International Journal of Recent Research and Review, Vol. V, March 2013

ISSN 2277 – 8322

A component-centric UML based approach for modeling the architecture of web applications.

Mukesh Kataria[1]
[1]*Affiliated to Rajasthan Technical University, Research Scholar (M. Tech.), Computer Science Department, Poornima College of Engineering, Jaipur, Rajasthan, INDIA,*
mukeshkataria31@gmail.com

Raj Yadav[2]
[2]*Software Engineer, Computer Science Department, Kautilya Institute of Technology and Engineering, Jaipur, Rajasthan, INDIA,*
yadavrajc@gmail.com

Ajay Khunteta[3]
[3]*Associate Professor, Computer Science Department, Poornima College of Engineering, Jaipur, Rajasthan, INDIA,*
ajay_khunteta@rediffmail.com

Abstract- **Web applications use components developed in various technologies through an abstraction space richer than that of the object oriented paradigm. The architecture of web applications can be represented by showing specific web components, their compositions, navigations and inter-component relationships. In this paper, we propose a component-centric UML based approach for modeling the architecture of web applications. Our approach is based on a classification of components and inter-component relationships that typically occur in web applications. We use UML extension mechanisms to represent specific web components and their relationships.**

We categorize the web application components and their inter-component relationships. Based on that categorization, a stereotype suite for UML is developed. We analyze various web technologies including HTML, ASP, JSP, PHP, Servlet and JavaBean to develop the consolidate stereotypes.

Keywords- UML, ADL, HTML, ASP, JSP, PHP, ACME, OMG.

1. INTRODUCTION

A typical web application uses components developed in various technologies through an abstraction space richer than that of the object oriented paradigm. Web applications have concepts like session and cookies, which are specific to the domain of web applications. The aim of this project is to develop a UML based modeling language for describing the architecture of web applications. This language should be able to specify, design and document the functionality and behavior of a web application that could be developed using any of the available web technologies. Our work includes the use of UML extension mechanisms to describe web specific components and their relationships with other web components and traditional middle-tier elements. Our final goal is to design various diagrams for modeling static and dynamic behavior of the web applications precisely.

As the size and complexity of software system increase, the design and specification of overall system become more significant issue to develop and maintain them. We have various Architecture Description Language (ADL) including Adage, Aesop, Darwin, Wright and ACME, to formally specify the architecture of software systems. But as observed in, ADLs have not come into extensive use in industries, because we have to learn a distinct notation which is specific to architecture, and ADLs do not address all stages of development process. On the other hand, UML is a standard modeling language for visualizing, specifying, constructing and documenting the architecture of software systems. UML diagrams are graphical representation of system elements i.e. class, object, component etc and their inter-relations including association, aggregation and inheritance. In forward engineering, UML helps us in developing a new system from problem statements and provide efficient communication among stakeholders including developers, architects, project managers, maintainers and testers. In reverse engineering, it helps in precisely representing inter-relationships between system elements, which is very important to maintain and update already build systems.

2. WEB APPLICATIONS AND WEB COMPONENTS

A web application is a distributed application that is developed using various web technologies and whose functionality is accessed via web over a network such as the Internet or an intranet. We can update and maintain them without distributing and installing software on potentially thousands of client computers. Due to this ability web applications are very popular than the desktop applications. Web applications are not limited to one typo of application. They can range from simple static web pages to sophisticated applications. Different categories of web applications are grouped together according to their data and control complexity.

- Brochure Web Applications: This is the first generation of web applications. They are composed of static web pages and tend not to have much programming logic in them. When developing them the focus is on content development and the layout of graphics and text. Examples include personal web pages which contains their resume and personal information, or web pages about company products.

- Service Oriented Applications: These applications offer a service to web users and contain the programming logic needed to implement the service. The layout of the data is often a secondary concern. Examples include web mail services and online word-processing systems.

- Data Intensive Applications: These applications provide an interface to browse and query large amount of data. The main emphasis in these applications is on the data, with minimal amount of programming logic involved. An example of this application type is online library catalogs.

- Information System Applications: These applications combine the service oriented applications and the data intensive applications. Developers of information system applications are concerned with the data flow for browsing and retrieving data, and control flow for the different phases involved in the manipulation of the data. Electronically book stores or online banking are example of this application type.

