ÁP DỤNG MẪU THIẾT KẾ HƯỚNG ĐỐI TƯỢNG TRONG PHÁT TRIỂN PHẦN MỀM WEB

Trần Đan Thư, Huỳnh Thụy Bảo Trân
Trường Đại Học Khoa Học Tự Nhiên, ĐHQG-HCM
(Bài nhận ngày 20 tháng 05 năm 2007, hoàn chỉnh sửa chữa ngày 08 tháng 09 năm 2007)

***TÓM TẮT**: Các ứng dụng trên môi trường Web ngày càng đóng vai trò quan trọng và thay thế dần các phần mềm truyền thống nhằm hỗ trợ tốt hơn các hoạt động đa dạng trong đời sống xã hội. Trong những năm gần đây, việc xây dựng và phát triển các ứng dụng Web được quan tâm đúng mức bởi giới công nghiệp cũng như các nhà nghiên cứu. Sự phát triển của phương pháp luận cũng như công nghệ và kỹ thuật hướng đối tượng đã tạo nhiều điều kiện thuận lợi để phát triển phần mềm Web nhờ sử dụng lại các lớp đối tượng xây dựng sẵn. Trong bài báo này, chúng tôi trình bày kết quả nghiên cứu về việc vận dụng và phát triển các mẫu thiết kế hướng đối tượng trong tiến trình xây dựng và phát triển các phần mềm chạy phân bố trên môi trường Web.*

***Từ khóa**: Kỹ thuật hướng đối tượng, Mẫu thiết kế, Ứng dụng Web, Ứng dụng phân bố.*

1.GIỚI THIỆU

Trong những năm gần đây, phần mềm ứng dụng chạy trên Web ngày càng được sử dụng rộng rãi và ưa chuộng xuất phát từ tính tiện dụng và rộng khắp của hệ thống mạng Internet toàn cầu. Phần mềm Web có giao diện đẹp và dễ sử dụng, không cần cài đặt và có thể được khai thác ở bất cứ nơi nào đang có sẵn Internet. Hơn nữa, phần mềm Web – xuất phát từ bản chất công nghệ của chúng – đã giải quyết được nhiều vấn đề về chia sẻ và phân phối tài nguyên chung cũng như các vấn đề về truy cập và khai thác tài nguyên ở xa. Việc phát triển các ứng dụng chạy trên Web ngày càng chiếm tỉ lệ cao so với phát triển các phần mềm truyền thống. Các hợp đồng gia công phần mềm Web ngày càng gia tăng đáng kể.

Về mặt công nghệ, nhiều kỹ thuật hỗ trợ phát triển ứng dụng Web được đề xuất: khởi đầu là các trang Web tĩnh *HTML*, kế đến là các trang Web động *DHTML*, sau đó là các ngôn ngữ script như *JavaScript* và *VBScript* cho phép lập trình trong các trang HTML, và hiện nay là các trang *PHP*, *JSP*, *ASP* hay *ASP.NET* [14,10,18,23]. Đặc biệt, sự ra đời của ngôn ngữ *XML (Extensible Markup Language* [26]) đã tạo một bước ngoặc quan trọng cho kỹ thuật biểu diễn văn bản phức hợp và đối tượng phân bố. Công nghệ *AJAX (Asynchronous JavaScript and XML* [5,13]) cũng được đề xuất: công nghệ này góp phần cải tiến giao diện người dùng trong các ứng dụng Web, tạo điều kiện thuận lợi để người sử dụng khai thác các ứng dụng Web dễ dàng hơn.

Tuy nhiên, sự phát triển đa dạng của những công nghệ mới cùng với nhu cầu rất lớn về số lượng các ứng dụng Web làm nảy sinh một số vấn đề trong xây dựng phần mềm Web [10]. Nguyên do của những vấn đề này là khi sử dụng một công nghệ cụ thể với những hạn chế nhất định, các kỹ sư phần mềm hay lập trình viên đã không đầu tư thời gian (hoặc là áp lực quá lớn của hạn định giao nộp không cho phép họ có thời gian) để thiết kế tốt kiến trúc ứng dụng, phát triển các "chương trình sạch" (*clean code*) dễ bảo trì, có thể tái sử dụng và mở rộng. Các kỹ sư phần mềm thường chọn ngay các giải pháp cứu cánh trước mắt để khắc phục hạn chế công nghệ mà hướng tới mục tiêu đáp ứng hạn định giao nộp. Rất nhiều ứng dụng Web rất lớn, đã

được xây dựng và vận hành, đang lâm vào tình trạng rất khó bảo trì và mở rộng. Chúng tôi nghiên cứu tổng hợp các vấn đề nảy sinh và đề xuất kỹ thuật để hạn chế, khắc phục những vấn đề này với mục đích hỗ trợ cho người phát triển ứng dụng Web.

Về mặt phương pháp luận, các phương pháp phân tích thiết kế hướng đối tượng [22] đã phát triển rất mạnh mẽ và góp phần đáng kể vào việc cải tiến chất lượng của phần mềm nhờ vào khả năng xây dựng các lớp đối tượng có tính tái sử dụng cao, dễ bảo trì và mở rộng. Ngôn ngữ UML *(Unified Modeling Language* [17]) được đề xuất để sử dụng như một ngôn ngữ chuẩn để mô hình hóa các thành tố phần mềm trong quá trình phân tích thiết kế hướng đối tượng. Một số tác giả cũng nghiên cứu vận dụng phương pháp hướng đối tượng, cụ thể hóa các phương pháp này để phù hợp với tiến trình xây dựng các ứng dụng Web [3,7,10].

Tuy nhiên, các phương pháp hướng đối tượng tập trung chủ yếu vào các hoạt động tổng thể trong tiến trình phát triển phần mềm hướng đối tượng. Những phương pháp này thường không giải quyết các vấn đề chi tiết nảy sinh trong quá trình thiết kế phần mềm. Để bổ sung cho phương pháp hướng đối tượng, các mẫu thiết hướng đối tượng (*mẫu thiết kế GoF*, Gamma và cộng sự [12]) là một tiếp cận độc đáo, được đề xuất để giải quyết các vấn đề nảy sinh trong quá trình thiết kế phần mềm hướng đối tượng. Các mẫu GoF có tầm quan trọng và ảnh hưởng rất lớn đối với giới nghiên cứu cũng như giới công nghiệp phần mềm. Rất nhiều công trình đặc sắc khác về mẫu thiết kế hướng đối tượng được đề xuất để giải nhiều vấn đề đặc thù cho từng lĩnh vực ứng dụng phần mềm [2,4,11,21]. Chúng tôi quan tâm đến việc nghiên cứu các mẫu thiết kế hướng đối tượng để áp dụng trong quá trình phát triển phần mềm hướng đối tượng, đặc biệt là giải quyết các vấn đề về cài đặt giao diện người dùng và các vấn đề liên quan đến các ứng dụng phân bố trên Internet [6,24,25].

