**Cơ bản về phân tích và thiết kế hướng đối tượg với UML**

**Trong những năm gần đây phương thức lập trình hướng đối tượng đã thống lĩnh thị trường lập trình phần mềm và UML cũng đã trở thành ngôn ngữ mô hình hóa phổ biến trong sản xuất phần mềm.  Hầu hết các trường đại học, cao đẳng đã đưa hai môn này vào đào tạo chính khóa và cũng có không ít tài liệu viết về những vấn đề này. Tuy nhiên, nó vẫn còn rất khó hiểu và khó áp dụng với sinh viên, và những bạn trẻ đang làm về Công nghệ thông tin.**

Trong loạt bài này, chúng tôi sẽ cố gắng trình bày một cách đơn giản và dễ hiểu nhất các kiến thức về Phân tích và thiết kế hướng đối tượng và UML để giúp các bạn hiểu và áp dụng vào thực tế. Các bài viết sẽ hướng dẫn các bạn phân tích và thiết kế một ứng dụng cụ thể để từ đó tự rút ra bài học kinh nghiệm cho mình và tiếp tục nghiên cứu sâu hơn.

Bài đầu tiên sẽ bàn một cách cơ bản về Phân tích thiết kế hướng đối tượng và UML.

***Lưu ý:*** Để hiểu được loạt bài này bạn phải có kiến thức cơ bản về lập trình hướng đối tượng, vì chúng tôi sẽ không đi chi tiết vào các định nghĩa về lập trình hướng đối tượng.

1. **Khái niệm về Phân tích và thiết kế hướng đối tượng (Object Oriented Analysis and Design: OOAD)**

Trong kỹ nghệ phần mềm để sản xuất được một sản phẩm phần mềm người ta chia quá trình phát triển sản phẩm ra nhiều giai đoạn như thu thập và phân tích yêu cầu, phân tích và thiết kế hệ thống, phát triển (coding), kiểm thử, triển khai và bảo trì. Trong đó, giai đoạn phân tích, thiết kế bao giờ cũng là giai đoạn khó khăn và phức tạp nhất. Giai đoạn này giúp chúng ta hiểu rõ yêu cầu đặt ra, xác định giải pháp, mô tả chi tiết giải pháp. Nó trả lời 2 câu hỏi What (phần mềm này làm cái gì?) và How (làm nó như thế nào?).

Để phân tích và thiết kế một phần mềm thì có nhiều cách làm, một trong những cách làm đó là xem hệ thống gồm những đối tượng sống trong đó và tương tác với nhau. Việc mô tả được tất cả các đối tượng và sự tương tác của chúng sẽ giúp chúng ta hiểu rõ hệ thống và cài đặt được nó. Phương thức này gọi là Phân tích thiết kế hướng đối tượng (OOAD)

1. **Khái niệm về UML (Unified Modeling Language)**

UML là ngôn ngữ mô hình hóa hợp nhất dùng để biểu diễn hệ thống. Nói một cách đơn giản là nó dùng để tạo ra các bản vẽ nhằm mô tả thiết kế hệ thống. Các bản vẽ này được sử dụng để các nhóm thiết kế trao đổi với nhau cũng như dùng để thi công hệ thống (phát triển), thuyết phục khách hàng, các nhà đầu tư v.v.. (Giống như trong xây dựng người ta dùng các bản vẽ thiết kế để hướng dẫn và kiểm soát thi công, bán hàng  căn hộ v.v..)

1. **Tại sao lại là OOAD và UML?**

OOAD cần các bản vẽ để mô tả hệ thống được thiết kế, còn UML là ngôn ngữ mô tả các bản vẽ nên cần nội dung thể hiện.  Do vậy, chúng ta phân tích và thiết kế theo hướng đối tượng và sử dụng UML để biểu diễn các thiết kế đó nên chúng thường đi đôi với nhau.

1. **OOAD sử dụng UML**

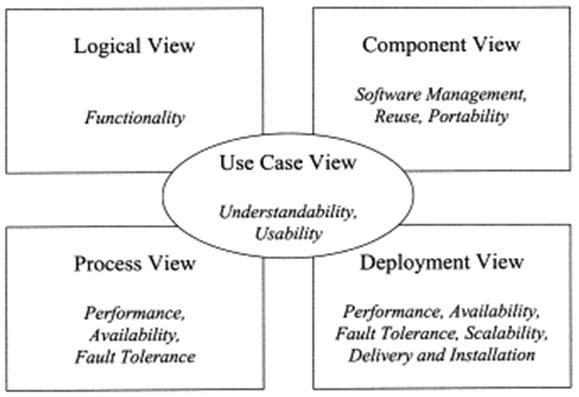
UML sử dụng để vẽ cho nhiều lĩnh vực khác nhau như phần mềm, cơ khí, xây dựng v… trong phạm vi các bài viết này chúng ta chỉ nghiên cứu cách sử dụng UML cho phân tích và thiết kế hướng đối tượng trong ngành phần mềm. OOAD sử dụng UML bao gồm các thành phần sau:

* View (góc nhìn)
* Diagram (bản vẽ)
* Notations (ký hiệu)
* Mechanisms (qui tắc, cơ chế)

Chúng ta sẽ tìm hiểu kỹ hơn các thành phần trên.

