A Component-Based Method for Developing Web Applications

Jonas A. Montilva C.[*] and Judith Barrios A.[**]

Abstract

We describe, in this paper, a component-based software engineering method for helping development teams to plan, organize, control, and develop web applications. The method is described in terms of three methodological elements: a product model that captures the architectural characteristics of web applications, a team model that describes the different roles to be played by the members of a team during the development of web applications, and a process model that integrates the managerial and technical activities that are required to develop componentized web applications of high quality. The main features of the model are its component-based approach that helps reduce costs and development time; its ability to integrate managerial and development processes into a unique process model; and its emphasis on business modelling as a way of gaining a better understanding of the application domain objectives, functions and requirements.

Keywords: Component-Based Software Engineering, Web Engineering, Web Information Systems

1. Introduction

Developing web applications is a very complex process that demands highly skilful personnel. Its complexity is due to the fact that it involves many different information technologies including WWW, human factors, information systems, business modelling, programming languages, distributed systems, and databases. A web application is defined as a software application that uses a web-based user interface to provide a set of information services to its users. These services, called the business logic of the application, comprise the input, storage, update, access, retrieval, and manipulation of data objects that are stored in one or more servers. Web information systems, e-commerce systems, e-business portals, e-government systems, and instructional web sites are some typical examples of web applications.

The scope, purpose and complexity of web applications vary widely from small scale informational services to large-scale enterprise applications that support several business processes. Ginige and Murugesan (2001a) groups web applications into the following categories: informational, interactive, transactional, workflow-based, collaborative work environments, online communities, marketplaces, and web portals.

Although a great number of web applications are actually in use everywhere, the way most of them were developed raises many concerns. A survey conducted in Nov. 2000 by the Cutter Consortium evidences the following problems related to web-based development projects: (1) 84% of the delivery systems didn't meet business needs; (2) 79% of the projects reported schedule delays; (3) 63% exceeded the budget; (4) 53% of the delivery systems didn't provide the required functionality; and (5) 52% of the deliverables were of poor quality. Ginige and

[*] University of Los Andes, Faculty of Engineering, School of System Engineering, Computer Science Department, Merida, Venezuela. Email: jonas@ing.ula.ve,
[**] University of Los Andes, Faculty of Engineering, School of System Engineering, Computer Science Department, Merida, Venezuela Email:, ijudith@ing.ula.ve

Murugesan (2001a) highlight the main problems associated to web application development. Some of these problems are the following: (a) the developers pay little attention to development methodologies, quality assurance and project management; (b) many developers considers web development as an authoring process, rather than an application development process; (c) the complexity of web applications has grown significantly; and (d) quality attributes are not given the due consideration they deserve during the development process.

The need for better ways of developing web applications that take into account the nature and complexity of this type of applications has been recognized by many researchers (Ginige and Murugesan, 2001b; Deshpande and Hansen, 2001; Barry and Lang, 2001; Hadjerrouit, 2001; and Gaedke and Rehse, 2000). To address this need, Web engineering has emerged very recently as a discipline that deals with systematic, disciplined and quantifiable approaches to development, operation, and maintenance of Web-based systems and applications (Murugesan and Deshpande, 2002).

A set of methods, techniques, guidelines, and methodologies have emerged, during the last three years, to help development teams to produce better web applications. Some of the most well-documented ones have been proposed by Conallen (2000), Fraternali and Paolini (2000), Hadjerrouit (2001), Gaedke, et al (2000), and Allen (2001). The Conallen's method is an adaptation of the Rational Unified Process (Krutchen, 1998). It uses an object oriented approach to model web applications. The method described by Fraternali and Paolini (2000) makes more emphasis on the hypermedia nature of web applications. Hadjerrouit (2001) stresses the use of software engineering process in the development of web application, but recognizes the relevance of user interfaces in this type of applications. The methods proposed by Gaedke, et al (2000) and Allen (2001) emphasize the importance of reusability in the process of developing web applications. Both of them apply principles and process models borrowed from Component-Based Software Engineering.

We believe that a component-based approach to the development of web applications has several advantages over those approaches based on object-orientation and human factors. Firstly, the reuse of components has proven to reduce the cost and time to deliver a solution, since the application is not designed and nor written from scratch. Secondly, the component-based approach promotes a better quality of the product due to the reuse of already proven components. Finally, better design of the web application architectures can be obtained by reusing architectural patterns that promotes separation of concerns.