Trong bài báo này, chúng tôi trình bày việc nghiên cứu áp dụng mẫu thiết kế hướng đối tượng trong quá trình xây dựng các ứng dụng Web. Xuất phát từ các đặc điểm công nghệ của của môi trường Web, các mẫu thiết kế cần phải được vận dụng và phát triển một cách thích hợp. Phần 2 của bài báo sẽ phân tích các công nghệ phát triển ứng dụng Web và các vấn đề nảy sinh xuất phát từ hạn chế của công nghệ. Trong phần 3, chúng tôi giới thiệu các mẫu thiết kế thường được sử dụng cho ứng dụng Web để giải quyết các vấn đề nảy sinh. Kế đến chúng tôi sẽ minh họa việc sử dụng mẫu thiết kế trong phần 4. Sau cùng, trong phần 5, chúng tôi tổng kết và đề xuất hướng phát triển cho chủ đề nghiên cứu này trong tương lai.

2.CÔNG NGHỆ WEB VÀ CÁC VẤN ĐỀ NẢY SINH

Phần này sẽ tóm tắt hiện trạng công nghệ của việc thiết kế và hiện thực các ứng dụng Web. Các mô hình công nghệ, ngôn ngữ lập trình và môi trường hỗ trợ được tiến hóa liên tục để nâng cao hiệu suất lao động của lập trình viên cũng như cải tiến chất lượng của ứng dụng Web được xây dựng. Tuy nhiên, một số vấn đề nhất định đã nảy sinh do sự hạn chế bản chất của một số công nghệ cụ thể. Mặc dù có nhiều giải pháp công nghệ đề xuất để giải quyết những vấn đề này, nhưng những giải pháp về phương pháp luận luôn đóng vai trò rất quan trọng, được vận dụng để bổ sung và hỗ trợ đúng lúc cho các giải pháp công nghệ.

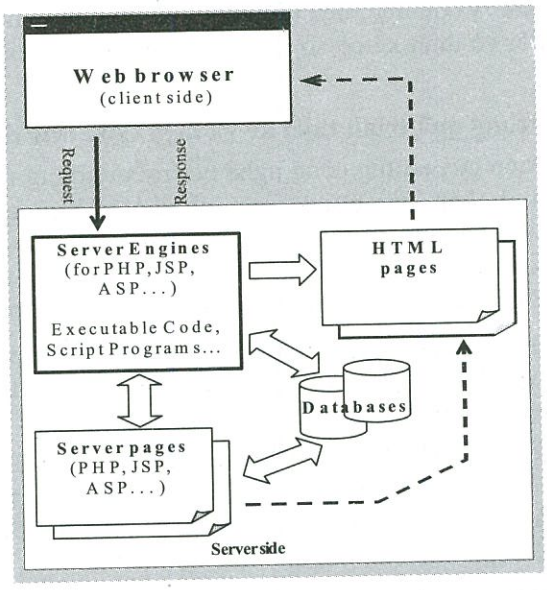
2.1.Đặc điểm của công nghệ xây dựng ứng dụng web

Công nghệ Web được khởi đầu với những trang *HTML* tĩnh, chỉ nhằm mục đích trình bày các thông tin quảng bá của các công ty hay tổ chức nào đó. Những trang này được đặt tại các máy chủ Web (*Web servers*) và cho phép truy cập thông qua Internet. Việc giao tiếp với người sử dụng chủ yếu nhờ các liên kết trang: từ một trang Web đi đến trang khác nhờ lựa chọn liên kết trang. Do tính tiện dụng và rộng khắp của mạng Internet toàn cầu, những nhà sản xuất phần mềm có ý tưởng phát triển các hệ thống phần mềm có thể vận hành trên Internet. Ý tưởng độc

đáo này được thực hiện rất thành công với những *hệ thống thư tín điện tử trên Web*. Sau đó là các hệ thống bán hàng, kinh doanh bất động sản, đặt vé tàu, quản lý tài khoản ngân hàng... trên Internet.

Với nhu cầu ngày càng cao của khách hàng, các hãng sản xuất phần mềm nỗ lực nâng cao chất lượng của các hệ thống Web nhằm mục đích đạt được các ứng dụng trên Web có thể hoạt động như các phần mềm thông thường với nhiều chức năng giao tiếp đa dạng với người sử dụng hơn là chỉ gói gọn trong các liên kết trang. Nỗ lực này được khởi đầu với các chương trình *Perl*, *Common Gateway Interface (CGI)* và *C* chạy trên các *Web server*; kế đến là các công nghệ phát triển ứng dụng Web, chẳng hạn như: *JavaScript, VBScript, PHP, JSP, ASP*...

Hình 1 trình bày kiến trúc tổng thể của ứng dụng Web. Về mặt kỹ thuật, khi người sử dụng truy cập đến một địa chỉ *WWW* thông qua một trình duyệt Web (*Web browser*) chạy trên máy khách (*client computer*), yêu cầu sẽ được gởi đến máy chủ Web. Đối với trường hợp trang Web tĩnh thì yêu cầu được đáp ứng ngay bằng cách gởi ngay trở lại trang *HTML* cho máy khách. Trường hợp trang Web động: máy chủ Web sẽ chạy một chương trình thích hợp để thực hiện một số thao tác nhất định nhằm phát sinh ra trang *HTML*, sau đó trang này được gởi đến máy khách.



Hình 1. Kiến trúc tổng thể của các ứng dụng Web

Có 2 dạng chính cho chương trình ở máy chủ Web để phát sinh ra trang *HTML*: chương trình script (*script program*) hay trang server (*server page*).

- Chương trình *script*: như chương trình bình thường được lập trình theo giao thức *HTTP* (*Hyper Text Transfer Protocol*), sử dụng các lệnh dạng chuỗi để phát sinh ra trang *HTML* theo yêu cầu. Trong thực tế, ngoài các script như *CGI Script hay JavaServlet*, có thể là một chương trình bất kỳ, lập trình giao thức *HTTP*, được dịch thành mã thực thi và cài đặt tại máy chủ Web.