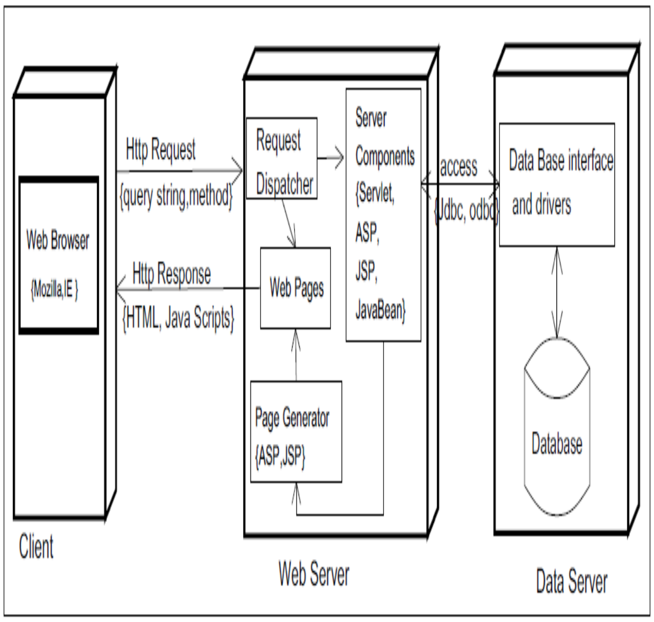


Fig. 1. Typical Architecture of a Web Application

3. UML PROFILE FOR WEB APPLICATIONS

UML is a standard and industry accepted graphical modeling language to model the software systems. However, since it is a general purpose modeling language, it lacks elements to model and represent concrete concepts of specific domains. This lack of expressiveness also exists for the domain of web applications. As a solution, OMG (Object Management Group) has created a mechanism for extending the syntax and semantic of UML to express more specific concepts of certain application domains. In this chapter, we first briefly explain the UML extension mechanisms used for creating UML profiles. Then we explain the construction of UML profile for web application domain.

UML extensibility mechanisms permit us to extend the language in controlled ways. The controlled ways means the changes are easily understandable by the UML users. These mechanisms include stereotypes, tagged values, and constraints. We extend the UML by adding new model elements, creating new properties, and specifying new semantics. Stereotypes: The stereotypes extend the vocabulary of the UML, allowing us to create new kinds of model elements that are derived from existing modeling elements but are specific to our domain i.e. web applications. We represent a stereotype by placing the name of stereotype between << >> and above the name of an existing UML element. For example, when modeling a web application, we might want to represent client presentation component. We can use stereotypes << ClientPresentation >> by extending the UML standard model element i.e. component. As an option, the stereotyped element may be rendered by using a new icon associated with that stereotype.

4. CONSTRUCTION OF THE PROPOSED UML PROFILE

For the creation of UML profile, we use the aforementioned UML extension mechanisms. We follow the given below procedure to develop the UML profile for web applications.

4.1 Building the Meta-Model for Web Applications

A web application has many web elements including client static page, form, client dynamic page, presentation component, service component, hyperlink and link. These all elements represent an abstraction drawn from the web application development. The definition of these elements defines the meta model for the web application domain.

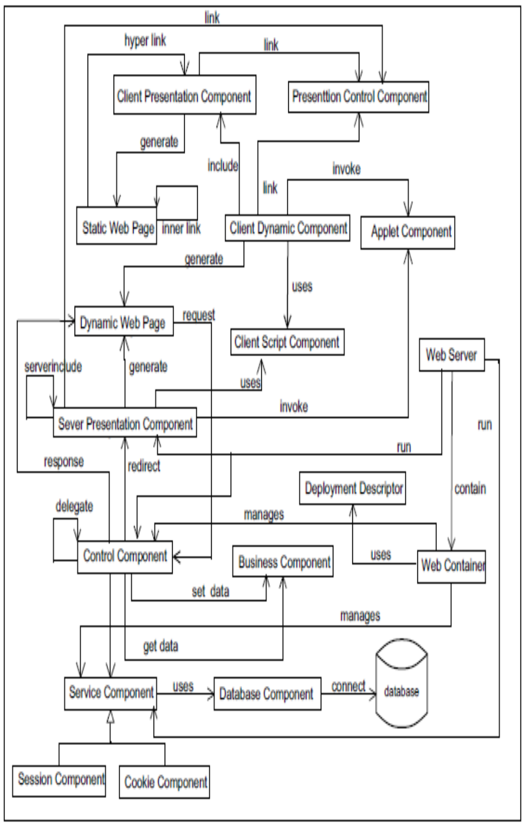


Fig. 2. A Metamodel for Web Application Domain

4.2 Define Stereotype for Web Elements

In this step, we define a stereotype for each metamodel element (web Page, form) that we find in the previous step and want to include in the profile. We select some element and create the corresponding stereotype for each one of them.