* **4.1 View (góc nhìn)**

Mỗi góc nhìn như thầy bói xem voi, nó không thể hiện hết hệ thống nhưng thể hiện rõ hệ thống ở một khía cạnh. Chính vì thế trong xây dựng có bản vẽ kiến trúc (nhìn về mặt kiến trúc), bản vẽ kết cấu (nhìn về mặt kết cấu), bản vẽ thi công (nhìn về mặt thi công). Trong phần mềm cũng như vậy, OOAD sử dụng UML có các góc nhìn sau:



Hình 1. Các View trong OOAD sử dụng UML

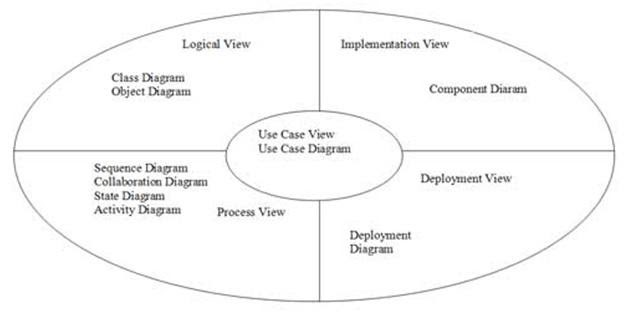
Trong đó,

* ***Use Case View***: cung cấp góc nhìn về các ca sử dụng giúp chúng ta hiểu hệ thống có gì? ai dùng và dùng nó như thế nào.
* ***Logical View***: cung cấp góc nhìn về cấu trúc hệ thống, xem nó được tổ chức như thế nào. Bên trong nó có gì.
* ***Process View:***cung cấp góc nhìn động về hệ thống, xem các thành phần trong hệ thống tương tác với nhau như thế nào.
* ***Component View:***Cũng là một góc nhìn về cấu trúc giúp chúng ta hiểu cách phân bổ và sử dụng lại các thành phần trong hệ thống ra sao.
* ***Deployment View***: cung cấp góc nhìn về triển khai hệ thống, nó cũng ảnh hưởng lớn đến kiến trúc hệ thống.

Tập hợp các góc nhìn này sẽ giúp chúng ta hiểu rõ hệ thống cần phân tích, thiết kế. Trong hình 1 chúng ta thấy góc nhìn Use Case View nằm ở giữa và chi phối tất cả các góc nhìn còn lại. Chính vì thế chúng ta thường thấy các tài liệu nói về 4 view + 1 chứ không phải 5 view nhằm nhấn mạnh vai trò của Use Case View.

* **4.2  Diagram (Bản vẽ)**

Diagram các bạn có thể dịch là sơ đồ. Tuy nhiên ở đây chúng ta sử dụng từ bản vẽ cho dễ hình dung. Các bản vẽ được dùng để thể hiện các góc nhìn của hệ thống.



Hình 2. Các bản vẽ trong OOAD sử dụng UML

Trong đó,

* **Use Case Diagram**: bản vẽ mô tả về ca sử dụng của hệ thống. Bản vẽ này sẽ giúp chúng ta biết được ai sử dụng hệ thống, hệ thống có những chức năng gì. Lập được bản vẽ này bạn sẽ hiểu được yêu cầu của hệ thống cần xây dựng.
* **Class Diagram**: bản vẽ này mô tả cấu trúc của hệ thống, tức hệ thống được cấu tạo từ những thành phần nào. Nó mô tả khía cạnh tĩnh của hệ thống.
* **Object Diagram**: Tương tự như Class Diagram nhưng nó mô tả đến đối tượng thay vì lớp (Class).
* **Sequence Diagarm**: là bản vẽ mô tả sự tương tác của các đối tượng trong hệ thống với nhau được mô tả tuần tự các bước tương tác theo thời gian.
* **Collaboration Diagram**: tương tự như sequence Diagram nhưng nhấn mạnh về sự tương tác thay vì tuần tự theo thời gian.
* **State Diagram:**bản vẽ mô tả sự thay đổi trạng thái của một đối tượng. Nó được dùng để theo dõi các đối tượng có trạng thái thay đổi nhiều trong hệ thống.
* **Activity Diagram:** bản vẽ mô tả các hoạt động của đối tượng, thường được sử dụng để hiểu về nghiệp vụ của hệ thống.
* **Component Diagram:**bản vẽ mô tả về việc bố trí các thành phần của hệ thống cũng như việc sử dụng các thành phần đó.
* **Deployment Diagram:** bản vẽ mô tả việc triển khai của hệ thống như việc kết nối, cài đặt, hiệu năng của hệ thống v.v…

Chúng ta sẽ bàn kỹ các bản vẽ này trong các bài tiếp theo. Vì nó chính là hạt nhân của loạt bài này.

**Lưu ý**: Ở đây chúng ta sử dụng từ hệ thống tương đương với sản phẩm phần mềm.

* **4.3 Notations (các ký hiệu)**

Notations là các ký hiệu để vẽ, nó như từ vựng trong ngôn ngữ tự nhiên. Bạn phải biết từ vựng thì mới ghép thành câu, thành bài được. Chúng ta sẽ tìm hiểu kỹ các notations trong từng bản vẽ sau này. Dưới đây là vài ví dụ về notation.



Hình 3. Ký hiệu về Use Case



Hình 4. Ký hiệu về Class



Hình 5. Ký hiệu về Actor

* **4.4 Mechanisms (Rules)**

Mechanisms là các qui tắc để lập nên bản vẽ, mỗi bản vẽ có qui tắc riêng và bạn phải nắm được để tạo nên các bản vẽ thiết kế đúng. Các qui tắc này chúng ta sẽ bàn kỹ trong các bài về các bản vẽ.

1. **Kết luận**

Nguyên tắc phân tích, thiết kế một hệ thống phần mềm cũng không khác việc xây dựng một cái nhà trong xây dựng. Bạn hãy nhớ cách tiếp cận này để dễ hiểu hơn trong việc phân tích và thiết kế hệ thống. Hãy giữ mọi thứ cho thật đơn giản để dễ hiểu và dễ áp dụng.