- Trang *server*: chương trình được cấu trúc bằng cách nhúng các mã script vào trang *HTML* dự kiến sẽ trả về. Khi có yêu cầu từ máy khách, các đoạn mã script được thực hiện để

phát sinh ra trang *HTML* thực sự được gởi về trình duyệt Web đang chạy ở máy khách. Ví dụ cho dạng này là các công nghệ *PHP, ASP, và JSP.*

Hai dạng chương trình nói trên có những lợi thế và hạn chế riêng của mỗi loại. Các chương trình script thì thiên về xử lý các yêu cầu và thực hiện các tính toán; người lập trình sẽ rất cực nhọc nếu sử dụng dạng chương trình này quá nhiều để thực hiện việc bố trí, định dạng kiểu cách trang *HTML*. Ngược lại, các trang *server* rất thuận lợi cho việc định dạng trang *HTML*; nhưng nếu người lập trình chèn tùy tiện các mã *script* vào trang *HTML* thì sẽ đưa đến nhiều vấn đề khó khăn trong việc bảo trì và phát triển hệ thống sau này.

Đối với các ứng dụng Web có chức năng lưu trữ và quản lý dữ liệu, tùy theo mức độ mà chúng ta có thể sử dụng hệ thống tập tin trên máy chủ hay là một cơ sở dữ liệu thực sự. Hệ thống dữ liệu được truy xuất và cập nhật thông qua các chương trình *script* hay là các *trang server* (xem hình 1). Khi dùng hệ thống tập tin của máy chủ, tùy trường hợp người thiết kế có thể tổ chức các tập tin cấu trúc. Trong những năm gần đây, tập tin dạng *XML* thường được chọn để lưu trữ dữ liệu có cấu trúc phức hợp. Việc tổ chức cơ sở dữ liệu được thực hiện tùy theo qui mô của ứng dụng. Đối với ứng dụng nhỏ: có thể quản lý tập trung bởi một hệ quản trị cơ sở dữ liệu. Tuy nhiên, với các ứng dụng Web có qui mô lớn thì dữ liệu có thể được quản lý tập trung hay phân tán trên nhiều máy chủ khác nhau. Bên cạnh đó, việc thực hiện truy xuất dữ liệu cũng cần được coi trọng. Ví dụ như, nếu các truy vấn *SQL* được nhúng trực tiếp vào trang *PHP*, thì khi có sự thay đổi về thiết kế cơ sở dữ liệu thì chúng ta phải tìm và sửa lại toàn bộ các câu truy vấn đó.

2.2.Các vấn đề nảy sinh trong quá trình thiết kế và hiện thực ứng dụng Web

Từ các đặc điểm nêu trên của những công nghệ hỗ trợ xây dựng ứng dụng Web cộng với sự thiếu định hướng phương pháp luận trong tiến trình phát triển ứng dụng, nhiều vấn đề đã nảy sinh và dẫn đến các ứng dụng cồng kềnh, khó bảo trì, khó mở rộng. Trong phần này, chúng tôi sẽ hệ thống lại các vấn đề quan trọng nhất dựa trên những nghiên cứu, quan sát và phân tích theo quan điểm phát triển công nghệ phần mềm một cách bền vững [2, 10, 11, 24].

- **Mã hóa cứng** *(hard coding)* những đoạn mã nguồn chỉ dùng một lần, không thể tái sử dụng hay mở rộng để dùng cho các tình huống tương tự nhưng có thay đổi chút ít. Mã hóa cứng sẽ gây trở ngại cho việc mở rộng hay nâng cấp hệ thống sau này;
- **Trùng lặp mã nguồn** (*code duplication*): kết quả của việc sao chép vật lý các đoạn mã, chỉ sửa lại tên hay giá trị biến. Trường hợp này cũng xảy ra khi lập trình viên sao chép các trang HTML (có chèn mã nguồn) để sửa lại theo yêu cầu nào đó. Các hệ thống phần mềm trùng lắp mã nguồn rất khó bảo trì và chỉnh sửa;
- **Trộn lẫn mã nguồn** nghiệp vụ của ứng dụng với mã nguồn phát sinh trang Web: không tách biệt được tính trừu tượng của nghiệp vụ với các chi tiết kỹ thuật liên quan đến công nghệ đang dùng để xây dựng ứng dụng. Hệ thống sẽ khó sửa đổi khi có các yêu cầu mở rộng về mặt nghiệp vụ của ứng dụng đang phát triển.
- **Nhúng trực tiếp các câu lệnh SQL vào các trang server**: cách làm này gắn chặt các trang server với mô hình cơ sở dữ liệu của ứng dụng. Một sửa đổi nhỏ trong mô hình dữ liệu có khả đưa đến việc sửa rất nhiều trang server của ứng dụng đang phát triển.
- **Chèn trực tiếp các công thức tính toán, biểu thức kiểm tra điều kiện ràng buộc dữ liệu vào các trang server**: các ràng buộc về nghiệp vụ ứng dụng trộn lẫn với việc trình bày trang HTML. Việc thay đổi trong nghiệp vụ ứng dụng có thể dẫn đến sửa đổi nhiều trang Web.

Về mặt phương pháp luận, người phát triển ứng dụng phải làm cách nào đó để tránh được các vấn đề nói trên. Để làm được như vậy họ cần phải được hỗ trợ về kỹ thuật thiết kế ứng

dụng, tổ chức các chương trình script, thiết lập các trang server sao cho không gặp phải các vấn đề này.

2.3.Phương pháp hướng đối tượng phát triển ứng dụng Web

Một số phương pháp hướng đối tượng hỗ trợ tiến trình xây dựng ứng dụng Web [3,7] đã được đề xuất. Ngôn ngữ *UML* được mở rộng (*Web Application Extension for UML – WAE)* để mô hình hóa các thành phần của ứng dụng Web được sản sinh trong quá trình phân tích thiết kế. Các hoạt động tham gia vào các pha xây dựng phần mềm trong tiến trình *RUP (Rational Unified Process)* được vận dụng để phát triển ứng dụng Web. Các phương pháp này góp phần đáng kể vào việc nâng cao chất lượng của ứng dụng Web, giải quyết một phần nào các vấn đề nảy sinh do hiện trạng công nghệ Web. Tuy nhiên, các phương pháp hướng đối tượng cho ứng dụng Web chỉ hỗ trợ các bước tổng thể cần phải thực hiện trong tiến trình thiết kế. Các phương pháp này không hỗ trợ nhiều cho việc giải quyết các vấn đề cụ thể nảy sinh trong thiết kế và hiện thực ứng dụng Web.

