CONTEXT-ORIENTED WEB METHODOLOGY WITH A QUALITY APPROACH

Anna Grimán, María Pérez, Maryoly Ortega and Luis E. Mendoza
Processes and Systems Department, Simón Bolívar University, PO Box 89000, Caracas 1080-A, Venezuela

Keywords: Web Development, Methodology, Software Quality, Quality Assurance, Agile and Plan-driven methodologies.

Abstract: Dependency on Web systems and applications has increased in recent years. Their use and quality have gained relevance and demands in this field have significantly increased in time. This has driven to the application of processes that include considerations related to business dynamism and quality expectations for the final product. This work is aimed at describing a methodology for Web applications development that ensures quality at all phases and is especially useful in small and medium-sized projects regardless of the platform or architecture used. We performed an analysis of the main existing methodologies that allowed us to extract the best practices known and combining them in the proposed solution. Comparison between agile and plan-driven methodologies established the most suitable process model for this type of development. As a result thereof, a context-oriented web methodology -COWM- was obtained, including best practices to ensure quality throughout the whole process. Finally, a COWM evaluation was performed on a case study in order to prove its applicability and efficiency for Web systems development.

1 INTRODUCTION

According to (Offutt, 2002) (Barry & Lang, 2001) and (Lowe & Henderson-Sellers, 2001), there is a remarkable difference in the quality characteristics of Web applications when compared to traditional developments. Some of the aspects that determine such difference are: coupling among the business model and the system technical design, criticality of the architecture modularity, technological instability, importance of actualization, maintenance of the content and emphasis of the user interface. According to Offutt (2002), this quality dimension is so crucial that Web developing organizations should focus on quality and leave delivery times in a second place.

This article describes a methodology for developing Web applications that ensures quality at all its phases, according to IEEE 1012 (1998) and Kan (2002), regardless of the platform used. We performed an analysis of the most popular methodologies that allowed us to extract the best practices to be included in our proposal. The comparison between agile and plan-driven methodologies resulted in a context-oriented web methodology –COWM- which takes advantage of the efficiency of agile methodologies and the stability of plan-driven methodologies, and includes those practices guaranteeing quality throughout the entire process.

Lastly, a COWM evaluation was performed on a case study in order to prove its applicability and efficiency for a specific Web development.

2 PRIOR WORK ANALYSIS

This section includes the analysis of 9 methodologies for the development of Web applications, based on the 11 criteria proposed in (Boehm & Turner, 2004): Business Modelling, Planning, Risk management, Integration strategies, Change and configuration management, Process improvement, Integration with the client, Use of prototypes, Frequent deliveries, Lifecycle phases, and Quality.

The analyzed methodologies are: Scrum (Schwaber, 1995), Crystal Clear (Letelier & Penadés, 2004), Microsoft Solutions Framework - MSF (Reynoso, 2004), Adaptative Software Development –ASD (Abrahamsson et al., 2002), Dynamic Systems Development Method –DSDM (Canós et al., 2004), Feature Driven Development –FDD (Letelier & Penadés, 2004), eXtreme Programming - XP (Abrahamsson et al., 2002), Watch (Montilva &

Grimán A., Pérez M., Ortega M. and E. Mendoza L. (2008).
CONTEXT-ORIENTED WEB METHODOLOGY WITH A QUALITY APPROACH.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - ISAS*, pages 213-219
DOI: 10.5220/0001689802130219

Barrios, 2002), Rational Unified Process -RUP (Kruchten, 2003).

Upon the analysis of the 11 criteria on these methodologies, we conclude that a methodology that guarantees quality at all phases should fulfil the following requirements: System documentation and user manuals, Change control, Risk management, Knowledge management, Participation of a HCI, Functional test design and inspection, Good planning aimed at determining reasonable delivery times in accordance with the project budget, QA (in particular, maintainability and usability will be the main quality characteristics for Web applications).

