2010 34th Annual IEEE Computer Software and Applications Conference

Process Patterns for Web Engineering

Reza Babanezhad, Yusef Mehrdad Bibalan, Raman Ramsin
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
babanezhad@ce.sharif.edu, bibalan@ce.sharif.edu, ramsin@sharif.edu

***Abstract*—Web Engineering has been enriched with processes and modeling languages that focus on the specific features of web-based systems, taking into account the special requirements and constraints that are associated with this specific context.**
Process Patterns, on the other hand, represent elements of knowledge and experience in software engineering; they also act as reusable method chunks that can be used for constructing bespoke methodologies that are tailored to fit specific project situations.
We propose a set of process patterns based on current web development practices. A number of prominent web development methodologies have been studied, and a set of process patterns has been elicited through abstracting their commonalities and identifying the essential activities required in a web engineering endeavor. Furthermore, a web-based systems development framework has been proposed that organizes these patterns into a generic lifecycle. The process patterns can be instantiated and assembled into a high-level development process based on the generic framework proposed.

Keywords–Web-Based Systems Development, Software Development Methodology, Situational Method Engineering, Process Pattern.

I. INTRODUCTION

Due to the rapid expansion and evolution of computer networks and the Internet, Web-based systems have expanded in scale and scope. Organizations face an increasing need to address distribution requirements, and the advent of the Web has provided a powerful medium for tackling this issue. Using the web, however, requires adherence to certain standards and communication protocols. A Web-based system is a software-intensive system that is based on technologies and standards of the World Wide Web Consortium (W3C), and that provides Web-specific resources through a Web browser [1]. These resources are usually in the form of information content or services.

Although developing web-based systems by using traditional software development methodologies has long been the prevalent practice, such an approach tends to overlook the special characteristics of Web-based systems and projects, and quality suffers as a consequence. Some important requirements of Web-based products are: context sensitivity, high security, and maintainability in a continuous sense. Moreover, Web development projects typically have to cope with special constraints, such as limited time and budget, changing organizational processes, competitive pressure, and extreme volatility of requirements [2]. Over the years, this has resulted in the advent of methodologies specialized for Web-based development. Web Engineering has thus become an important discipline in its own right. Web development processes have special properties such as short development cycles, flexibility, configurability, attention to certain Web-specific non-functional requirements, focus on organizational processes and their mutability, attention to user variety, embracing requirements volatility, explicit support for maintenance activities, provisions for managing multitudes of small-sized teams, and support for parallel and distributed development.

A software pattern is an abstraction that describes a proven solution to a common problem in a certain software development context. Similarly, software *process* patterns are the results of applying abstraction to successful software development activities and practices. Process patterns were first defined as “the patterns of activity within an organization (and hence within its project)” [3]; however, the definition that is widely accepted today has been provided by Ambler, who regards a process pattern as “a pattern which describes a proven, successful approach and/or series of actions for developing software” [4].

Ambler has also proposed a set of process patterns for object-oriented development [4, 5], dividing them into three categories based on granularity and abstraction level: Phase, Stage and Task. A Task process pattern defines the steps necessary for performing a specific, fine-grained task in a project. A Stage process pattern depicts the steps in a single project stage and can contain Tasks or other Stages. Stages are often performed in an iterative manner. A Phase process pattern corresponds to a coarse-grained phase of the development lifecycle, and represents a number of constituent Stages and their interactions. A generic process framework can be created by organizing Phase process patterns into a development lifecycle. Ambler has introduced the Object-Oriented Software Process (OOSP) as a generic process framework for object-oriented software development. An object-oriented process can be generated by instantiating this framework and its constituent Phase, Stage, and Task process patterns.

Process patterns can thus be used as components for building a methodology, especially in Situational Method Engineering (SME), where a custom methodology is engineered based on the requirements of the development situation at hand. In assembly-based SME [6], method chunks/fragments are selected from a repository and assembled in such a way as to satisfy project requirements and/or organizational needs. Process patterns can be used as method chunks; as an example, the OPEN Process Framework (OPF) incorporates a library of reusable method components, many of which are process patterns [7, 8].

We propose a set of process patterns for Web Engineering, extracted from Web systems development processes and practices. Furthermore, a generic process framework (at the lifecycle level) for Web-based systems development is also introduced. This framework and its constituent process patterns can be used for process construction and evaluation. Web-based development processes can be compared and assessed as to their conformance to the proposed framework and patterns. The process patterns can be imported into a Computer-Aided Method Engineering (CAME) tool, where they can be instantiated, tailored, and assembled by method engineers into bespoke methodologies.

The rest of this paper is structured as follows: In Section 2, a brief review of seven Web development methodologies is presented; Section 3 provides a description of the proposed generic framework; in Section 4, the proposed process patterns are defined in detail; in Section 5, the main properties of the framework are discussed; Section 6 validates the process patterns through demonstrating how they correspond to existing Web Development methodologies; and Section 7 presents the conclusions as well as suggestions for furthering this research.

II. A Review of Prominent Web Development Methodologies

In order to elicit the target process patterns, a select set of prominent Web development methodologies have been studied in detail. In this section, seven methodologies of this set are briefly reviewed. The main reasons for selecting these methodologies are that they all offer novel features for Web development, and that adequate resources and documentation are available on their processes, thus facilitating their analysis. The methodologies reviewed herein are: XWebProcess, AWE, ICDM, WebHelix, MPM, OOHDM, and the Conallen methodology [9].

