thiết kế.
- Công cụ Lower CASE tổng hợp mã nguồn ngôn ngữ máy tính từ các thiết kế CASE.
- CASE tích hợp, thực hiện các chức năng của cả upper và lower CASE.

Ví dụ CASE tool

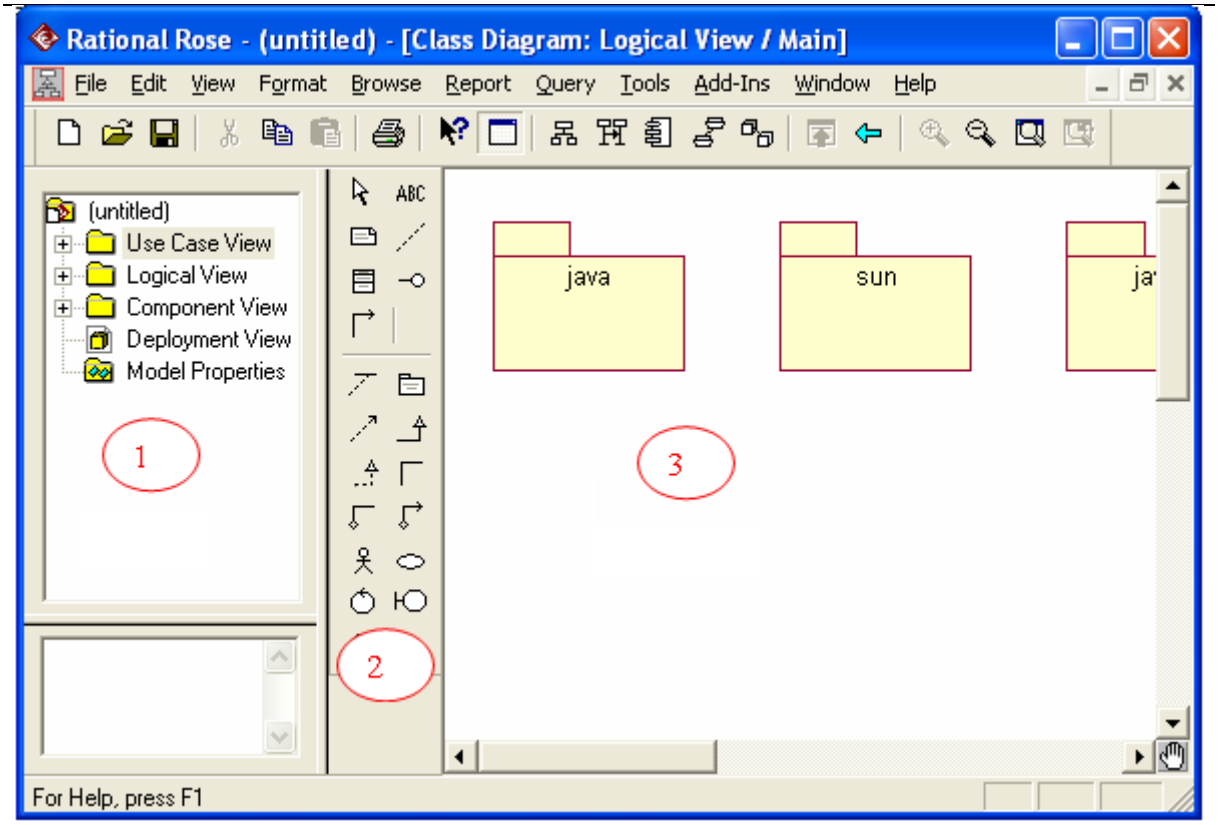
- Phần mềm Rational Rose
- Phần mềm Case Studio 2

6.1. Giới thiệu công cụ trợ giúp Rational Rose

Rational Rose là một bộ công cụ được sử dụng cho phát triển các hệ phần mềm hướng đối tượng theo ngôn ngữ mô hình hóa UML. Với chức năng của một bộ công cụ trực quan, Rational Rose cho phép chúng ta tạo, quan sát, sửa đổi và quản lý các biểu đồ. Tập ký hiệu mà Rational Rose cung cấp thống nhất với các ký hiệu trong các biểu đồ của UML. Ngoài ra, Rational Rose còn cung cấp chức năng hỗ trợ quản lý dự án phát triển phần mềm, cung cấp các thư viện để hỗ trợ sinh khung mã cho hệ thống theo một ngôn ngữ lập trình nào đó.

Khởi động Rational Rose

Khi khởi động của Rational Rose, xuất hiện hộp hội thoại Create New Model, chọn gói **jdk-116** và chọn **OK** ta có cửa sổ sau:



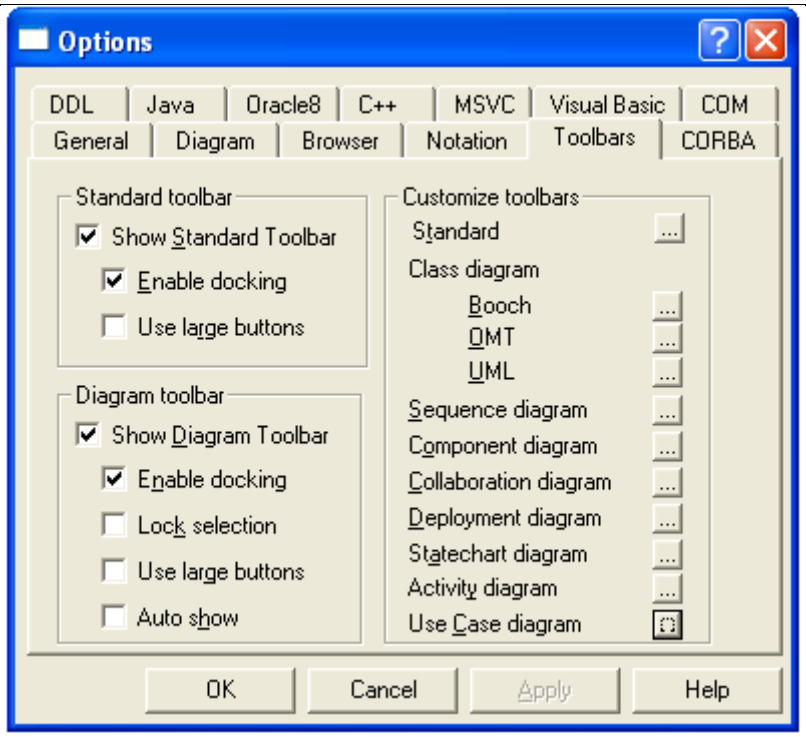
Cửa sổ này gồm 3 phần:

+ Phần 1: Browser Windows cho phép người sử dụng chuyển tiếp nhanh giữa các biểu đồ trong các View khác nhau.
+ Phần 2: ToolBox chứa các công cụ dùng để vẽ các biểu đồ, ứng với mỗi dạng biểu đồ thì sẽ có một dạng Toolbox tương ứng.
+ Phần 3: Diagram Windows là không gian để vẽ và hiệu chỉnh các biểu đồ.

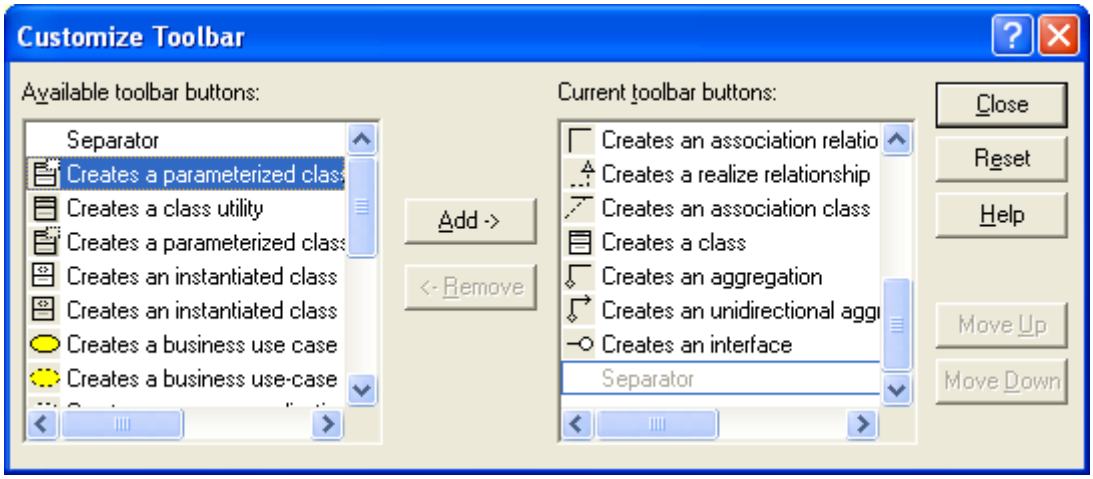
Cài đặt các biểu tượng trên thanh công cụ ToolBox

Ứng với mỗi dạng biểu đồ sẽ có một thanh công cụ Toolbox tương ứng, và mỗi một ToolBox chứa những ký hiệu khác nhau.

Ban đầu, mỗi thanh Toolbox chỉ chứa những ký hiệu cơ bản nhất, trong quá trình vẽ các biểu đồ nếu thiếu một ký hiệu nào đó thì ta có thể lấy chúng ra bằng cách: chọn menu Tools/ Options/ chọn tab Toolbars, ta có cửa sổ sau:



Cần cài đặt thêm các ký hiệu cho thanh công cụ ứng với biểu đồ nào thì ta chọn nút … ứng với biểu đồ đó, ví dụ kích chuột vào nút … cạnh biểu đồ - Use Case diagram ta có:



Cần ký hiệu nào thì ta chọn ký hiệu đó, rồi bấm nút Add, kết quả ký hiệu đó sẽ xuất hiện trên thanh công cụ.

Để vẽ các biểu đồ bằng công cụ Rational Rose, ta có thể chọn phần tạo mới biểu đồ trong các View khác nhau trên Browser Windows

6.2. Công cụ trợ giúp Case Studio 2

Case Studio là công cụ giúp ta có thể vẽ biểu đồ thực thể liên kết ERD, giúp ta có thể thiết kế cơ sở dữ liệu một cách dễ dàng.

Yêu cầu sinh viên chuẩn bị:

Đọc trước đề cương bài giảng chi tiết và slides bài giảng, xem video bài giảng, làm bài trắc nghiệm bài 1.