We propose, in this paper, a component-based method for developing web applications. The method integrates the UML Components process (Cheesman and Daniels, 2000) with the methodological framework provided by the IEEE 1074 standard for developing software life cycles (IEEE, 1995). The method uses a watch metaphor for integrating the management and development processes that a team requires to plan, organize, coordinate, control, and develop web applications using a component-based approach.

Section 2 of this paper delineates the methodological elements that comprise the method: the *product model* which describes the generic characteristics of web applications, the *team model* that suggests an appropriated way of organizing the development team, and the *process model* that describes the managerial and technical activities that are needed to develop web applications. Sections 3 – 5 present, in more detail, each of the three methodological elements of the method. Finally, Section 6 summarizes the concluding remarks of this research.

2. The Methodological Elements Of The Watch Method

The design of the *Watch* method was based on principles and concepts taken from Method Engineering (ME) and Software Engineering (SE). Odell (1996) defines ME as a systematic and coordinated approach to establishing work methods. An important ME principle establishes that a well-defined method should be described in terms of two components: a model of the product to be developed and a process model that explains how to develop the product (Brinkkemper, 1996). We extended the application of this principle by adding a third element: a team model, which describes the roles that the members of the development team play during the development process.

To describe our method and its application, we elaborated a two level abstraction hierarchy that explains the relationships between the three models mentioned above and the development of a particular web application (see Figure 1). The upper level is made of the three models that compose the method: a product model, a process model, and a team model. The *product model* describes the concepts and architectural patterns that are common to most web applications. The *process model* describes the activities that the development team should follow to develop web applications. The *team model* is a description of the roles played by the participants of the process of developing a web application.

To apply the method in a specific web application project, the team leader must instantiate the three models of the method (the upper layer). The product model helps to design the architecture of the web application. The process model is used to define the activities, techniques, tools, and notations that the team requires to develop the web application. The team model helps to organize the team and define the responsibilities of its members.

We describe in the next sections each of these three models.

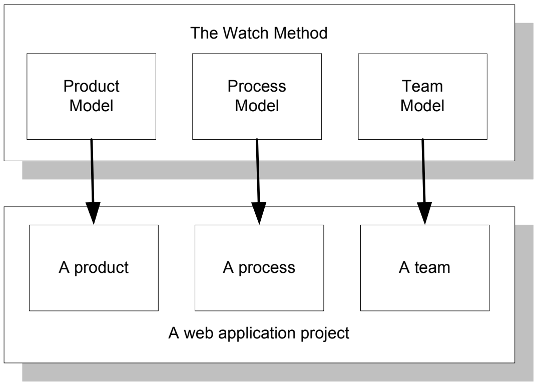

Figure 1. The structure and application of the Watch method

3. The Product Model

A product model is a generic description of the common elements that are found in the final products elaborated using a method. As described in this paper, the *Watch* method addresses the development of web applications.

A web application is a web-based software application that allows its users to execute business logic with a web browser (Conallen, 1999). The user interacts with the application through a web browser that allows him/her to input data and request the execution of functions or services from an application server. A common architectural style used for designing web applications is the *n-tier architecture.* The front tiers deals with the presentation aspects of the application, including the user interface and the user dialog management. The user interacts with the application through this tier. The middle tier is related with the business logic of the application. It is the engine of the web application. It processes the input data and the transactions or services requested by the users through the front tier. It executes the operations needed to accomplish the services, process the input data, send/retrieve data to/from the back tier, and organizes the output data to be sent to the front tier. The back tier is concerned with data management. It behaves as a data server that stores and retrieves the application data into one or more databases.

Based on this architectural style and the architectural patterns proposed by Conallen (2000) and Curphey (2002) for building web applications, we designed a product model for this kind of systems. This model is described in terms of a domain architecture whose structural view is shown in Figure 2. This view organises the components of a web application into three layers. Each layer includes one or more tiers as described next.