XWebProcess [10] is an XP-based agile process for Web development. This methodology extends the XP process [11] by changing and adding certain activities. The Web-development activities that have been incorporated are: Web navigation and presentation design, Web testing, and Web support. The main objective of the methodology is to build high quality Web applications while using an effective scheduling scheme.

AWE [12] is an agile process developed to address Web development issues and requirements, such as short development cycles, management of small and multidisciplinary teams, and delivering the software along with the data. Delivering software along with data means that data should be prepared during development, unlike traditional development where only software components need to be delivered. Business analysis and evaluation of legacy systems are considered as separate phases in this methodology's process.

ICDM is a framework for developing business Web-based system. E-business systems have three main properties: they are business-based, which means that they should be defined by the business strategy, not the implementation technology; they are customer-based, which means that the requirements of external users should be considered as pivotal; and they should be change-centric, requiring that short development cycles and evolutionary methods be used in their development, so that changes in customer requirements are easily accommodated [13]. This methodology tries to address these issues by incorporating specialized activities, such as development of strategies, into its process.

The objective of WebHelix [14] is to present a light and effective way for developing Web-based systems. Some major characteristics of the methodology's process are: support for prototyping, iterative-incremental model, and minimal production of documents. An enhanced version of this methodology was introduced in [15].

MPM [16] offers a process based on prototyping. This methodology considers Web-based systems as organic systems which need to be adapted continuously with their environment. The methodology puts great emphasis on maintenance activities. In the maintenance phase, all activities that were executed in previous phases will be repeated to satisfy new or modified requirements.

OOHDM [17] provides a model-based approach to Web development. This methodology does not incorporate a specific process model. Its process consists of five steps or phases that can be performed based on an iterative-incremental or prototyping development model. The distinguishing feature of this methodology is its strong emphasis on Web design, prescribed as consisting of three design phases: conceptual design, navigational design, and abstract interface design.

The Conallen methodology [9] is based on RUP [18] and the Iconix Unified Process [19], and uses UML as its modeling language. Like RUP, this methodology is use-case driven and architecture-centric, and sports an iterative-incremental process.

III. Proposed Web-Based Software Development Process (WBSDP)

Based on the processes studied, a generic process framework for Web-based system development has been developed, a detailed description of which will be provided in this section; we will refer to the framework as the Web-Based Software Development Process (WBSDP). The framework provides a high level organization for the process patterns proposed (Fig. 1). WBSDP consists of four phases: Start-up, Construction, Transition, and Maintenance. These phases are preformed serially, but each phase contains stages which can be executed iteratively. In other words, WBSDP is "serial in the large and iterative in the small" [4].

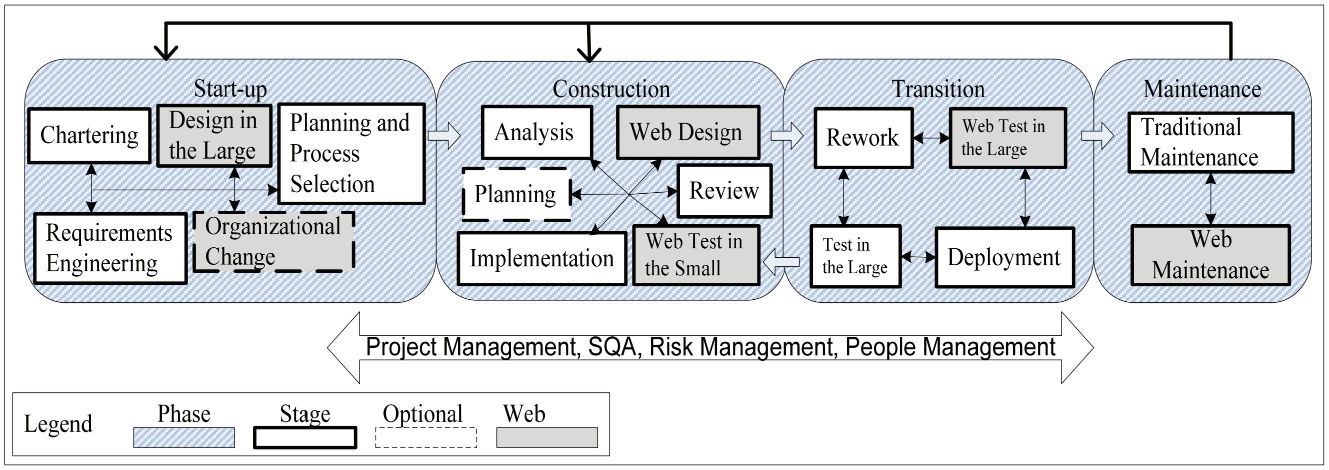

Figure 1. Web-Based Software Development Process (WBSDP)

The Start-up phase includes the activities essential for initiating a project. In Construction, software design, implementation, and test will be performed. Transition includes those activities which are needed for deploying the developed system into user environment. Maintenance is started after deployment, and as expected, consists of maintenance and support activities. In order to correct the errors detected during Transition, redesign and reimplementation are necessitated; hence the backward arc from Transition to Construction. Some changes requested during Maintenance may need a change in system architecture and scope; it should therefore be possible to move from Maintenance to Start-up. Some of the changes requested during Maintenance do not affect the system's core definition, and some others are just requirements that have not been implemented because of time shortages or low priority. In order to implement these types of requirements, a transition from Maintenance to Construction has been incorporated into the process.