3.VẬN DỤNG MẪU THIẾT KẾ TRONG TIẾN TRÌNH XÂY DỰNG ỨNG DỤNG WEB

Trong phần này, chúng tôi trình bày kỹ thuật sử dụng mẫu thiết kế hướng đối tượng để giải quyết các vấn đề nảy sinh do hiện trạng của các công nghệ xây dựng ứng dụng Web. Trước tiên, chúng tôi nhắc lại các mẫu GoF [12] bởi vì đây là những mẫu cơ sở cho phát triển phần mềm hướng đối tượng nói chung, hơn nữa có thể sử dụng hiệu quả khi phát triển ứng dụng Web. Kế đến chúng tôi trình bày về các mẫu thiết kế tiêu biểu cho ứng dụng Web. Sau cùng một số thư viện lớp cho ứng dụng Web được giới thiệu. Những thư viện lớp này được phát triển dựa trên nền tảng các mẫu thiết kế.

3.1.Mẫu thiết kế hướng đối tượng GoF

Gamma và cộng sự đã đề xuất 23 mẫu thiết kế cơ sở (thường được gọi là mẫu GoF [12]). Mỗi mẫu GoF giải quyết một vấn đề cụ thể nào đó trong tiến trình xây dựng phần mềm hướng đối tượng. Trong thực tế, một nhóm mẫu GoF thường được sử dụng phối hợp nhau để giải quyết các vấn đề thiết kế. Những mẫu này đã được giới công nghiệp sử dụng rất hiệu quả để thiết lập các mô hình công nghệ phục vụ cho việc xây dựng các phần mềm. Các kỹ sư phần mềm cũng vận dụng thường xuyên các mẫu này trong quá trình thiết kế phần mềm. Đa số mẫu thiết kế hiện nay đều có nguồn gốc từ mẫu GoF, có thể là dạng biến thể của một mẫu GoF hay là sự phối hợp một cách hợp lý các mẫu GoF để giải quyết các vấn đề trong thiết kế hướng đối tượng.

Bởi vì các chương trình Web được viết bằng ngôn ngữ script, kỹ thuật cài đặt các mẫu GoF có thể không hoàn giống như hướng dẫn của Gamma trong tài liệu tham khảo [12]. Trong phạm vi bài báo này chúng tôi chỉ tóm tắt lại về ý nghĩa và ứng dụng của những mẫu GoF tiêu biểu thường được sử dụng khi xây dựng các ứng dụng Web.

- Sự phát triển đa dạng của công nghệ Web đưa đến nhiều hệ thống thư viện lớp khác nhau phục vụ cho cùng một mục đích, do đó những đoạn mã nguồn sử dụng trực tiếp các lớp của một thư viện cụ thể sẽ không thể dùng lại một cách độc lập với thư viện đó. Mẫu Adapter dùng để tạo giao diện lập trình không phụ thuộc một thư viện lớp cụ thể.
- Mẫu State thường được dùng để cài đặt các đối tượng Web có hành vi thay đổi theo trạng thái của chúng, hoạt động của đối tượng được điều khiển bằng cách thay đổi trạng thái khi thích hợp.
- Mẫu Brigde được sử dụng để tách biệt tính trừu tượng về mặt nghiệp vụ khỏi các chi tiết kỹ thuật cài đặt phụ thuộc công nghệ Web. Nói chung, mẫu này được dùng rất hiệu quả

nhằm tách biệt (không trộn lẫn) 2 phạm trù tương đối độc lập nhau, chẳng hạn: giao diện người dùng và xử lý về mặt nghiệp vụ, trình bày dữ liệu và biểu diễn dữ liệu...

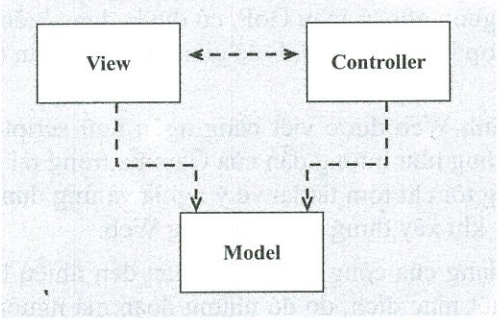
- Mẫu Singleton nhằm cài đặt các lớp chỉ có duy nhất một đối tượng, ngăn ngừa được việc tạo đối tượng thứ hai trở đi (nếu có tạo thì chỉ trả về đối tượng đã tạo rồi). Mẫu này thường được dùng trong các ứng dụng Web cần thiết cài đặt các lớp một đối tượng bởi vì việc tạo nhiều đối tượng không cần thiết sẽ hao tốn bộ nhớ.
- Các ứng dụng Web thường dùng mẫu Strategy để giải quyết việc lựa chọn các hình thức hiển thị khác nhau (nhưng giống cấu trúc) của cùng một trang Web, chẳng hạn như việc hỗ trợ nhiều ngôn ngữ và đơn vị tiền tệ cho một trang Web.

Ngoài các mẫu GoF nói trên, một số mẫu khác như *Prototype* (đối tượng sao chép; cloning), *Composite* (đối tượng phức hợp), *Template method* (phương thức khuôn, xử lý tổng quát) cũng thường được sử dụng trong xây dựng ứng dụng Web. Tuy nhiên, các mẫu này được hỗ trợ đương nhiên bởi các ngôn ngữ lập trình *script* và các công nghệ Web tiên tiến.

3.2.Mẫu thiết kế cho ứng dụng web

Mẫu thiết kế tiêu biểu và đóng vai cho quan trọng nhất trong công nghệ phát triển ứng dụng Web là mẫu *MVC (Model-View-Controller* [10, 11, 19]). Mẫu này có ý tưởng xuất phát từ mẫu Observer của Gamma và cộng sự [12]. Thực ra từ cuối những năm 70, trước khi các mẫu GoF được công bố, ý tưởng này đã được triển khai thành công cho các thư viện đồ họa trên ngôn ngữ lập trình Smalltalk. Tuy nhiên, trong khi mẫu Observer được đề xuất để giải quyết việc trình bày dữ liệu cho các ứng dụng truyền thống trên máy đơn, thì mẫu MVC dùng để thiết lập kiến trúc của ứng dụng trên Web.