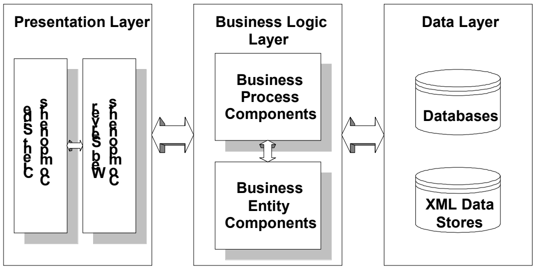

Figure 2. The product model of the method

The presentation layer is responsible for the interaction with the end users. It deals with the presentation of information to the users, the capture of inputs, and the management of the user's dialog. This layer is made of two tiers, one for each type of component: client-side and web server components. The client-side components are related with the HTML pages of the web application that are rendered by web browsers on client machines. They provide user interface functionality that is not achievable by HTML elements alone. They may be implemented using Java Applets, ActiveX controls, and Scriptlets. The web server components, on the other hand, are executed by web servers. They provide the functionality needed at the web server in order to:

- Prepare the HTML formatted pages that are required by web browsers, and
- Request, from the business logic tier, the services demanded by end users.

The web server components may be implemented as scripts that are interpreted by the web servers, or as compiled binary files that are executed by the web servers.

The business logic layer consists of two kinds of components: business process components and business entity components. The business process components handle the services or transactions that are requested by users through the user interface. They determine the operations of the business entity components that must be invoked and the order in which they must be executed. The business entity components are persistent. They represent the business entity types of the application domain whose state must be stored by the application.

The data layer is responsible for managing the data stores where the states of the business entities are stored. Databases and XML data stores, together with their engines, are the main architectural components of this layer.

Figure 3 illustrates the main architectural components of a web application and their relationship with the deployment infrastructure used to install and execute these components.

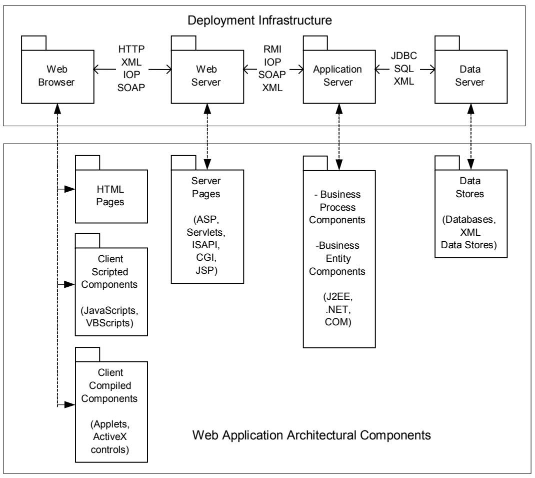
Deployment Infrastructure
Web Browser
HTTP XML IOP SOAP
Web Server
RMI IOP SOAP XML
Application Server
JDBC SQL XML
Data Server
HTML Pages
Client Scripted Components (JavaScripts, VBScripts)
Client Compiled Components (Applets, ActiveX controls)
Server Pages (ASP, Servlets, ISAPI, CGI, JSP)
- Business Process Components -Business Entity Components (J2EE, .NET, COM)
Data Stores (Databases, XML Data Stores)
Web Application Architectural Components

Figure 3. Main components of a web application

According to the product model outlined in this section, the development of a web application involves the specification, design, implementation and testing of the three groups of components:

- **The presentation group.** It is composed by HTML pages, client scripted or compiled components, and server page components. This group is responsible for the user interface aspects of the web application.
- **The business logic group**. It is made of business process components and business entity components. They are responsible for processing the logic of the application. The business process components implement the services that are requested by the users through the presentation components. Each of these services involves the execution of a workflow that access and/or modify the state of the business entity components.
- **The data group**. It includes the databases or data stores that are managed by their corresponding engines. The state of the business entity components are accessed and/or updated by this group of components.

It should be observed that these three groups of components involve the use of technologies that are quite different. Separating the web application components into these groups has, at least, two advantages. Firstly, the development of the components can be more easily assigned to members of the development team according to their expertise and technological skills. Secondly, the separation of concerns facilitates the decisions to be made during the design of the application.

4. The Team Model

An important aspect of any software development project is the organisation of the effort that will be required to develop the application. The definition of team roles are important because it helps the project leader to select the right people and assign them based on the skills needed to perform the different types of task that a project requires. We will refer to the group of people that participates in the web application development process as the *development team*.