Umbrella activities are also considered in the framework, as elicited from the methodologies studied. These activities are depicted on the arrow at the bottom of Fig. 1. Umbrella activities are essential for proper project management, as well as for quality assurance, risk management, and people management.

IV. PROPOSED PROCESS PATTERNS

In this section, the process patterns proposed and organized in the WBSDP are described in detail. Each Phase process pattern is described in a separate subsection along with its constituent patterns. Stages which are exclusive or essential to Web development (shown as shaded boxes in Fig. 1) are explained in more detail. Common stages, which are the same as their counterparts in traditional processes, are just touched upon; the interested reader is referred to [4, 5, 20] for more detailed descriptions on these. To present the patterns, we use a concise version of the template suggested by Ambler [4]. It should be noted that some patterns have been designated as optional, as their inclusion in a Web development process is not mandatory; indeed, many of the existing Web development methodologies do not address these optional activities in their processes.

A. Start-up Phase

This phase is the first phase of Web-based software development, and as such, sets the stage for the main development phases. At the beginning of this phase, development teams do not have any essential information about the system and the project. Much of the team's knowledge comes from previous experiences in similar system development projects. Knowledge should be gained on issues such as the main problems motivating the development project, and the requirements and constraints that should be addressed. The goal in this phase is to gain familiarity with the system, and also to prepare for transition to the Construction phase. This phase includes activities such as: determination of system development goals and objectives, determination of system development problems and constraints, specification of core functional and non-functional requirements, assessment and study of organizational processes/strategies (and their improvement), and determination of the main components of the system. After managing the risks involved and determining a process for the Construction phase, a plan is also prepared to guide the software construction effort. These activities are performed through the phase's constituent stages, which are described throughout the rest of this subsection.

Chartering. This stage initiates the Start-up phase. The main intent of Chartering is to achieve basic familiarity with the system, especially as to its scope and objectives. The activities of this stage include: determination of the main development goal, essential problems, current state and desirable state of the organization, system scope and target market, and assessment of the organization's legacy systems and determination of their main shortcomings. Feasibility analysis should also be performed, and risks should be identified. An early organizational business process model may also be developed.

Requirements Engineering. The goal of this stage is to obtain detailed information about the problem domain in order to identify core functional and non-functional requirements. Functional requirements can be modeled by use-case models or through specifying system operations. Non-functional requirements, especially those related to Web applications (such as security and usability), should also be determined. If the organization is dependent on legacy systems (which is the typical case), assessing and observing these systems can be helpful in eliciting the requirements.

An initial problem domain model is usually developed as an output of this stage. It typically consists of structural object-oriented models (such as class diagrams) and/or information models (such as entity-relationship diagrams). Behavioral models can also be developed to highlight significant interactions.

Prototyping is also conducted, mainly in order to help determine the requirements and the risks involved; however, these prototypes are usually thrown away once they have served their purpose. As in all stages focused on requirements elicitation and specification, system users should get actively involved in the process.

Design in the Large. Constraints, objectives and requirements have so far been determined, but there is no overall view that describes how these constraints and requirements should be realized. The intent of this stage is to present a high level design which defines the architecture of the system as presented from various viewpoints. Sub-systems, major components, hardware architecture, and the distribution of software components on hardware elements should be determined. Constraints and non-functional requirements should be considered in the development of the architecture.

The architecture is typically developed as a prototype, which is gradually refined and completed. The tasks which are performed in this stage focus on designing the architecture of the system, application, software, and information. The distributed architecture of network hardware components is determined during System Architecture Design. Application Architecture Design determines the subsystems and main components which constitute the system. Software components are determined during Software Architecture Design; this architecture shows the constituent Web servers and application servers, and their relationships. This task is optional, since it is not needed for simple systems. Information Architecture Design aims at providing a sitemap which shows how information is made accessible throughout the system.

The architecture defined in this stage is basic and should be completed in downstream stages. If the Web system communicates with other systems, the communication technologies and middleware involved should also be determined. The tasks which constitute this stage are shown in Fig. 2.

Organizational Change. Introducing a Web system into an organization may necessitate applying changes to its business processes and strategies. In this stage, current business processes and strategies are revised and updated according to the specifications of the system under development. It is therefore necessary to complete the business process model which was initially produced during the Chartering stage. Some special properties of Web systems, such as distribution, may necessitate applying extensive modifications to organizational business processes.

Migrating to e-business is a strategic decision for business organizations. Introducing Web-based systems tends to entail an organization-wide ripple effect. It can therefore have a great impact on business strategies, even at the enterprise level. The Organizational Change stage addresses this issue as well. This stage is optional, however, since certain Web-based systems (such as static and document-centric systems) may have no effect on business processes and organizational strategies.

Planning, and Process Selection. An initial architecture has thus far been prepared, and the main functional and non-functional requirements have been specified. This stage intends to determine an overall plan and a development process that will govern the development effort throughout the Construction phase, with the ultimate aim of realizing the requirements. During this stage, development teams are formed, communication paths and mechanisms are established, and management processes are determined. It is important to note that unlike development teams in many traditional methodologies, Web development teams are multidisciplinary: people with widely varying skills, such as programmers, analysts, graphic designers, and Web interface/navigation designers, are members of the same team. Therefore, inter-team and intra-team communication is a critical issue.