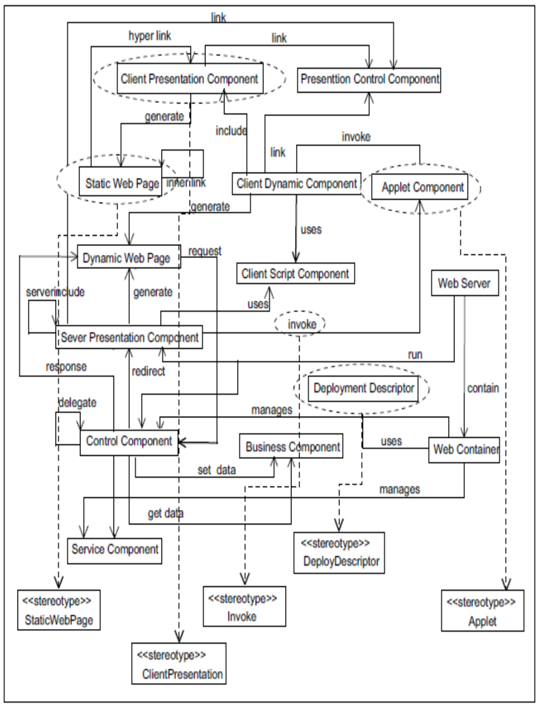
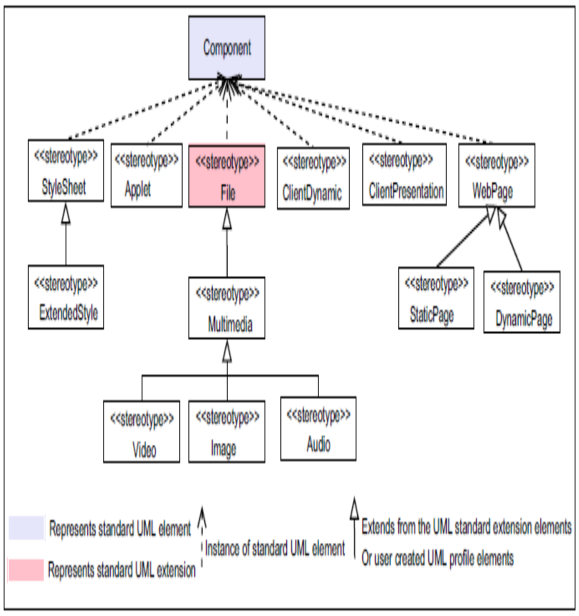


Fig. 3 Stereotypes for the Metamodel Elements

4.3 Extension of UML Modeling Elements

We identify the UML elements that will be extended with each stereotype. We identify various UML elements including class, component, package, association for creating UML profiles. We create various stereotypes by extending UML modeling element component, for representing client component precisely. The << ClientDynamic >> stereotype specify a client component that generates an interactive web page. It may uses or invoke other client components.

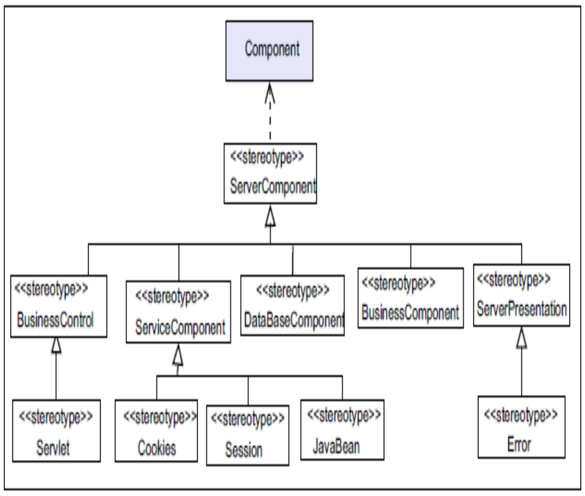
The << ClientDynamic >> stereotype specify a client component that generates an interactive web page. It may uses or invoke other client components. We define the various stereotypes including stylesheet, client script and multimedia by extending the UML standard extension element << file >>. We define <<StyleSheet >> stereotype to represent a presentation control component and << Applet >> stereotypes to specify an executable component that may runs independently or invoke by other components to provide dynamic behavior at client side.