A development team can be organised in many different ways depending on the complexity of the web application to be developed. Allen and Frost (2001) provide a catalog of team roles organized into three groups: application development roles, component development roles, and support roles. Based on this catalog and the *Watch* product model, we chose a set of team roles and adapted these roles to fit the development of componentised web applications. The resulting team configuration is made of the following roles and its associated responsibilities:

- **Project manager/team leader.** This role has associated the responsibility for planning, organizing, directing (leading), supervising, and controlling the application development process.
- **Adviser user/ambassador user.** Brings to the project the knowledge about the business or application domain. Provides the business requirements and acts as a bridge between the development team and the user community.
- **Application developer/Senior developer.** Interprets and models user requirements and uses technical skills to design and evaluate application architectures and their components.
- **Web developer.** Its responsibility is to specify, design and develop web-based user interfaces. It encompasses human factors skills (e.g., graphic design) and technical skills related to the web technology.
- **Business component developer.** Models and interprets business requirements and uses technical skills to design, code, integrate, test, and deploy business components.
- **Data component developer.** Models and interprets data requirements and uses technical skills to design, create, integrate, and evaluate databases and other types of data stores (e.g. XML data stores).

This configuration includes only core roles, i.e., roles that are more or less permanent throughout the project. Depending on the complexity and size of the project, more roles and responsibilities can be added or removed from this configuration.

5. The Process Model

We use a watch metaphor for designing the structure and dynamics of the *Watch* process model. Using this metaphor, the development of a web application can be seen as a watch or clock whose moving hands can be set and controlled by the project manager. The numbered dial represents the development phases, while its moving hands represent the order in which these phases are executed. The management processes are, therefore, at the center of the watch. They control the progress of the phases and decide when to advance to the next phase or go back to a previous phase to review or correct a deliverable or product.

The structure of our process model is based on the IEEE 1074 Standard for developing software life cycles processes (IEEE, 1995) and the UML Component process for developing component-based software (Chessman and Daniels, 2000). We chose and adapted those process groups of the IEEE 1074 standard and the UML Component process that are most appropriate to the development of small and medium size web applications. These processes were then reorganized into two groups: management and development processes, as shown in Table 1.

Table 1. The process structure of the method

Management Process Group	Development Process Group
Project Management	Business Modelling
Software Quality Management	Requirements Definition
Software Configuration Management	Architectural Design
Verification & Validation	Component Specification
Risk Management	Component Provisioning
Training	Component Assembly
	Application Testing
	Application Delivery

The management process group encloses those activities that are concerned with the processes of managing the project, assuring the quality of the application, managing changes and risks, and training the team. The development process group, on the other hand, encompasses the technical activities that are needed to model the application domain, elicit and specify requirements, and design, code, test, and deliver the application. The development process group is adapted from the UML Components process.

Figure 4 shows the dynamics of the *Watch* process model expressed in terms of its phases, their execution order and the relationship between the development processes and the management processes.

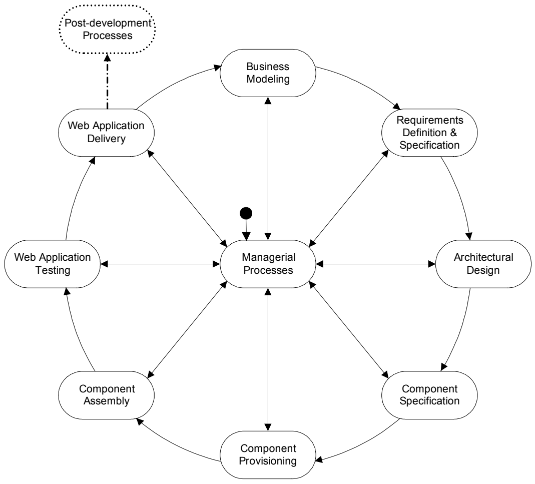

Figure 4. The Watch process model

The development of a web application starts at the center of the model, i.e., at the management process. Project initiation, project planning, and team organisation are the first project management activities to be performed. Business modelling is the first development activity to be executed after the project has been planned and organised. The other development activities are then performed in sequential order under the control of the verification and validation process.