Requirements are prioritized, tasks are assigned to development teams and individuals, and a development schedule is determined in this stage. Furthermore, tasks which can be performed in parallel are identified, and milestones are defined in order to facilitate project monitoring and control.

A process model for the Construction phase should also be determined in this stage; when developing a small and stable system, a waterfall model may be chosen, whereas a rapidly changing system will probably require an iterative process. If an iterative process is chosen, the number and duration of the iterations should also be estimated.

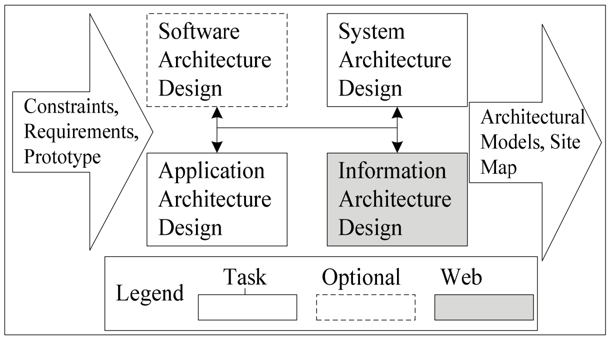
Constraints, Requirements, Prototype
Software Architecture Design
System Architecture Design
Application Architecture Design
Information Architecture Design
Architectural Models, Site Map
Legend
Task
Optional
Web

Figure 2. Design in the Large

B. Construction Phase

The intention in this phase is to construct the system so that the requirements are met. Detailed analysis of the problem domain and the requirements, architectural and detailed design of Web-specific artifacts, implementation, and testing (verification and validation) are the most important activities in this phase. Integration, if necessary, will also be performed. The functionality of the system and the realization of certain Web-related requirements is verified and validated through applying testing-in-the-small techniques.
The phase also includes a review stage, during which all construction activities, artifacts, overall process model, and plans, are reviewed and revised as required.

Planning. The Construction phase is executed based on the process selected in the previous phase. The Planning stage is only performed if an iterative process has been chosen for the Construction phase. The aim of this stage is to develop a detailed plan for the current iteration. The requirements that should be implemented during the current iteration are determined, implementation tasks are assigned to teams and individuals, and an iteration schedule is prepared.

Parallel execution of construction tasks by multiple development teams is desirable, especially in Web-development projects where a high degree of concurrency is achievable; for example, navigation and interface design can be performed in parallel with design and implementation of the business logic. However, parallel construction requires rigorous planning and scheduling so that proper coordination is applied.

Analysis. The development teams have already obtained basic knowledge about the problem domain. However, building a software system requires a more detailed analytical knowledge of the problem domain. This stage intends to perform a detailed analysis of the problem domain, and produce a descriptive model.

If an iterative process model is used for the Construction phase, detailed analysis is only done on those parts of the system which are focused upon in the current iteration. Typically, the detailed problem domain model produced in this stage accentuates the structural and behavioral aspects that are relevant to the development effort undertaken in the current iteration.

Web Design. The aim of this stage is to present a design model for the Web-based system. As shown in Fig. 3, this stage includes four principal substages: Server-side Design, Presentation Design, Navigation Design, and Content Design. Two optional tasks can also be performed in this stage: Prototyping, and Assessing Similar Web Pages. Prototyping can be used in all the four principal substages as well. Studying and assessing similar Web Pages can help enrich the design produced. The four principal substages will be further explained throughout the rest of this subsection.

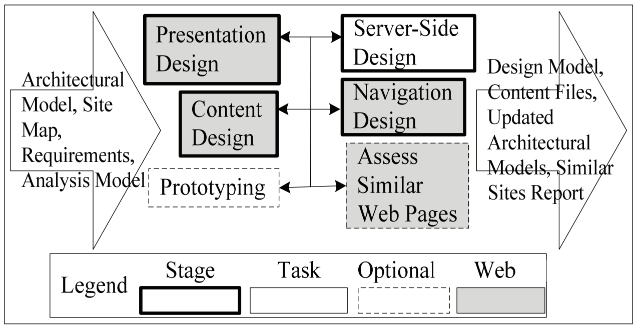
Architectural Model, Site Map, Requirements, Analysis Model
Presentation Design
Server-Side Design
Content Design
Navigation Design
Prototyping
Assess Similar Web Pages
Design Model, Content Files, Updated Architectural Models, Similar Sites Report
Legend
Stage
Task
Optional
Web

Figure 3. Web Design

Server-side Design. The aim of this stage is to transform problem domain models to solution domain models by designing the server-side logic. Architectural and information models are mainly used for this purpose; the models are usually refined and updated during the course of this stage. This stage is very similar to its *design* counterpart in traditional processes, and will therefore not be elaborated further.

Content Design. The aim of this stage is to determine and design the contents of the Web pages in detail. The constituent tasks are depicted In Fig. 4.

Content Specification determines the contents of each page. During Content Provision, the contents are designed as presentable objects. Client-side Logic Design determines the processing logic executed by clients, typically implemented by using script language or applets. This sort of logic is usually very straightforward, typically just involving the application of simple checks and verifications. Transaction Specification specifies the transactions that are managed by the page, the events that trigger them, and the data interchanged with Web servers. We can also have a Content Model for Web pages whose contents are generated dynamically. In this model, each dynamic page element is related to one or more server-side entities.