Mô hình tổng thể của mẫu *MVC* được trình bày như trong hình 2. Chú ý rằng các mũi tên đứt nét có ý nghĩa là "*có thể truy xuất*" hay là "*biết thông tin về*": *View* và *Controller* truy xuất được lẫn nhau, cùng biết thông tin về *Model*; tuy nhiên *Model* không thể truy xuất đến *View* cũng như *Controller*. Theo mô hình này, mỗi thành phần của ứng dụng Web được tổ chức tách biệt thành 3 phần: *Model* (mô hình bên trong), *View* (hiển thị bên ngoài), *Controller* (bộ điều khiển nhập xuất và cập nhật phần hiển thị). Nhờ sự tách biệt ba khía cạnh này mà mẫu *MVC* giải quyết được nhiều vấn đề nảy sinh khi phát triển ứng dụng Web.



Hình 2. Mô hình tổng thể của mẫu MVC [10, 11]

- *Mô hình trong (Model)*: là đối tượng biểu diễn thông tin nghiệp vụ bên trong ứng dụng đang xây dựng. Đối tượng này bao bọc các thành phần dữ liệu và các phương thức liên quan đến ứng xử của nó. Khi phát triển các lớp đối tượng này, người lập trình chỉ quan tâm cài đặt các xử lý hay tiến trình tác nghiệp của ứng dụng mà không cần quan tâm đến chúng được hiển thị ra các thiết bị xuất hay lấy vào từ thiết bị nhập như thế nào.

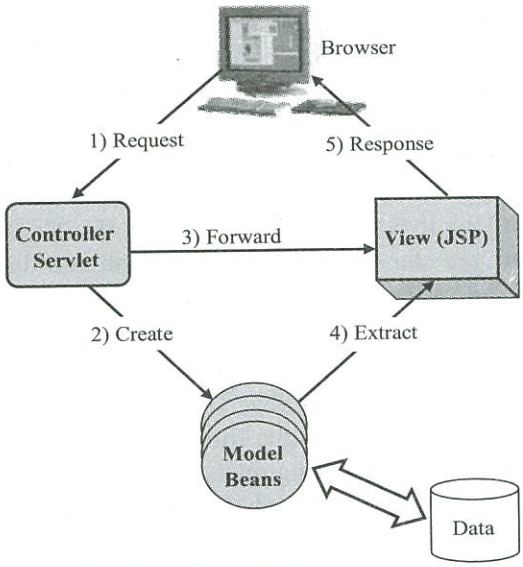
- *Hiển thị bên ngoài (View)*: là thành phần liên quan đến giao diện người dùng. Người sử dụng "thấy" được đối tượng nghiệp vụ bên trong ứng dụng nhờ phần hiển thị (tức là *View*) của nó. Đối tượng có thể được hiển thị dưới dạng một trang *HTML*, một hộp chọn (*listbox*), hay một danh sách chọn dạng cây (*tree view*)...

- *Bộ điều khiển (Controller)*: đảm nhiệm việc cập nhật bộ phận hiển thị (*View*) khi cần thiết. Bộ điều khiển này nhận dữ liệu nhập từ người dùng, truy xuất các thông tin cần thiết từ mô hình trong (*Model*), và cập nhật thích hợp phần hiển thị (*View*).

Giao diện với người sử dụng phần mềm được thiết lập nhờ sự tương tác qua lại giữ *View* và *Controller*: hai bộ phận này chính là phần trình bày bên ngoài của đối tượng biểu diễn bên trong. Người sử dụng chỉ biết về đối tượng bên trong thông qua phần bên ngoài là *View* và *Controller*.

Trong mô hình *MVC*, sự tách biệt giữa phần trình bày (*View* và *Controller*) khỏi phần biểu diễn trong (*Model*) chính là yếu tố quan trọng góp phần nâng cao chất lượng thiết kế phần mềm. Yếu tố này khắc phục được các vấn đề đã được thảo luận trong phần trước: tách biệt được mã nguồn liên quan đến nghiệp vụ ứng dụng và mã nguồn giao diện người dùng, tạo cơ chế để tránh được mã hóa cứng và trùng lặp mã nguồn, sự sửa đổi về mô hình trong không ảnh hưởng dây chuyền đưa đến việc sửa đổi nhiều phần giao diện người dùng bên ngoài. Ứng dụng có thể phát triển và mở rộng: với cùng một mô hình trong có thể có nhiều hình thức giao tiếp bên ngoài với người sử dụng (trình duyệt Web, giao tiếp dòng lệnh, hiển thị đồ họa...).

Hầu hết các công nghệ hỗ trợ phát triển ứng dụng Web hiện đại đều sử dụng mô hình *MVC* để định hướng cho người phát triển phần mềm Web thiết kế được các ứng dụng dễ mở rộng và bảo trì sau này. Hình 3 trình bày một thể hiện của mô hình *MVC* cho công nghệ *JSP*, các mũi tên trong hình này chỉ hướng đi của thông điệp hay dữ liệu. Chu kỳ thực hiện một yêu cầu từ máy khách bao gồm các bước như sau.



Hình 3. Thể hiện của mẫu MVC trong công nghệ JSP [10]

• Bước 1 (*Request*): từ máy khách, người sử dụng thực hiện thao tác phát sinh yêu cầu đến bộ điều khiển (*Controller servlet*) ở máy chủ Web ;

• Bước 2 (*Create*): bộ điều khiển (*Controller servlet*) tạo một hay nhiều đối tượng JavaBean ;

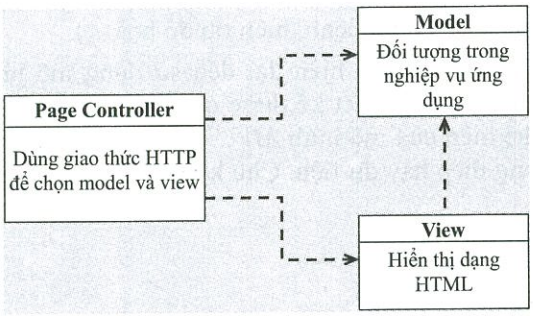
• Bước 3 (*Forward*): bộ điều khiển (*Controller servlet*) chuyển điều khiển đến bộ hiển thị (View – JSP) ;

• Bước 4 (*Extract*): bộ hiển thị (*View – JSP*) trích thông tin từ đối tượng JavaBean ;

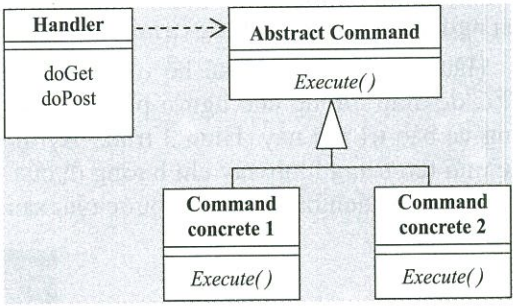
• Bước 5 (*Response*): bộ hiển thị (*View – JSP*) gởi thông tin (trang HTML) về máy khách để hiển thị trong trình duyệt Web.