Component
<<stereotype>> StyleSheet
<<stereotype>> Applet
<<stereotype>> File
<<stereotype>> ClientDynamic
<<stereotype>> ClientPresentation
<<stereotype>> WebPage
<<stereotype>> ExtendedStyle
<<stereotype>> Multimedia
<<stereotype>> StaticPage
<<stereotype>> DynamicPage
<<stereotype>> Video
<<stereotype>> Image
<<stereotype>> Audio
Represents standard UML element
Instance of standard UML element
Extends from the UML standard extension elements
Or user created UML profile elements
Represents standard UML extension

Fig. 4 UML Stereotypes for Client Components

Web application has various server components that are either process the user request or provide service to other components. They are managed by web container and are run on web server. Figure 4.6 shows stereotypes for representing the server components. The stereotype << ServerComponent >> specifies a component that is executed by web server, when it receives the request from client or other server component. We create a << Server Presentation >> stereotype to specify a server presentation component.



Component
<<stereotype>> ServerComponent
<<stereotype>> BusinessControl
<<stereotype>> ServiceComponent
<<stereotype>> DataBaseComponent
<<stereotype>> BusinessComponent
<<stereotype>> ServerPresentation
<<stereotype>> Servlet
<<stereotype>> Cookies
<<stereotype>> Session
<<stereotype>> JavaBean
<<stereotype>> Error

Fig. 5 UML Stereotypes for Server Components

The << ActionLink >> stereotype represents a association relationship that specify the source component request is processed by target component. The << ResponseLink >> stereotype specifies the source component generate the target as response to the user request. To specify the include directive that is used to physically include the contents of target into source file, we create a << ServerInclude >> stereotype by extending an association element. The << JspInclude >> stereotype specifies the inclusion of either a static or dynamic resource in a JSP page. If the resource is static, its content is included in the calling JSP page. If the resource is dynamic then it sends a request to another page and includes the response in the response from the calling page.

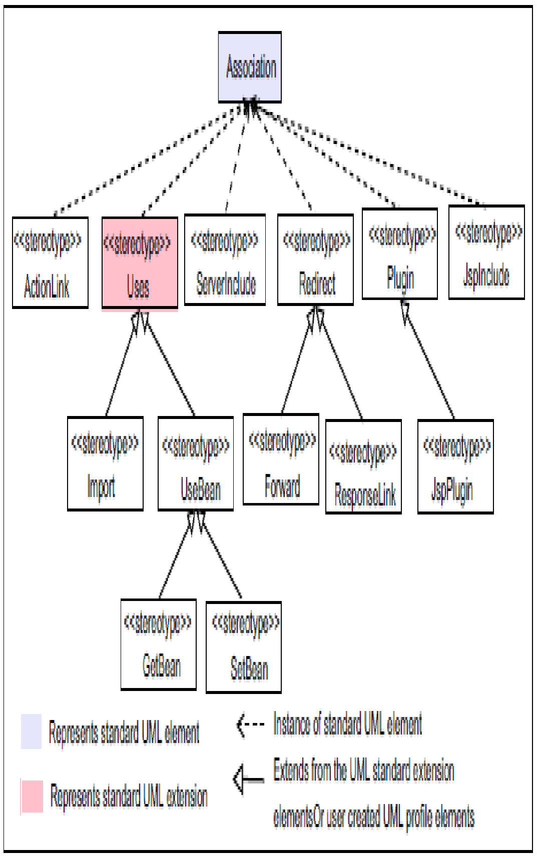

Association
<<stereotype>> ActionLink
<<stereotype>> Uses
<<stereotype>> ServerInclude
<<stereotype>> Redirect
<<stereotype>> Plugin
<<stereotype>> JspInclude
<<stereotype>> Import
<<stereotype>> UseBean
<<stereotype>> Forward
<<stereotype>> ResponseLink
<<stereotype>> JspPlugin
<<stereotype>> GetBean
<<stereotype>> SetBean
Represents standard UML element
Represents standard UML extension
Instance of standard UML element
Extends from the UML standard extension elements Or user created UML profile elements

Fig. 6. UML Stereotypes for Server Component Relations

5. CONCEPTUAL COMPONENT DIAGRAM

A conceptual component is referred to as the program module that follows some specific specification and has interaction with other components. A conceptual component represents the component properties, interfaces and events that is used for interaction with other components. Properties are not like traditional member variables of a class, instead these are high level attributes of a component that can be accessed by other web components. Interfaces are the subset of accessible operations which represent the interaction points of a component.