Navigation Design. In Web engineering, it is not enough to just design the contents of the Web pages; it is also necessary to specify how the information contained in the Web pages can be accessed through traversing the pages. This stage intends to design these access paths.

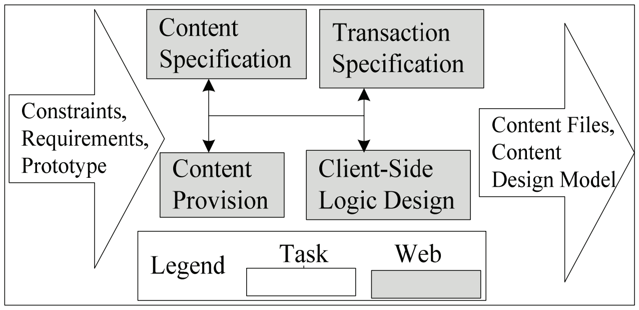
Constraints, Requirements, Prototype
Content Specification
Transaction Specification
Content Provision
Client-Side Logic Design
Content Files, Content Design Model
Legend
Task
Web

Figure 4. Content Design

The most important task in the Navigation Design stage is Link Specification. During this task, the relationships between Web pages are determined, and multidimensional content is generated via using links. In dynamic pages, a Navigation Model is produced based on the content model. The Navigation Model determines the dynamic links and the relationships with entities of the logic layer. Page Access Specification specifies the links through which a page is made accessible. Page Location indicates the location of a page as related to the site. The site map is updated and refined during this stage. Fig. 5 shows the constituent tasks of this stage.

Presentation Design. This stage intends to determine how information entities and links should be presented in Web pages. During this stage, the presentation aspects of Web pages are designed, and the constituent GUI elements, such as menus, windows, buttons, and images, are specified. The tasks of this stage are depicted in Fig. 6.
During the Layout Design task, different page templates are designed and the mechanism for applying them is specified. It is also determined how information entities and links can be represented by GUI elements. Triggering events, which start the transactions specified in Content Design, are assigned to GUI elements. The dynamic parts of the pages are determined, and suitable mechanisms are selected for implementing them.
Users should be involved and in close contact with interface designers during Presentation Design. User-interface-dependent non-functional requirements, such as usability and context sensitivity, are very important in this stage, and should be stringently observed.

Implementation. In this stage, the system (or part of it, as planned for the current iteration) is implemented. The implementation effort focuses on the information objects determined during Content Design, and the GUI elements specified during Presentation Design.

Implementing the GUI layer is typically performed by using an appropriate language, such as HTML. After implementation, integration is typically performed to consolidate the increment produced with the system built so far. Tools are extensively used during this stage of the development process.

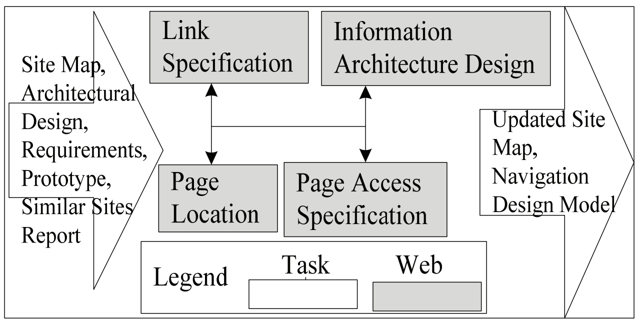
Site Map, Architectural Design, Requirements, Prototype, Similar Sites Report
Link Specification
Information Architecture Design
Page Location
Page Access Specification
Updated Site Map, Navigation Design Model
Legend
Task
Web

Figure 5. Navigation Design

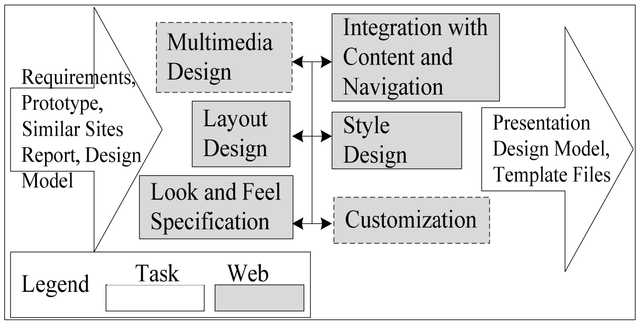
Requirements, Prototype, Similar Sites Report, Design Model
Multimedia Design
Integration with Content and Navigation
Layout Design
Style Design
Look and Feel Specification
Customization
Presentation Design Model, Template Files
Legend
Task
Web

Figure 6. Presentation Design

Web Test in the Small. The system (or parts of it) has been developed, and verification and validation should now be performed in order to ensure the correctness, validity, and adequate quality of the parts implemented thus far. This stage includes two constituent substages, as shown in Fig. 7.

Functional Test in the Small contains activities that verify the realization of the functional requirements and the server-side logic (much similar to the Test in the Small stage of OOSP [4]).

Web-Specific Test in the Small focuses on verifying and validating the Web-related parts of the system (designed during the Web Design stage).

Other common activities, such as using test tools, generating test documents, and analyzing test results, are also performed during this stage. Users should actively participate in all Web-specific testing activities. The two substages of this stage are further explained throughout the rest of this subsection

Functional Test in the Small. The functional requirements that have been implemented should be verified and validated in this substage. This substage intends to test those parts of the system which implement the processing logic, often installed on servers.