Trong bước 4 và bước 5, các đối tượng *JavaBean* có thể giao tiếp với hệ thống lưu trữ dữ liệu (hệ thống tập tin hoặc là cơ sở dữ liệu trên máy chủ Web) để truy cập các thông tin cần thiết.

Mẫu *MVC* có thể được thể hiện bằng nhiều cách khác nhau tùy thuộc vào người thiết kế và tùy thuộc vào yêu cầu thực tiễn. Hình 4 mô tả mẫu *Page controller* [11]. Bộ điều khiển trang (*Page controller*) sử dụng giao thức *HTTP* (trực tiếp hay gián tiếp) để chọn đối tượng nghiệp vụ (*Model*) và bộ hiển thị thích hợp (*View*), bộ hiển thị sử dụng thông tin từ đối tượng nghiệp vụ để tạo trang HTML được gởi về máy khách. Mẫu *Page controller* có thể được cài đặt bằng cách dùng chương trình script (như *CGI script hay servlet*) hay là các trang *server*.



Hình 4. Mẫu Page controller [11]



Hình 5. Mẫu Front controller [11]

Trường hợp bộ điều khiển cần phải xử trí nhiều yêu cầu đa dạng cho một trang Web, mẫu *Front controller* [11] thường được sử dụng (xem hình 5). Mẫu này cấu trúc gồm 2 phần: một thẻ Web (*web handler*) và một lệnh thay đổi được vào lúc thực thi. Thẻ Web đón nhận các yêu cầu từ máy chủ Web, tùy theo yêu cầu này mà quyết định lệnh nào được thực hiện (nhờ tận dụng cơ chế đa hình). Mẫu *Front controller* được cài đặt bằng các lớp đối tượng thay vì dùng các trang *server*.

Các mẫu nói trên, đặc biệt là mẫu *MVC*, được sử dụng thông dụng ở mức độ thiết lập kiến trúc ứng dụng Web. Một số nhóm nghiên cứu khác [1, 8, 9, 15] cũng đề xuất các mẫu giải quyết những vấn đề cụ thể hơn, xuất hiện trong quá trình xây dựng ứng dụng Web. Chẳng hạn mẫu *Abstract form* [1] đề xuất một mô hình tổng quát cho bảng nhập liệu, có thể sử dụng để tạo các bảng nhập liệu cho nhiều ứng dụng khác nhau. Mẫu *Remote Authenticator/Authorizer* [9] cung cấp cơ chế kiểm tra quyền truy cập hệ thống hợp lệ. Đặc biệt, nhóm nghiên cứu [16, 20] đang tiến hành một đề án xây dựng thư viện mẫu thiết kế giao diện người dùng cho các ứng dụng Web nhằm mục đích hỗ trợ việc phát triển các ứng dụng Web.

3.3.Thư viện lớp cho ứng dụng web dựa trên mẫu thiết kế

Một số hệ thống thư viện lớp (được gọi là *class framework*) được phát triển dựa trên kiến trúc *MVC* nhằm hỗ trợ cho người phát triển ứng dụng Web. Bảng 1 giới thiệu một số

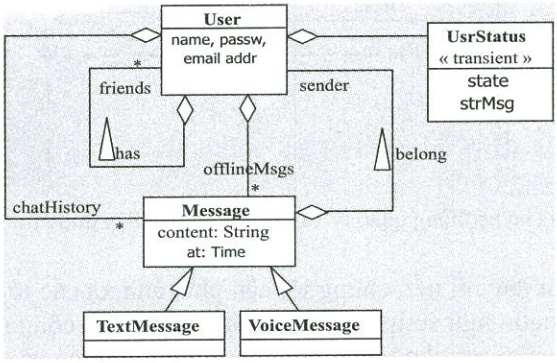
framework tiêu biểu cho công nghệ phát triển Web bằng Java. Hầu hết các hệ thống này đều là mã nguồn mở, có thể tải về để sử dụng. Tác giả N. Ford đã trình bày một cách súc tích và hệ thống hóa về cách khai thác các framework này để xây dựng ứng dụng Web [10].

Bảng 1 .Các framework cho các ứng dụng web [10]

Framework	Địa chỉ	Mô tả
Struts	http://jakarta.apache.org/struts	Một framework mã nguồn mở, được thiết kế chủ yếu cho việc xây dựng các ứng dụng theo Model 2
Velocity	http://jakarta.apache.org/velocity	Là một engine mẫu dựa trên ngôn ngữ Java. Framework này cho phép sử dụng ngôn ngữ template để tham chiếu đến các đối tượng được định nghĩa trong mã nguồn.
Tapestry	http://jakarta.apache.org/tapestry	Là một framework có thể chọn để thay thế cho JavaServer Pages. Thực hiện thay sự phát sinh các script và các đoạn mã nguồn bằng một mô hình đối tượng thành phần hoàn chỉnh.
WebWork	http://sourceforge.net/projects/opensymphony	Đây là một dự án chung được xây dựng theo qui trình mã nguồn mở, nhằm cung cấp công cụ và một framework cho việc xây dựng website trong thời gian ngắn nhất để dễ dàng hiểu và bảo trì.
Turbine	http://jakarta.apache.org/turbine	Là một framework rất lớn, mã nguồn mở, framework dựa trên dịch vụ cho việc xây dựng các ứng dụng web lớn, như site về e-commerce.