3 METHODOLOGY PROPOSAL

Our proposal combines the aforementioned requirements and adapts some practices of plan-driven and agile methodologies to the particular Web systems features, in order to make the development process more adequate for the construction of Web applications. As a result, we have proposed a context-oriented web methodology (COWM). COWM considers 3 phases and other RUP elements as a starting point. It improves delivery times while proposes a clear and specific strategy that ensures Web application quality. Following, we present the practices supporting our proposal; each practice is expressed in COWM as a set of activities or methodological elements.

- **Context-oriented Development.** Contexts are use-cases groups that share similar characteristics. This lets the delivery of high-valuable system versions.
- **Frequent Deliveries.** These let obtaining ongoing feedback from users by verifying their conformity during the development process.
- **Iterative Software Development.** Software is developed in little steps or short iterations, which let a risk prompt identification and verification to provide a proper response.
- **Requirements Management.** It comprises the identification and requirements change management. Both, business and system functional/non functional requirements are managed. This flexibility is achieved by versioning documents and managing traceability among models.
- **Visual Software Modelling.** When using UML, architecture and design can be clearly specified and communicated to all involved parties.
- **Component-based Development.** Software architecture presents more maintainability with a component-based approach, as well as substantial savings of time, resources and efforts for future developments.
- **Methodology Refining.** COWM proposes holding a meeting after every iteration to identify possible changes and improvements to adapt the methodology to the development project features.
- **Frequent Meetings.** COWM proposes weekly meetings in order to verify the project status, identify any obstacles, and eliminate and perform corrective measures.
- **Continuing Quality Verification.** COWM proposes for each phase a group of quality assurance techniques. Each technique is applied to an artefact for allowing us to estimate a quality characteristic. Actually, quality characteristics are evaluated through different metrics depending on the nature of the phase and the artefact evaluated. Table 1 provides some examples of quality features evaluated for the different phases and artefacts.

3.1 Phases

COWM is composed of 4 development phases that are inspired in RUP; however it tailors several RUP components to Web domain. A quality check is performed at the end of each phase.

3.1.1 Definition Phase

The main purpose of this phase is that the stakeholders define the project scope, identify risks, and design the phases and iterations plans. This phase focuses on business-related documents. Table 2 provides an overview of the activities performed in this phase. The artefacts generated in these activities should later be formally revised as follows:

Table 1: Examples of quality features evaluated.

Features	Phases	Metrics
Functionality	Construction	Does the *context* work properly once it is integrated to the rest of the functionality?

Definition Phase Quality Verification

- *Business Architecture Document Inspection.* Objective: Check for properly defined rules, objectives and business processes; concordance

between defined roles and their responsibility; and object adequacy with respect to events. Quality characteristic**:** functionality, reliability, maintainability.

- *Glossary Inspection.* Objective: Verification of terms included in the glossary, properly defined terms, and concordance of project terms. Quality characteristic: functionality.
- *Vision Document Inspection.* Objective: Check that the need originated from this application is properly described from the perspective of the involved individuals; proper identification of those individuals; well-defined high-quality features. Quality characteristic: functionality, reliability, maintainability.
- *Requirements Specification Document Inspection*. Objective: Check for understandability and correctness of specified requirements. Quality characteristic: functionality, reliability, maintainability, usability and accessibility.
- *Project Plan Inspection.* Objective: Check for well-defined work structures; adequate effort estimates; reasonable duration and cost estimates. Quality characteristic**:** functionality, correctness, reliability, maintainability.
- *Risks List Inspection*. Objective: Check for well-defined, complete and properly classified risks, including clear mitigation strategies. Quality characteristic: functionality, reliability, maintainability and usability.
- *Quality Assurance Plan Inspection*. Objective: Check that each test specified in the quality assurance plan has a clearly-defined objective and type. Quality characteristic: functionality, maturity and maintainability.
- *Creative Design Summary Inspection.* Objective: Check that the site's design is in accordance with the target culture (types of users), and that its look and feel is in accordance with the image of the company for which the development is performed. Quality characteristic: usability and accessibility.

3.1.2 Architectural Baseline Phase

Necessary resources and activities are planned by specifying use cases and architecture design. We do not recommend performing more than one iteration in this phase.