Unit testing is typically performed on each operation or method. Regression testing and integration testing is performed as and when required. Test tools and test-driven development environments can be extensively used in this stage.

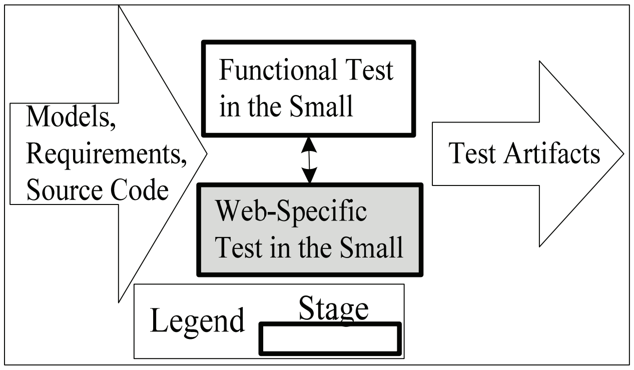
Models, Requirements, Source Code
Functional Test in the Small
Web-Specific Test in the Small
Test Artifacts
Legend
Stage

Figure 7. Web Test in the Small

Web-Specific Test in the Small. The content, navigation, and presentation elements that have been implemented (at least in part) are verified and validated during this substage.

Special attention is given to verification and validation of the design and layout of GUI elements, links, dynamic aspects of Web pages, and context sensitivity issues. Realization of non-functional requirements, such as usability, is meticulously tested.

During content testing, content accuracy and client-side logic are verified and validated. For each of the Web pages, transactions which start from other pages and end at the current page are tested for correctness. The main tasks of this stage are depicted in Fig. 8. A variety of test documents are produced as the output of this substage.

Review. A review is performed at the end of the Construction phase, and also at the end of each stage and iteration, in order to assess the work completed so far. During stage reviews, stage artifacts are scrutinized, plans and schedules are revisited and checked against the actual progress of the project, and the problems which have occurred during the stage are discussed and resolved.

In a phase or iteration review, in addition to stage review issues, the Construction process is scrutinized for possible deficiencies, and changes are made to the process as needed. The requirements which have not been implemented are determined, and provision is made for their implementation in later iterations by making the necessary changes to the plans and schedules.

C. Transition Phase

The whole system or parts of it have been designed and implemented during the Construction phase, ready to be delivered to the end-user. The Transition phase includes activities for deploying Construction artifacts into the user environment.

Activities such as large-scale (system- or subsystem-wide) verification and validation, training, deployment, and tuning, are performed during this phase. Errors detected during this phase can be corrected in this phase, or can be relegated back to the Construction phase if extensive rework is required. Deployment can be performed iteratively, or just once at the very end of the development effort.

The stages of this phase are further explained throughout the rest of this subsection.

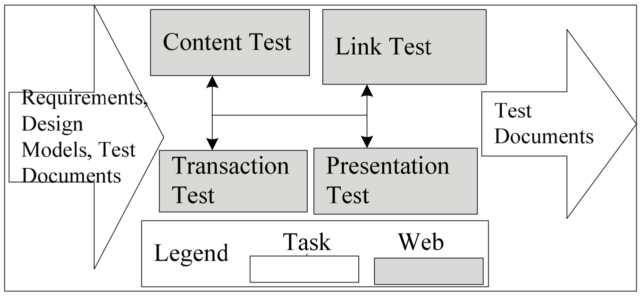

Figure 8. Web-Specific Test in the Small

Test in the Large. A set of functional and non-functional requirements have been implemented in the increment produced, and thorough testing is required before deployment can commence. This stage aims at large-scale verification and validation of the increment, or if the system is a small one, the whole system.

The main focus is on testing server-side logic, thus treating the Web system like a traditional system, totally ignorant of the Web-specific aspects. Acceptance testing and system testing are the main types of tests performed in this stage.

Web Test in the Large. This stage intends to verify that the developed product satisfies Web-specific functional and non-functional requirements. The tasks of this stage are depicted in Fig. 9. User interface artifacts are checked to ensure that user interface elements function and interact in an orderly and correct fashion, customizable pages work properly and can be customized easily, and the user interface is suitably displayed on different devices. Another issue to be tested is the context sensitivity of Web pages: the user interface should be adjusted and presented according to the user's location, time zone, culture, and special needs. Web-specific non-functional requirements such as security, performance, usability, and scalability are tested. Complex transactions, and the dynamic pages which are generated as a result of a transaction, are also tested for inaccuracies and inconsistencies.

Rework. This stage aims at removing the errors discovered during previous stages. In this stage, errors will be assessed, and if error correction is straightforward, corrections are made to the code. If error correction requires extensive redesign, correction is relegated to the Construction phase.

Deployment. This stage aims at system conversion: ultimate deployment of the Web application into the user environment. Traditional deployment activities, such as training, installation of the hardware/software platform, documentation, and data conversion are performed as in any other type of software development project. Furthermore, Web-specific deployment activities, such as installation of network hardware, application servers, and Web servers, are also performed.