4.MỘT TRƯỜNG HỢP NGHIÊN CỨU ĐIỂN HÌNH

Trong phần này chúng tôi trình bày tóm tắt một ví dụ điển hình về xây dựng ứng dụng *WChat* để trao đổi thông điệp trên Web (*Web chat*). Mục đích chính của ví dụ này là để minh họa về nghiên cứu vận dụng mẫu thiết kế hướng đối tượng trong quá trình thiết kế và cài đặt ứng dụng Web. Vì vậy chúng tôi tập trung vào nghiên cứu kỹ lưỡng việc thiết kế kiến trúc cho hệ thống này, nhằm thực hiện từng bước mục tiêu xây dựng phương pháp luận triển khai các ứng dụng Web. Sau khi cài đặt ứng dụng thử nghiệm, các bản thiết kế sẽ được xem xét trở lại để phân tích, đánh giá, chỉnh sửa để có được bản thiết kế tốt. Cách làm này sẽ được thực hiện cho nhiều hệ thống khác. Tư tưởng chính ở đây là mượn việc triển khai các hệ thống này để nghiên cứu thực tiễn về qui trình thiết kế hướng đối tượng cho các ứng dụng Web dựa trên cơ sở các mẫu thiết kế.



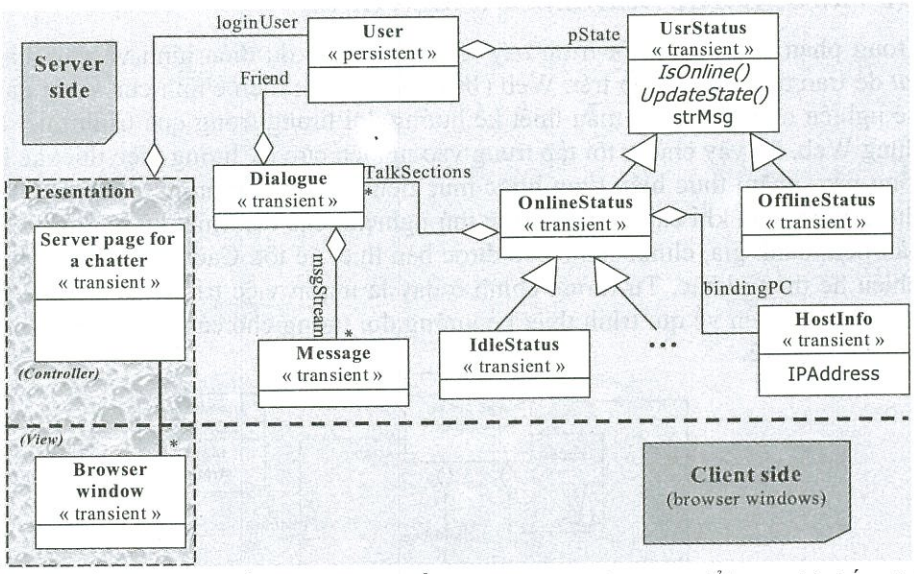
Hình 6. Mô hình dữ liệu quản lý "chat user"

Hình 6 trình bày sơ đồ lớp UML để mô hình hóa hệ thống quản lý người sử dụng hệ thống *WCHAT*. Mô hình dữ liệu này hoàn toàn độc lập với phần trình bày ra bên ngoài cho người dùng, là mô hình lưu trữ bên trong và được cài đặt tại máy chủ Web. Đây là thiết kế khởi đầu cho ứng dụng thử nghiệm: thiết kế này sẽ được phân tích và chỉnh sửa trong quá trình triển

khai hệ thống. Thiết kế chi tiết hơn về bộ phận quản lý thông điệp và trạng thái người dùng được trình bày trong sơ đồ lớp trong hình 7.

Trong các sơ đồ lớp này (hình 6 và hình 7): các đối tượng của lớp được hiểu mặc nhiên là sẽ lưu trên một cơ sở dữ liệu nào đó (tức là *persistent objects*), đối với các lớp có chú giải mở rộng *<<transient>>* thì đối tượng của chúng là các đối tượng tạm trong bộ nhớ (tức là *transient objects*) sẽ bị hủy khi người sử dụng thoát khỏi hệ thống *WCHAT*. Một số điểm lưu ý như sau về việc áp dụng mẫu thiết kế Web trong ví dụ này:

- Ý tưởng của mẫu *MVC* được sử dụng để tách rời phần giao tiếp với người dùng (tầng *presentation* bao gồm *View* và *Controller*) khỏi tầng biểu diễn bên trong của các đối tượng (*Model*) được thể hiện trong sơ đồ lớp ở hình 7 ;
- Mẫu *State* của Gamma [12] được sử dụng để quản lý trạng thái của người sử dụng hệ thống WCHAT: lớp *UsrStatus* được phân cấp theo quan hệ kế thừa tương ứng với những trạng thái khác nhau (*online, offline, Idle...*) của người sử dụng ;
- Dạng biến thể của mẫu *Composite* [12] được sử dụng trong thiết kế này: khi người dùng (đối tượng của lớp *User*) ở trạng thái online (lớp *OnlineStatus*) thì có thể có nhiều cuộc nói chuyện bằng thông điệp (*TalkSections* là một tập hợp các *Dialogue*), mỗi cuộc nói chuyện lại liên quan đến người bạn đang nói chuyện (*Friend*): người bạn cũng chính là một thể hiện của lớp *User*. Như vậy các thể hiện của lớp *User* chính là dạng đối tượng phức hợp, được thiết kế không hoàn toàn giống như mẫu *Composite - GoF*.



Hình 7. Sơ đồ lớp cho hệ thống quản lý và hiển thị thông điệp được trao đổi trong hệ thống "chat"

Trong thiết kế và cài đặt chi tiết, chúng tôi cần phải ánh xạ các lớp đối tượng này vào một cơ sở dữ liệu cụ thể và ngôn ngữ script hay các trang server của công nghệ phát triển ứng dụng Web cụ thể. Tùy trường hợp lựa chọn hệ quản trị cơ sở dữ liệu mà có thể cần phải sử dụng một số kỹ thuật hay mẫu chuyển đổi. Ví dụ nếu chọn *MySQL* thì phải giải quyết việc ánh xạ các lớp đối tượng vào cơ sở dữ liệu quan hệ. Đối với công nghệ Web, nếu chọn *PHP 4* thì phải giải quyết thêm các vấn đề liên quan đến thừa kế và đa hình; còn nếu chọn *PHP 5* thì thuận tiện hơn bởi vì *PHP 5* là ngôn ngữ hướng đối tượng hỗ trợ sẵn khả năng lập trình hướng đối tượng.