Upon completion of this phase, most use cases and actors should have been identified, and the basic software architecture should have been clearly described, including the creation of its prototype. Objectives are oriented to risk management. Table 3 provides an overview of the activities performed in this phase. Note that uses cases identified in the first activity are organized in contexts according to their nature and important for the user, those high-priority contexts will be analyzed and implemented in this phase whereas other contexts will be analyzed in the construction phase.

The artefacts generated in these activities should later be formally revised as follows:

Architectural Baseline Phase Quality Verification

- *Requirements Specification Document Inspection*. Objective: Check that use cases identified have not been already designated under a different name or similar task. Make sure that all use cases with a higher-than-normal impact are really as such, and that all use cases have been.

Table 2: Products' Definition Phase.

Activity	Description / Product / Roles
Business modelling	Describe the business process and define the scope and objectives of the business environment. Artefacts: Business Architecture Document and Project Glossary. Roles: Business Analyst and Business Expert
Scope formulation	Capture the most relevant context, requirements, and restrictions leading to final product acceptance. Artefacts: Vision Document, and Requirements Specification Document. Roles: Client, System Analyst, Quality Assurance Manager, Software Architect.
Initial planning	Prepare work plans, cost estimates, delivery dates, etc. Risks are identified and the quality assurance plan is prepared. Validation and Verification techniques are established to ensure quality of software to be subsequently delivered to users. Artefacts: Project plan, Risks List and Quality Assurance Plan Roles: Project Manager, Quality Assurance Manager, Client, Programmer, and Tester.
Creative interface design	Guarantee basic quality components such as learnability, efficiency, memorization, and satisfaction. Artefacts: Creative Design Summary. Roles: Usability Manager, Interface Designer and Client.
Phase final meeting	Analyze the feasibility of continuing with the project. If the project fails in the following items, it should be cancelled or reanalyzed. Artefacts: Updated plan for the next iteration. Roles: all Stakeholders.

Table 3: Products' Architectural baseline Phase.

Activity	Description / Product / Roles
Identification of Use Cases and Contexts	Identify and classify use cases by context; identify and prioritize use cases with highest impact on the architecture, and proceed with their analysis. The remaining use cases will be analyzed in detail by context at the construction phase. Artefacts: Requirements Specification Document including use cases and contexts. Roles: Analyst and Client.
Architecture analysis and design	The Software Architecture Document –SAD– is a preliminary version that does not include the study of all system use cases. This document shall be improved all along the construction phase by including elements corresponding to the use cases analyzed at each construction iteration. Artefacts: SAD first version (including behaviour diagrams, class diagrams, navigation map, and E-R diagram). Roles: Software Architect and System Analyst.
Preparation of the base application architecture	Obtain an executable architecture version. This version is conceived as an evolving prototype for the purpose of adding system requirements on an incremental basis. Artefacts: Prototype. Roles: Programmer, Interface Designer and Software Architect.
Preparation of the creative design composition	Submit to the client approval a set of visual options for the site's style. This should be done through diagrams that simulate the site's ordering and look. Artefacts: Creative design composition. Roles: Interface designer, Usability Manager and Client.
Phase final meeting	Analyze the feasibility of continuing with the project. If the project fails in the following items, it should be cancelled or reanalyzed. Artefacts: Plan updated for the next iteration. Roles: all the stakeholders.

properly classified into the corresponding groups. Quality characteristic: Functionality, reliability, maintainability

- *Behaviour Diagrams' Inspection.* Objective: Check that behaviour diagrams include the correct interactions. Quality characteristic: Functionality, reliability, maintainability.
- *Class Diagrams Inspection*. Objective: Check for class diagrams' correctness. Quality characteristic: functionality, reliability, maintainability.
- *Initial Navigation Map Review*. Objective: Check for coherence and clear definition of the initial navigation map. Quality characteristic**:** usability, functionality and accessibility.
- *E-R Diagram's Review*. Objective: Check for proper and complete nomenclature. Quality characteristic: functionality, reliability and maintainability.
- *Review of Software Architecture Document against Requirements Specification Document.* Objective: The base architecture designed should support the remaining use cases not described in detail. Quality characteristic: functionality, reliability and maintainability.
- *Architecture Stress Testing*. Objective: The architecture should fulfil the system requirements and support the application's most critical functionalities. Quality characteristic: functionality and reliability.
- *System Compatibility Tests.* Objective: Evaluate system compatibility with external systems. Quality characteristic: functionality, reliability and interoperability.
- *Review of the Creative Design Summary*. Objective: Check for compliance and traceability of visual options in the prototype with the Creative Design Summary. Quality characteristic: usability and accessibility.
- *Review of Look and Feel.* Objective: Determine the visual option that best suits the client's taste and expectations. Quality characteristic: usability and accessibility.