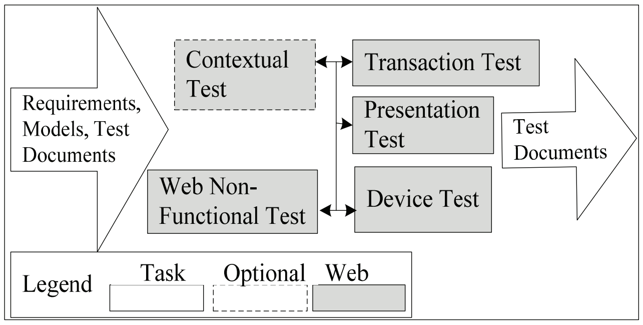

Figure 9. Web Test in the Large

D. Maintenance Phase

Now that the Web-based system has been successfully put into production, support and maintenance activities should commence. System maintenance is the main goal of this phase. The main activities of this phase include: modifying the system to keep it free of malfunctions, performing system operation and support activities, and conducting special maintenance activities necessary for Web systems. The stages of this phase are further detailed throughout the rest of this subsection.

Traditional Maintenance. This stage contains the typical activities performed for maintaining software systems. Operational and support activities are also performed in this stage. Preventive, adaptive, corrective, and perfective maintenance activities are performed. For complex changes, it is necessary to restart the development cycle; a return is therefore made to the Start-up or Construction phases, depending on the type and scale of the change requested.

Web Maintenance. Web systems need additional activities for maintenance. As shown in Fig. 10, this stage includes four main Web-specific maintenance tasks. The Manage Content task includes activities for adding new content to the Web pages, or updating them. Web pages which are not updated frequently enough are considered inaccurate and old, thus losing their appeal. Making periodic quality checks, especially aiming at assessing critical issues such as security and performance, is essential. If the target of maintenance is a business system, and user notification is considered important, issues such as advertising in other sites, registering in search engines, and sending newsletters, should be addressed. Maintaining network hardware is also of utmost importance.

V. Characteristics of WBSDP

As mentioned earlier, WBSDP is the result of extracting and organizing process patterns from a multitude of Web development processes. In this section, we present some of the important characteristics of this framework. These characteristics, as listed below, address the essential requirements of Web development:

- Generating processes based on different process models: Processes instantiated from WBSDP can be based on different process models, such as iterative-incremental, RAD, and prototyping.

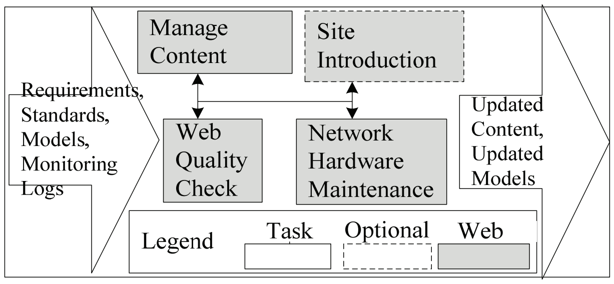
Manage Content
Site Introduction
Requirements, Standards, Models, Monitoring Logs
Web Quality Check
Network Hardware Maintenance
Updated Content, Updated Models
Legend
Task
Optional
Web

Figure 10. Web Maintenance

- Attention to process selection: There is a specific stage in the Start-up phase to select a process for the Construction phase; WBSDP is therefore flexible.
- Supporting agility: Properties such as embracing change, user involvement, continuous review, and frequent releases make the WBSDP capable of accommodating agile processes.
- Adequate consideration for project initiation activities: Misunderstanding the user's requirements and overlooking initial analysis are the two main reasons for the failure of Web projects; in WBSDP, initiation is performed in a dedicated phase.
- Attention to organizational issues: A stage has been dedicated to analyzing business processes and strategies, and modifying them accordingly.
- Architecture-centricity: The initial architecture of the system will be developed before Construction, and non-functional requirements are considered during architectural design.
- Specification of Web-specific activities as separate stages: examples include: Web Design, Web Test, and Web Maintenance.

VI. Realization of Proposed Process Patterns in Web-Based Methodologies

Completeness and adequate coverage of the proposed process patterns needs to be assessed by demonstrating that each phase of the studied Web-based methodologies is indeed covered by the process patterns proposed. Table 1 shows the correspondence between the patterns proposed and the seven methodologies studied. The comparison shows that the proposed framework and patterns do indeed provide adequate coverage of Web engineering activities.

VII. Conclusions and Future Work

We have proposed a set of process patterns and a process framework for Web engineering. To generate these patterns, the processes of a select set of Web development methodologies have been studied, and subprocesses which are common among many methodologies have been extracted as patterns. These patterns were then organized into a generic process framework. To assess the completeness and adequate coverage of the proposed process patterns, the activities of the seven methodologies have been mapped to the phases and stages of the proposed framework.

This research can be furthered by completing the framework and customizing it for each and every type of Web system. Task process patterns should be further refined and detailed, thus facilitating the use of the framework and patterns in real SME projects. Another strand of research can focus on developing an extension framework for adding Web-development support to traditional development processes.

Acknowledgement

We wish to thank Iran Telecommunications Research Center (ITRC) for sponsoring this research.