The model is highly iterative. Once the project is planned and the team is organized, the development processes are initiated under the control of the managerial processes in a clockwise order, as indicated by the dial of the model. The verification & validation process determines, through a product review, if the progress of the project needs go back to a previous phase in order to correct detected failures, errors or deficiencies in the deliverables or products of the project.

The management processes and each of the development phases of the model are described in the next two sections.

5.1 The management processes

Management is a critical success factor in software development. The quality of the product, the delivery of the right product on time and under budget, and the proper control of the system configuration can only be ensured by the correct application of managerial processes.

In the *Watch* process model, the managerial processes group is similar in functionality to the engine of a watch. It directs and controls the execution of the development processes. That is the reason why we envisaged the managerial processes at the center of the watch; they control the progress of the phases and decide when to advance to the next phase or go back to a previous phase to review or correct a product.

Table 3. The Managerial Activities and Deliverables

Processes	Activities	Deliverables
Project Management	Project Initiation Project Planning Team Organisation Team Direction Team Staffing Project Control	Project Plan Team Structure Progress Reports
S/W Quality Management	S/W Quality Planning Quality Assurance	S/W Quality Plan Quality Model
S/W Configuration Management (CM)	CM Planning Configuration Control	CM Plan Change Procedures
Verification & Validation (V&V)	V & V Planning Products Review	V&V Plan (testing plans) Review Reports
Risk Management	Risk Identification Risk Analysis Risk Prioritization Risk Management Planning Risk Resolution	Risk Checklist Risk Mgmt Plan
Training & Documentation	Training Planning Team Training Training Guides Production	Training Plan Document Plan Training Reports User's Training Manuals

We identified, as crucial to the management of a web application project, six managerial processes: Project Management, Software Quality Management, Software Configuration Management, and Verification & Validation, Training and Risk Management. According to the IEEE 1074 standard, the two last mentioned processes have an integral nature. We included them into the management process group because they are under the control of the project manager. The main managerial activities and their deliverables are itemized in Table 3. The description of these activities is omitted from this paper due to space reasons.

5.2 The development processes

The *Watch* process model comprises eight development phases. These phases are executed sequentially in a clockwise order starting by the Business Modelling Phase and ending at the Web Application Delivery Phase. From any phase, it is always possible to return to a previous one in order to:

– Add or modify a deliverable, such as the business model, the requirement specification, the application architecture or a component;
– Repair an error detected in a deliverable; or
– Review and elaborate a task or activity of a previous phase.

The Verification & Validation Process of the managerial group determines when the project has to move in a counterclockwise direction.

The Business Modelling Phase

The first development phase to be executed is called *Business Modelling*. This activity identifies and describes in detail the business domain to be served by the web application. The purpose of this phase is to allow the development team to get a good understanding of the business domain before initiating the requirements definition phase. The deliverable of this phase is the *Business Model* which captures the objectives, business processes, business objects, actors, business rules, events, and job structure of the business domain.

This phase starts by defining the scope and objectives of the business domain. The business processes and actors are identified and modeled next. A business process is a set of business activities that, when executed correctly, contributes to reach an objective. Business processes are executed by a set of actors (e.g., managers, employees or customers) with the support of the web application. Business objects and rules are modeled next. Business objects are the types of entities that are related to the domain (e.g. clients, accounts, resources and products) and whose state are accessed or modified by business processes. Business rules are precise statements that describe, constrain and control the structure, operations, and strategies of a business (ILOG, 2001). Events are then identified. An event triggers the execution of one or more business processes. They capture the dynamics of a business. Finally, the job structure of the business domain must be represented. A job structure describes the organisation of roles, responsibilities and authority lines of the business.

The business model describes each of these business elements and the structural and dynamic relationships between them. Eriksson and Penker (2000) describe an extension to UML that can be used to build the structure of the business. The dynamics aspects of the business can be modeled using Petri Nets extensions for workflows, such as the Proclets notation described by Van Der Aalst, et al (2000).