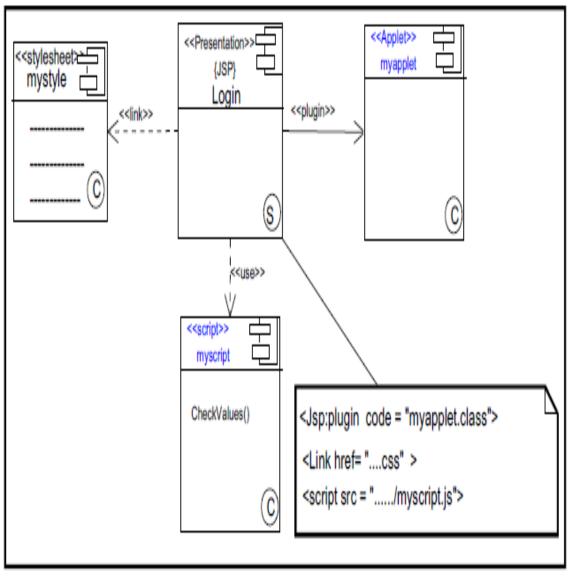


Fig. 7. A Conceptual Component Diagram

It shows presentation component invokes an applet to provide dynamic behavior at client side and uses a style sheet to control the layout of generated web page. For the user data validation at client side, presentation component also uses a client script component.

6. CONCLUSION

In this paper, we proposed a component-centric UML based approach to model the architecture of web applications. We found many reasons to use and extend the UML for modeling web applications. We analyzed various web technologies including HTML, ASP, JSP, PHP, Servlet and JavaBean to develop a consolidate UML profile for web applications. Using this profile, one can represent the architecture of web applications which are developed by use of available technologies. The paper also explored various kinds of diagrams through which typical web application component architecture can be represented. It was found that many existing diagrams can be used by applying proposed UML profiles to them.

7. REFERENCES

[1] Ahmed and Richard, Architecture recovery of web applications, Proceedings of International Conference on Software Engineering (2002).

[2] Paolini Baresi, Garzotto, Extending UML for modeling web applications, Proceedings of the 34th Annual Hawaii International Conference on System Sciences(HICSS-34)-(2008).

[3] Longo Bochicchio, Conceptual modeling of data intensive and information intensive web applications, Proceedings 10th International Multimedia Modelling Conference (2007)

[4] Jacobson, Rumbaugh, The unified modeling language user guide, Addision-Wesley, 2006.

[5] Matera Ceri, Fraternali, Conceptual modeling of data-intensive web applications, IEEE Internet Computing, vol.06 (2008), 20-30

[6] Jun Zhou Chang-ai Sun, Rearchjbs: a tool for automated software architecture recovery of javabeans-based applications, Proceedings of the Software Engineering Conference (2005), 270- 280.

[7] Kazman Clements, Bass, Software architecture in practice, Addision-Wesley (2004).

[8] Jim Conallen, Modeling web application architectures with uml, Communications of the ACM 42 (1999), no. 10, 63-70.

[9] Mandel Hennicker, Koch, Extending uml to model navigation and presentation in web applications, Workshop on Modelling Web Applications in UML (2000).

[10] Ping Chen Jing, Jian Chen, Modeling web application architecture with uml, Proceedings of the 36th International Conference on Technology of Object-Oriented Languages and Systems pp.265-274 (2006).

[11] Nora Koch and Andreas Kraus, Towards a common metamodel for the development of web applications, Proceedings 3rd International Conference on Web Engineering (ICWE 2003) (2003), 497-506.

[12] N.Koch R.Hennicker, Modeling the user interface of web applications with UML, Workshop on Practical UML-Based Rigorous Development Methods (2006), 158-172.

[13] Joen Roh, Kim, Architecture modeling language based on UML 2.0, Proceedings of the 11th Asia-Paci_c Software Engineering Conference (APSEC04) (2008).