TABLE I. REALIZATION OF PROPOSED PROCESS PATTERNS IN STUDIED METHODOLOGIES

Methodologies	Phases	Corresponding Stage Process Patterns
XWEBProcess	Explore	Chartering, Design in the Large
	Define and Review Requirements	Requirements Engineering, Design in the Large
	Planning	Planning and Process Selection
	Develop	Analysis, Web Design, Implementation, Web Test in the Small
	Web testing	Web Test in the Large
	Production	Deployment
	Web Support	Maintenance (especially Web Maintenance)
AWE	Business Analysis	Chartering, Organizational Change
	Requirements Analysis	Requirements Engineering
	Design	Web Design, Review
	Implement	Implementation, Web Test in the Small
	Test	Test in the Large, Web Test in the Large
	Evaluation	Chartering, Test in the Large, Web Test in the Large
	Deploy Web Application	Deployment
ICDM	Strategy Development	Organizational Change, Chartering
	Requirement Analysis	Requirements Engineering
	Architecture	Design in the Large
	Design	Web Design
	Implementation	Implementation
WebHelix	Business Analysis	Chartering
	Planning	Planning and Process Selection, Design in the Large
	Analysis	Requirements Engineering, Analysis
	Design	Web Design, Planning
	Code	Implementation
	Test	Web Test in the Small, Web Test in the Large, Test in the Large
	Evaluation	Review
	Deploy	Deployment
	Maintain	Maintenance
MPM	Basic System Requirements	Requirements Engineering
	Architectural Decision	Design in the Large
	Building and Deploying Initial Version	Analysis, Web Design, Implementation, Web Test, Deployment
	Deve-maintenance	Maintenance, Construction
OOHDM	Requirements Gathering	Requirements Engineering
	Conceptual Design	Content Design, Server-Side Design
	Navigational Design	Navigation Design
	Abstract Interface Design	Presentation Design
	Implementation	Implementation
Conallen Methodology	Analyze Business and Perceived Problems	Chartering
	Develop Domain Model	Requirements Engineering, Analysis
	Analyze the Understood Problem	Requirements Engineering, Chartering
	Develop Vision Document	Chartering
	Develop Project Plan	Planning and Process Selection
	Iterate	Construction
	Deploy System	Deployment
	Maintenance	Maintenance

References

[1] G. Kappel, B. Pröll, S. Reich and W. Retschitzegger, ”An Introduction to Web Engineering,” In Web Engineering: The Discipline of Systematic Development of Web Applications, G. Kappel, B. Pröll, S. Reich, and W. Retschitzegger, Eds., John Wiley & Sons, 2006, pp. 1-22.

[2] A. McDonald and R. Welland, “Web Engineering in Practice,” Proc. Workshop on WWW10, 2001, pp. 21-30, doi: 10.1.1.73.4099.

[3] J.O. Coplien, “A Generative Development Process Pattern Language,” In Pattern Languages of Program Design, ACM Press/ Addison-Wesley, 1995, pp. 187–196.

[4] S.W. Ambler, Process Patterns: Building Large-Scale Systems Using Object Technology, Cambridge University Press, 1998.

[5] S.W. Ambler, More Process Patterns: Delivering Large-Scale Systems Using Object Technology, Cambridge University Press, 1999.

[6] J. Ralyté, R. Deneckere and C. Rolland, “Towards a generic model for situational method engineering,” Proc. International Conference on Advanced Information Systems Engineering (CAiSE’03), Jun. 2003, pp. 95-110, doi: 10.1007/3-540-45017-3_9.

[7] B. Henderson-Sellers, “Method Engineering for OO Systems Development,” Communications of the ACM, vol. 46, no. 10, Oct. 2003, pp. 73–78, doi: 10.1145/944217.944242.

[8] OPF Repository, available at: http://www.opfro.com.

[9] J. Conallen, Building Web Applications with UML, Addison Wesley, 2002.

[10] A. Sampaio, A. Vasconcelos, and P. Sampaio, “Design and Empirical Evaluation of an Agile Web Engineering Process,” Proc. 18th Brazilian Symposium on Software Engineering (SBES’04), 2004, pp. 194-209.

[11] K. Beck and C. Andres, Extreme Programming Explained: Embrace Change, Addison-Wesley, 2004.

[12] A. McDonald and R. Welland, “Agile Web Engineering (AWE) Process,” Technical Report TR-2001-98, University of Glasgow, Scotland, 2001, doi: 10.1.1.63.6832.

[13] C. Standing, “The requirements of Methodologies For developing Web applications,” In Web-Engineering: Principles and Techniques, W. Suh, Ed., Idea Group, 2005, pp. 261-280.

[14] G. Whitson, “WebHelix: Another Web Engineering Process,” Computing Sciences in Colleges, vol. 21, no. 5, May 2006, pp. 21-27, doi: 10.1109/52.730844.

[15] N. Subramanian and G. Whitson, Software engineering for modern web applications: Methodologies and technologies, IGI Global, 2008.

[16] J.Q. Chen, and R.D. Heath, ”Web Application Development Methodologies,” In Web-Engineering: Principles and Techniques, W. Suh, Ed., Idea Group, 2005, pp. 76-96.

[17] G. Rossi and D. Schwabe, ”Model-Based Web Application Development,” In Web Engineering, E. Mendes, and N. Mosley, Eds., Springer, 2005, pp. 303-333, doi: 10.1007/3-540-28218-1_10.

[18] P. Kruchten, The Rational Unified Process: An Introduction, Addision-Wesley, 1999.

[19] D. Rosenberg and K. Scott, Use Case Driven Object Modeling with UML: A Practical Approach, Addison-Wesley, 1999.

[20] J. Ralyté, S. Brinkkemper, and B. Henderson-Sellers, Eds., Situational Method Engineering: Fundamentals and Experiences, Springer, 2007.