The Requirement Definition & Specification Phase

The purpose of the *Requirements Definition & Specification phase* is to discover, define and specify the functional and non-functional requirements to be satisfied by the web application. The main deliverables of this phase are the Requirements Definition Document (RDD) and the Requirements Specification Document (RSD). The RDD is written in plain English with the help of the users. The adviser and ambassador users plays a major role in the elaboration of this document which is written for the users applying the terminology or "business language" of the application domain. RDD documents in functional and non-functional requirements as seen by users. Scenarios and use cases can be used in this phase to describe and specify the functional requirements.

The RSD expresses the user's requirements in a formal or technical way that can be understood by the developers without any ambiguity. It describes the user's requirements using a formal, semiformal, or graphical notation or language, such as the Unified Modelling Language (Booch, et al, 1999). The structure and behavior of the web application is specified using three models: functional, structural and dynamics. The *functional model* of the application can be built using the UML use case notation, which helps describe graphically the functionality to be provided by the application. The *structural model* may be built using UML class diagrams. This model captures the classes of business entities to be managed by the system, their relationships, their attributes, and theirs operations. The *dynamics model* represents the behavior of the application. UML sequence diagrams can be applied here to describe how each function specified by the functional model must be performed by the application. Workflow techniques can also be very helpful in documenting the application dynamics. Petri Net based techniques such as PROCLETS (Van der Aalst, et al, 2000) and WF nets, UML swimlanes and activity diagrams (Booch, et al., 1999), can be applied to describe the interaction between the business processes and the application.

The Application Design Phase

The *Application Design* phase translates the requirements specification into a solution, i.e., a design specification of the web application. The deliverable of this phase is the Application Design Document (ADD) which describes and specifies the web application architecture, the user interface and the database(s) to be used by the application. This phase is divided into the following fours steps:

- **User Interface Design.** The purpose of this step is to define the structural, aesthetics and visual characteristics of the web application user's interface. This interface is made of a set of interrelated web pages. Each web page is composed by a set of multimedia items (e.g., text, graphics, still images, animation, and video), forms, dialog boxes, menus, buttons and navigation bars that allows the user to navigate and operate the web application. The web developers design first the user interface structure based on the use cases described by the functional model. The aesthetics and visual properties of the interface are then decided with the collaboration of the adviser users. A prototype of the user interface is usually made, in this step, to visualize these properties and allow the users to validate and incorporate new requirements. Web design principles, such as those described by Lynch and S. Horton (1999), must also be used to ensure the usability requirements imposed to the application.
- **Architectural Design.** In this step, the development team designs the web application architecture. This architecture is made of a set of software components, connectors, and constraints that are normally organized into layers or tiers. The product model, shown in Figure 2, provides an architectural pattern that can be refined, extended or elaborated using the requirements, in order to define the application architecture. Web application architecture patterns, such as those described by Conallen (2000), can also be used to design the application architecture. Once the global view of the architecture is defined using the patterns, the components that conform the three architectural layers (presentation, business logic, and data layers) must be identified and initially specified. Each of these layers is made of a set of components. Each component is, in turn, composed of a collection of interrelated classes and one or more interfaces that define the services offered by the component. Components are identified using the business model and the three models (functional, structural, and dynamics) contained in the RSD. Business entity components, for instance, are discovered using the structural model which is composed by a collection of interrelated classes that represent business entity types. Business process components, on the other hand, are identified using the functional and dynamics models.
- **Database Design.** The data component developer must define, in this step, the type, structure and content of the data store(s) to be used by the application. The conceptual design of the data components (databases or XML data stores) must be produced in this step. The structural model contained in the RSD document provides the information needed to design in detail the components of the data layer.

The Components Specification Phase

The purpose of this phase is to specify, in more detail, the components that comprise the application architecture and the interactions between these components. The component model and the deployment framework to be used for implementing the components must also be defined in this phase. A component model specifies the standards and conventions used to implement the components. CORBA, J2EE and .NET provide their own component models that describe how the components interact in a distribute environment. A deployment framework provides a standard compose-time and run-time environment for deploying the components and executing the application (Bachman, et al., 2000). The main deliverable of this phase is the Component Specification Document (CSD).

A component specification is made for each component that comprises the presentation and business logic layers of the application architecture. The data layer is, commonly, comprised by a collection of tables or relations that comply with the Relational Model. Its data components do not require to be specified as programmed components, because of the nature of its constituents. Instead, the database conceptual design produced in the previous phase must be transformed into an implementation design and a physical design based on the DBMS used at the data layer. The relationships or mappings between the business entity components in the business logic layer and the data components in the data layer must also be specified in this step.