3.1.3 Construction Phase

During this phase, all components and functionalities of the application are developed, tested and integrated into a product. This phase consist of the construction process where emphasis should be made in resources' management and cost, agenda and quality control. This is the longest phase, and at the end of each iteration a product version is obtained (e.g.: alpha, beta or deliverable). This is a highly iterative phase, the main purpose of which is to produce valuable elements for the client. It is also aimed at reducing development costs through resources' optimization, while avoiding unnecessary time losses. Table 4 provides an overview of the activities performed in this phase. The artefacts generated in these activities should later be formally revised as follows:

Construction Phase Quality Verification. The following diagrams and models are evaluated using the same techniques than described in the previous phase: Behaviour diagram, class diagram, navigation map, and E-R diagram. Additional techniques are applied as follows:

- *Proposed Use Cases Inspection*. Objective: Check for correctness of use case models and specifications. Quality characteristic: functionality, reliability, maintainability.
- *State Machine Diagrams' Inspection*. Objective: Check for status diagrams' correctness. Quality characteristic: functionality, reliability, maintainability.
- *UX Models and Storyboards Inspection*. Objective: Check for correctness of models and storyboards. Quality characteristic: usability, functionality.
- *Review of the Creative Design Document*. Objective: Check that the context graphic interface prototype takes into account the style and other aspects described in the Creative Design Document. Quality characteristic: usability.
- *Code Walkthroughs or Direct Inspections*. Objective: Check for right code. Quality characteristic: functionality, reliability and maintainability.
- *Integrity Tests (gray-box), Task-oriented Functional and Exploratory Testing*. Objective: Check for functionality of the context implemented. Quality characteristic: functionality, reliability, usability and accessibility.
- *Incremental Integration Tests*. Objective: Check for proper integration of context elements. Quality characteristic: functionality and reliability.
- *Incremental Integration, Compatibility, and Configuration tests*. Objective: Check for proper integration of the context with other contexts already implemented. Quality characteristic: functionality and reliability.

3.1.4 Transition Phase

The purpose of this phase is to successfully deploy the system. Some amendments or new versions of the system may arise, which require the development of new releases, correction of issues, and inclusion of final features that were previously postponed. Table 5 provides an overview of the activities performed in this phase. The artefacts generated in these activities should later be formally revised as follows:

Transition Phase Quality Verification

- *Complete System Stress and Resources Testing*. Objective: Check the complete system to determine that its limits satisfy the project's expectations. Quality characteristic: functionality and reliability.
- *Functionality Testing.* Objective: Evaluate the complete system functionality. Quality characteristic: functionality.
- *Security and Warranty Testing*. Objective: Evaluate complete system security. Quality characteristic: reliability and functionality.
- *Gray-box Testing*. Objective: Overall system evaluation. Quality characteristic: functionality, reliability and usability.
- *Interoperability and Configuration Tests.* Objective: Check for proper operation of the system in different browsers. Quality characteristic: functionality, reliability, usability and compatibility.
- *On-line Help Testing.* Objective: Check for proper operation of the on-line help content. Quality characteristic: functionality, usability and accessibility.

Table 4: Products' Construction Phase.