5.KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Do đặc điểm công nghệ, việc xây dựng phần mềm Web cũng đối diện với nhiều vấn đề nảy sinh khác nhau. Trong phạm vi bài báo này, chúng tôi giới hạn trong các vấn đề xoay quanh những nỗ lực bảo đảm các nguyên lý công nghệ phần mềm trong quá trình phát triển phần mềm Web. Trong thực tế còn nhiều vấn đề quan trọng và thách thức khác kể cả về mặt công nghệ và về mặt nghiên cứu phương pháp luận. Chẳng hạn như vấn đề bảo mật của các ứng dụng Web, việc phân tích mã nguồn script để dò tìm các lỗ hổng bảo mật, về nâng cao chất lượng giao diện người dùng ở các máy khách (chẳng hạn như công nghệ đang phát triển *AJAX* [5,13] là nỗ lực quan trọng cho khía cạnh này), giải quyết các vấn đề liên quan đến cơ sở dữ liệu...

Để phát triển hướng nghiên cứu này, chúng tôi định hướng vào những công việc như sau:

- Nghiên cứu tiếp các mẫu thiết kế giải quyết các vấn đề khác nhau trong phát triển ứng dụng Web, hệ thống hóa và phân loại để định hướng áp dụng ;
- Áp dụng một cách hệ thống những mẫu thiết kế vào việc triển khai thử nghiệm các hệ thống phần mềm Web để rút ra những nguyên tắc phương pháp luận cần thiết cho việc vận dụng các mẫu này ;
- Về việc ứng dụng mẫu thiết kế khi sử dụng các framework được xây dựng sẵn hay trong những công nghệ mới. Một số *framework* và công nghệ Web đã cài đặt hỗ trợ sẵn các mẫu thiết kế, vấn đề là đối với từng tình huống cụ thể thì vận dụng như thế nào. Chẳng hạn đối với công nghệ *AJAX*, thì mẫu thiết kế đóng vai trò như thế nào...
- Xây dựng các công cụ cần thiết nhằm tích hợp và hỗ trợ việc sử dụng các mẫu thiết kế trong quá trình phát triển phần mềm Web.

APPLYING OBJECT-ORIENTED DESIGN PATTERNS TO DEVELOP WEB-BASED APPLICATIONS

Tran Dan Thu, Huynh Thuy Bao Tran
University of Natural Sciences, VNU-HCM

ABSTRACT: *Web-based applications are more and more important in supporting a great deal of business processes. In recent years, researchers as well as industrial developers have invested much effort in contribution of development methods and infrastructural environment for web-based applications. Moreover, the advances in object techniques, especially the emergence of object-oriented design patterns, favorize the construction of web-based applications.In this paper, we present certain results in applying design patterns in the development process of web-based applications.*

Keywords: *Object-Oriented Technique, Design Pattern, Web Application, Distributed Application.*

TÀI LIỆU THAM KHẢO

[1]. D. Bonura, R. Culmone, E. Merelli, *Patterns for web applications,* ACM International Conference Proceeding Series, Vol. 27, p. 739 - 746, (2002).

[2]. F. Buschmann, *Pattern-oriented Software Architecture - A System of Patterns,* John Wiley & Sons, (1996).

[3]. J. Conallen, *Building Web Applications with UML,* Addison-Wesley, (2002).

[4]. J. W. Cooper, *The design patterns Java companion,* Addison-Wesley, (1998).

[5]. D. Crane, E. Pascarello, D. James, *Ajax in Action,* Manning Publications, (2005).

[6]. DONG T. B. Thuy and TRAN D. Thu, *User Interface Design by Applying Object – Oriented Design Patterns,* Addendum Contributions to the 4th IEEE International Conference on Computer Sciences - Research, Innovation & Vision for the Future, February 12-16, Hochiminh City, Vietnam (RIVF 2006)

[7]. P. Eeles, K. Houston, W. Kozaczynski, *Building J2EE Applications with the Rational Unified Process,* Addison-Wesley, (2002).

[8]. M. Ewiss, 'Patterns for Web Applications', *Pattern Languages of Programs conference 2003 (PLoP 2003).*

[9]. E. B. Fernandez et al., *Remote Authenticator /Authorizer*, Pattern Languages of Programs conference 2003 (PLoP 2003).

[10]. N. Ford, *Art of Java Web Development,* Manning Publications, (2004).

[11]. M. Fowler, *Patterns of Enterprise Application Architecture,* Addison-Wesley, Pearson Education, (2003).

[12]. E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software,* Addison-Wesley Longman, (1995).

[13]. J. J. Garrett, *'Ajax: A New Approach to Web Applications'*, Adaptive Path 2005, available at http://www.adaptivepath.com/publications/essays/archives/000385.php.

[14]. *A.*Gutmans, S. S. Bakken, D. Rethans, *PHP 5 Power Programming,* Prentice Hall, (2004).

[15]. V. Hays, M. Loutrel, and E. B. Fernandez, *The Object Filter and Access Control Framework,* Pattern Languages of Programs conference 2000 (PLoP 2000).

[16]. D. Hong and K. Snow, 'Web Design Pattern Library Feature Requirements', *Center for Document Engineering Technical Report (CDE2006-TR08),* April 24, (2006).

[17]. Jacobson, G. Booch, J. Rumbaugh, *The Unified Software Development Process,* Addison Wesley Longman, Inc, (1999).

[18]. P. Kimmel, *Advanced C# Programming,* McGraw-Hill, (2002).

[19]. *A.* Leff, J. T. Rayfield, *Web-application development using the Model/View/Controller design pattern*, Proc. of the 5th IEEE International Conf. on Enterprise Distributed Object Computing Conference (EDOC 2001), 4-7 Sept., (2001).

[20]. M. Marks and K. Snow, *Methodology for Developing Web Design Patterns,* Center for Document Engineering Technical Report (CDE2006-TR06), April 24, (2006).

[21]. F. Marinescu, *EJB Design Patterns,* John Wiley & Son, (2002).

[22]. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual,* Addison Wesley Longman, Inc, (1999).

[23]. M. Stiefel, R. J. Oberg, *Application Development Using C# and .NET,* Prentice Hall PTR, (2001).

[24]. Trần Đ. Thư và Huỳnh T. B. Trân, *Mẫu thiết kế hướng đối tượng cho các ứng dụng phân bố,* Báo cáo trong Hội thảo Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông, Đà lạt, (06/2006)

[25]. Tran D. Thu, Huynh T.B. Tran, *A composite design pattern for object frameworks,* 2nd IEEE International Workshop on Software Architectures and Component Technologies, July 23-27, (2007), Beijing (SACT 07, in conjunction with the IEEE COMPSAC 2007)

[26]. W3C, *Extensible Markup Language (XML),* http://www.w3.org/XML.