Contracts are useful mechanisms for specifying the presentation and business logic components. Two types of contracts can be used to specify these components: usage contract and realization contract (Cheesman and Daniels, 2001). A usage contracts describe the relationship between the interface of a component instance and its clients (i.e., the other component instances that request services from the component instance through its interface). Usage contracts, on the other hand, establish the relationships between component specification and component implementation. Usage and realization contracts are consistent with those proposed by the Software Engineering Institute (Bachman, et al, 2000), called interaction contracts and component contracts, respectively.

Based on the categories of contracts defined by Cheesman and Daniels (2001), we divide this phase into the following steps:

- **Usage Contract Specification.** A component is seen as an integrated collection of classes that provides a given functionality through one or more well-defined interfaces. A usage contract specifies the interface of a component. An interface is a set of operations. Each operation defines a service (functionality or responsibility) that the component instances will perform when the clients request this service. The usage contract includes: (1) the list of operations that the interface provides, (2) a specification for each operation of the interface that describes the input and output parameters, and the constraints that apply to the operation, (3) a specification of the set of classes that implements the interface, and (4) the pre- and post conditions that specify the effect of each operation.
- **Realization Contract Specification**. A realization contract is concerned with the implementation of a component specification. A realization contract must be consistent with the component model. The component model and the deployment infrastructure must, therefore, be defined before specifying the realization contracts. A realization contract describes the interactions between the implementation of the component specified by the contract and the other components. The mapping of the components and the deployment infrastructure must also be specified by the contract. Conallen (1999) describes an UML extension that can be used in this phase to associate the classes of each component to the deployment infrastructure to be used for implementing the web application.

The Component Implementation Phase

The purpose of this phase is to implement the design specifications produced in the Application Design and the Components Specification phases. The main deliverable is a set of presentation, business logic and data components that are ready to be assembled.

This phase is divided into four steps:

- **User Interface Refinement.** The user interface prototype, produced in the Application Design phase, is refined in this step to accommodate new aesthetics and visual requirements. This prototype is an HTML user interface framework that is ready to be assembled with the presentation components.
- **Component Provisioning.** The purpose of this step is to acquire or develop the presentation and business logic components that were specified in the Component Specification Phase. Reusability is a normal practice to be applied in this step. The first activity to be performed in this step is, therefore, the search for components that can be reused from previous projects, component repositories, or acquired from third parties. Based on the results of the component searching step, the development team must accomplish next the following activities:
  - **Acquire.** Some components can be purchased from third parties or specialized software companies.

- **Subscribe.** The services to be offered by some components can be externally supplied on a subscription basis as web services or megaservices.
- **Wrap.** The functionality required from some components can be provided by legacy applications that must be wrapped to provide the required services.
- **Adapt.** Components that have been used in previous projects or that have been found in internal component repositories can be adapted to comply with component specifications and implementation restrictions, such as the component model and the infrastructure selected for deploying the components.
- **Develop.** Those components that can not be reused must be developed in-house from scratch based on the corresponding component specifications.
- **Component testing.** The software components that were developed, adapted or wrapped are tested individually using the testing techniques and procedures that are specified in the V&V plan.
- **Data components implementation.** The databases or XML data stores that were specified in the previous phases are implemented, in this step, using the selected DBMSs.

The Component Assembly Phase

The components acquired or produced in the Component Implementation phase must now be assembled to produce an integrated application, as described by the application architecture. The implementation view of the application architecture is refined and used in this phase to guide the assembly process.

The user interface prototype is extended with the presentation components which, in turn, are linked to the business logic components using the appropriated middleware of the deployment infrastructure (e.g., RMI, IOP, SOAP, and XML). The business entity components must also be linked to the data components using middleware facilities such as XML, JDBC, and ODBC.

The assembly of components must be tested using top-down, bottom-up or combined integration testing techniques, as indicated by the V&V plan.

The application documentation is also written in this phase following the indications given by the documentation plan.

The Web Application Testing Phase

The functional, performance, and acceptation tests are prepared and conducted, in this phase, based on the testing plan provided by the management processes. The main outcome of this phase is a tested web application that is ready to be installed.