Activity	Description / Product / Roles
Context design	Update the SAD in accordance with use cases, by adding detailed use cases, and context sequences diagrams and state machines as they are prepared. Artefacts: Improved Software Architecture Document. Roles: Software Architect and Analyst.
Context graphic interface design	Detail the interface graphic elements and usability related to the development context. First of all, the context interface document is generated and, then a prototype incorporating the new papers related to the context is generated. This design activity can be performed in parallel with the SAD update. Artefacts: Context Graphic Interface Document and Context Interface Prototype. Roles: Usability Manager and Interface Designer.
Context implementation	Once the context design and interface analysis are performed, we proceed with the implementation or adaptation of the elements identified. Artefacts: Implemented Context. Roles: Programmer, Interface Designer and Tester.
New context integration	The new context is integrated to the rest of the application. This integration is made by layer, beginning with the data layers and ending with the presentation layer. Artefacts: Integrated context. Roles: Programmer and Tester.
Phase final meeting	Analyze the feasibility of continuing with the project. If the project fails in the following items, it should be cancelled or reanalyzed. Artefacts: plan updated for the next iteration. Roles: all stakeholders.

Table 5: Products' Transition Phase.

Activity	Description / Product / Roles
Application deployment	Stabilization of the final solution in order to transfer the system from the development environment to the production environment. Artefacts: Application implemented. Roles: Programmer.
Users' training	However, if the application's final users are properly defined and accessible a training process should be considered. Artefacts: Users' manuals. Roles: Usability Manager.
Phase final meeting	Analyze the feasibility of releasing the project. If the project fails, it should be cancelled or reanalyzed. Artefacts: updated plan. Roles: all stakeholders.

- *Online Help Content Review*. Objective: Check that online help contents are understandable. Quality characteristic: functionality, usability and accessibility.
- *Users' Manual Inspection.* Objective: Check for documents with accurate and understandable information for the system target users. Quality characteristic: usability and accessibility.

As final recommendation, we suggest to distribute the development effort as follows: definition phase (10%), architecture baseline phase (10%), construction phase (50%) and transition phase (30%).

4 EVALUATION

We validated this methodology through its application on a specific organization. The evaluation method included two phases: 1) Activities for developing a system using COWM, and 2) COWM evaluation through certain proposed features.

4.1 Case Study

The case study consists of a system that provides support to a company (which for privacy reasons, shall be called *LearnEnglish*). Its main purpose is to provide an e-learning solution to those internet users that want to improve their English with focus on oral expression. Given that it is an e-learning application, the technological component is the central axis of the process. Considering that the development of the whole support system and multimedia material would take a long time, we decided to focus on the development of the *LEnglishAdmin* sub-project which supports all operations related to new users' recording, access control, and class reservations, among others.

4.2 Feature Analysis

Figure 1 shows features that we evaluated after applying COWM to the case study. These features were proposed based on Callaos (1992), Whitten et al. (2004), Krutchten (2004), Cockburn (1998) y WCAG (1999).

While general features evaluate the quality of the proposed methodology to describe and apply methodology–oriented concepts, specific features evaluate the presence and application of the following components: basic *Web* aspects; requirements of individuals with disabilities (*Accessibility*); short development *times; adaptability* to any type of project, technique, method or tool and support to requirements changes; satisfaction of the most relevant Web *quality* aspects (functionality, reliability, maintainability and usability). Upon completion of the case study review and application of an instrument to measure the aforementioned features, we observed that such features were deemed acceptable within the evaluation context, which corresponded to acceptance levels above 80%.

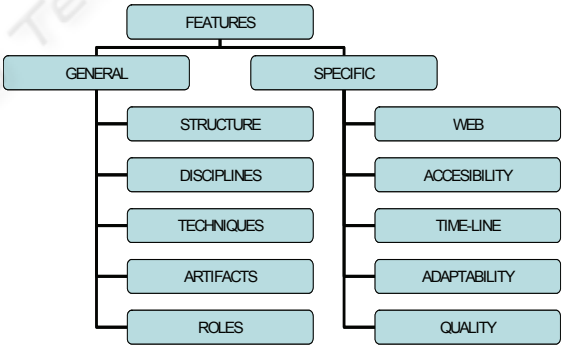

Figure 1: Evaluated features.

5 CONCLUSIONS

COWM has been conceived as a methodology that allows solving deficiencies in the analyzed methodologies through the combination of the advantages of both, agile and plan-driven methodologies and the inclusion of activities, roles and artefacts which facilitate to manage the particular characteristics of the Web applications. As a consequence, COWM pre-

sents a process which uses contexts to rapidly generate products being valuable for users without omitting neither documentation nor, QA activities needed for guaranteeing a high-quality Web system. Additionally, at each phase, COWM includes techniques to ensure quality of each artefact, thus reducing the project critical risks and guaranteeing successful project completion.

ACKNOWLEDGEMENTS

This research was supported by National Fund of Science, Technology and Innovation, Venezuela, under contract S1-2005000165. Authors also want to thank to Miurika Valery and Wilmer Sarmiento for their valuable contribution to this research.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., Juhani, W. (2002). Agile software development methods. Review and analysis. Espoo. VTT Publications.

Ambler, S., Nalbone, J., Vizdos, M. (2005). The Enterprise Unified Process: Extending the Rational Unified Process. Prentice Hall.

Barry, C., Lang, M. (2001). A Survey of Multimedia and Web Development Techniques and Methodology Usage. IEEE Multimedia, 2-10.

Boehm, B., Turner, R. (2004). Balancing Agility and Discipline: A Guide for the Perplexed. Pearson Education, Inc.

Callaos, N. (1992). A Systemic 'Systems methodology', 6º International Conference on Systems Research Informatics and Cybernetics, International Institute of Advanced Studies in Systems Research and Cybernetics and Society for Applied Systems Research, Florida.

Canós, J., Letelier, P., Penadés, M. (2004). Metodologías ágiles en el desarrollo de software. Universidad Politécnica de Valencia.

Coad P., Lefebvre E., De Luca J. (1999). Java Modelling In Colour With UML: Enterprise Components and Process. Prentice Hall.

Cockburn, A. (1998). Surviving Object-Oriented Projects. Addison-Wesley.

Ginige, A., Murugesan, S. (2001). "Web engineering: A methodology for developing scalable, maintainable Web applications". Cutter IT Journal, 14(7), 24-35.

Heumann, J. (2003). User experience storyboards: Building better UIs with RUP, UML, and use cases. The Rational Edge.

IEEE 1012. (1998). "Standard for Software Verification and Validation". Software Engineering Standards Committee.

ISO/IEC 9126-1. (2001). "Software engineering — Product quality — Part 1: Quality Model".

Kan, S. (2002). Metrics and Models in Software Quality Engineering, Second edition. Boston-EEUU: Addison Wesley.

Kroll, P., Kruchten, P. (2003). The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. Addison-Wesley.

Kruchten, P. (2003). The Rational Unified Process An Introduction. Third Edition. Addison-Wesley.

Kruchten, P. (2004). The Rational Unified Process – An Introduction. Addison Wesley.

Letelier, P., Penadés, M. (2004). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Universidad Politécnica de Valencia.

Lott, C. (1997). "Breathing new life into the waterfall model". Morristown, NJ. Bellcore.

Lowe, D., Henderson-Sellers, B. (2001). Infrastructure for e-business, e-education, and e-science. SSGRR.

Montilva, J., Barrios, J. (2002). A Component-Based Method for Developing Web Applications. Universidad de Los Andes.

Offutt, J. (2002). Quality Attributes of Web Software Applications. *IEEE Software: Special Issue on Software Engineering of Internet Software*, pp 25-32, Marzo/Abril 2002.

Reynoso, C. (2004). Métodos Heterodoxos en Desarrollo de Software. http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.asp

Schwaber, K. (1995). The Scrum Development Process. http://www.controlchaos.com/old-site/scrumwp.htm. OOPSLA'95 Workshop on Business Object Design and Implementation.

Schwaber K., Beedle M., Martin R.C. (2001). Agile Software Development with SCRUM. Prentice Hall.

Whitten, L., Bentley, L., Dittman, K. (2004). System Analysis and Design Methods. McGraw-Hill.

WCAG (1999). Web Content Accessibility Guidelines 1.0. W3C.

Woojong, S. (2005). Web Engineering: Principles and Techniques. Idea Group Inc.

Wu, Y., Offutt, J. (2002). "Modelling and Testing Web-based Applications". George Mason University: Information and Software Engineering Department.