The Web Application Delivery Phase

In this phase, the web application is installed and tested in its operational environment. The users and operators are trained based on the training plan produced by the management processes. Once the application is fully operational, it can be formally delivered to the client.

The end of the Delivery Phase signals the beginning of the post-development processes, which take control of the initiation of the web application, its operation, maintenance, and retirement.

6 Conclusions

We have introduced in this paper a component-based method for developing web applications. The distinguished features of the method are the following: (1) its ability to integrate managerial and technical activities into a process model; (2) its emphasis on reusability of components that contributes to reduce development costs and time; (3) its ability to iterate between development phases which is essential to incorporate changes and new requirements; (4) the emphasis on validation and verification that contributes to increase the quality of the deliverables of the project, and (5) the addressing of nontechnical issues such as a business modelling that helps developers to understand much better the system's overall functions, its objectives, business processes, and requirements.

References

[1] Allen, P., 2001. *Realizing e-Business with Components*, Addison-Wesley, London.

[2] Allen, P. and Frost, S., 2001. Planning Team Roles for CBD. In Heineman, G.T and Councill, W.T (Eds.), *Component-Based Software Engineering*. Addison-Wesley, Boston, 113—129.

[3] Bachman, F. et al, 2000. Volume II: Technical Concepts of Component-Based Software Engineering. Technical Report, *Software Engineering Institute*, CMU/SEI-2000-TR-008, May.

[4] Barry, Ch. And Lang, M. A Survey of Multimedia and Web Development Techniques and Methodology Use, *IEEE Multimedia*, 8 (2), 52—60.

[5] Booch, G., Rumbaugh, J. and Jacobson, I., 1999. *The Unified Modeling Language User Guide*, Addison-Wesley.

[6] Brinkkemper, S., 1996. Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, **38**, 275--280.

[7] Conallen, J., 1999. Modeling Web Application Architectures with UML, *Comm. of the ACM*, 42 (10), 63—70.

[8] Conallen, J., 2000. *Building Web Applications with UML*. Addison-Wesley, Reading, Massachusetts.

[9] Curphey, M. et al, 2002. A Guide to Building Secure Web Applications and Web Services. *The Open Web Application Security Project (OWASP).* http://www.owasp.org

[10] Deshpande, Y. and Hansen, S., 2001. Web Engineering: Creating a Discipline among Disciplines, *IEEE Multimedia*, 8 (2), 82—87.

[11] Fraternali, P. and Paolini, P., 2000. Model-Driven Development of Web Applications: The Autoweb System. ACM Trans. on Information Systems, 28 (4), 323-382.

[12] Gaedke, M. and Rehse, J., 2000. Supporting Compositional Reuse in Component-Based Web Engineering, Proc. Of the ACM SAC'2000 Conference, Como, Italy, 927—933.

[13] Hadjerrouit, S., 2001. Web-based Application Development: A Software Engineering Approach, *ACM SIGCSE Bulletin*, 33 (2), 31—34.

[14] Eriksson, H. E. and Penker, M., 2000. *Business Modeling with UML: Business Patterns at Work*, John Wiley & Sons, New York.

[15] Ginige, A. and Murugesan, S., 2001a. Web Engineering: An Introduction, *IEEE Multimedia*, 8 (1), 14—18.

[16] Ginige, A. and Murugesan, S., 2001b. The Essence of Web Engineering, *IEEE Multimedia*, 8 (2), 22—25.

[17] IEEE Std. 1074, 1995. IEEE Standard for Developing Software Life Cycle Processes, IEEE Computer Society, New York.

[18] ILOG. 2001. Business Rules: Powering Business and e-Business. White Paper. May. www.ilog.com

[19] Lynch, P. J. and Horton, S., 1999. *Web Style Guide: Basic Design Principles for Creating Web Sites*, New Haven: Yale University Press.

[20] Odell, J.J., 1996. A Primer to Method Engineering. INFOSYS: The electronic newsletter for information systems. 3 (19).

[21] Van der Aalst, W.M.P, Barthelmess, P., Ellis, C.A., Wainer, J., 2000. PROCLETS: A Framework for Lightweight Interacting Workflow Processes. *Int. J. of Cooperative Information Systems*, 1—40.