Critical Feature Method—A Lightweight Web Maintenance Methodology for SMEs

Xiaoying Kong, Faculty of Engineering, University of Technology, Sydney, xiaoying.kong@uts.edu.au
Li Liu, Project Management Outreach Programme, The University of Sydney, l.liu@pmoutreach.usyd.edu.au

Abstract

Security threats, dynamic business environment and the ambiguous multi-jurisdictional arrangements governing internet businesses often impose urgent change demands on businesses operating on the internet. A lightweight web maintenance methodology for SMEs has been proposed in this paper to effectively handle emergency situations that are common to web systems. The web maintenance states are analyzed and classified into three distinct states. The proposed process can be integrated with the normal evolution of web systems. The methodology is underpinned by a set of core values that emphasise collaboration between chief developer and the business executives; minimal feedback loop; close involvement of business executives; and rapid design focusing on critical features. Two rapid prototyping approaches have proposed. The Critical Feature Matrix and Normal Feature Matrix are introduced to replace the onerous conventional documentation.

Keywords

Web development, maintenance, methodology, process model, prototyping, matrix

Introduction

The global reach and relatively low development cost of websites, enabled by advances on Web technologies, makes it attractive for organizations to use websites to conduct businesses that they would not have been able to do without a website. As a result, websites are increasingly becoming an important platform for businesses, especially small and medium sized enterprises (SMEs). For a SME, it used to be a very expensive option, if not an impossible one, to explore overseas markets through conventional marketing channels. With a website, a SME could easily reach global markets through its website. There is evidence suggesting that web site is one of the best return on investment (ROI) results of all internet applications for SMEs (Telstra 2003). In 2002, 36% of small businesses (defined as business employing 1-19 people) and 82% medium businesses (employing 20-199 people) in Australia use their own web sites for internet activities such as promotion for their businesses, procurement, sell, tracking customer orders, providing a customer information database and email marketing over year 2002 (Telstra 2003).

However, using website to conduct business is not without caveat. Security threat from cyber theft and attach has been well documented. For example, Yahoo and a dozen other prominent commercial websites received denial-of-service attack recently. In early August 2003, thousands of local networks and PCs around the globe have been paralyzed by a virus called W32.Blaster.Worm. Identify fraud and cyber crimes are also on the increase. In one case, a perpetrator has stolen records of credit card details by breaking into the server of an Australian company. Lack of agreement on the jurisdiction over the internet also poses threat to business conducting business over the internet. For example, promotion over the internet could be seen as promoting to everyone who could potentially visit the website. As a result, relevant laws from every country in the world could potentially be applicable and could be prosecuted by other countries. It is impossible for SMEs to canvas the laws of all countries before conducting business over the internet. What is important to the SMEs transacting on the internet is that they should have the capability to make rapid changes to the website in response to a security threat or upon requests from foreign jurisdictions while maintaining the functionality of the website.

Despite the importance of web sites to SMEs and the need for rapid response capabilities, most existing methodologies are not suited for making rapid changes to an existing system while maintaining the functioning of the existing system. This paper proposes a lightweight methodology that can be used for the maintenance during emergency situation, recovery post emergency and normal development cycle of a system. The methodology is based on the experience of the authors in managing a web system for a small internet company. Its application can be extended to other types of systems. Below, existing methodologies are reviewed first and then the new methodology proposed.

Web Systems and Existing Methodologies

Research has shown that the nature of web system is very different from conventional software systems (Lowe 2000b, Lowe et al. 2001c, Lowe et al. 2002a). At technical level, web systems typically have tighter linkage between business model and architecture; have more pronounced open and modularized architectures; use technologies that change rapidly; demand effective information design and content management; place more emphasis on user interface; are susceptible to security threats; and, place increased importance on quality attributes in mission critical applications that are directly accessed by external users. At organizational level, web systems typically face high level of client uncertainty of their needs and also in understanding whether a design will satisfy their needs; have high levels of requirement, project scope and focus change due to the evolution of business model, have shorter delivery time; demand fine-grained evolution and maintenance with an ongoing process of content updating, editorial changes and interface tuning (Lowe et al. 2001c) and are subject to the laws of multiple jurisdictions. Because of these differences, the processes for the development and maintenance of web systems need to be able to cope with high levels of requirement uncertainty and to make changes very rapidly at very short notices.

Conventional software development models such as waterfall, prototyping, Rapid Application Development (RAD), incremental model, spiral model, component assembly, concurrent development and NASA model (Eljabiri et al. 2001, Pressman 2000, Sommerville 2001, Behforooz 1996) are originally designed for developing stand alone software systems. These models specify the activities, outputs and artifacts at different life cycle phases. They have been proven to be successful with domains that have stable requirements, but have difficulties dealing with new requirements of today's projects that have high levels of requirement uncertainty (Eckstein 2003) which is a common feature of web site development and maintenance (Lowe et al. 2003b).

To effectively cope with high levels of requirement uncertainty and frequent change, more and more practitioners start to adopt iterative and agile methodologies in web development and maintenance. These methodologies include Rational Unified Process (Kruchten 2000), eXtreme Programming (XP) (Beck 2000, Wallace et al. 2003), Cockburn's Crystal Family, Adaptive Software Development, Scrum (Schwaber et al. 2001), Feature Driven Development (Palmer et al. 2002), Dynamic System Development Method (Stapleton 1997) and Lean Development (Poppendieck et al. 2003). Researchers have proposed new web development models such as Web OPEN (Haire et al. 2001) and WebMutuality (Lowe et al. 2003c). Common features of these methodologies include emphasis on communication between client and developers, minimal documentation, iterative development and rapid feedback. These methodologies, however, are designed primarily for systems development under requirement uncertainty but not well suited for making changes in an emergency situation.

For example, there are urgent unforeseen circumstances that require changes be made to the existing web system at the shortest time possible. These demands could be due to security threats or unforeseen legal requirements imposed by overseas jurisdictions. Under such circumstances, the priority is to prevent the threat and comply with the legal requirements first even at the cost losing some existing functionalities. The lost functionalities can be restored post the emergency period after further systems maintenance. The normal development cycles start after the lost functions have been restored. This is the scenario happened to an Australia company that specializes on matching investors and businesses seeking investment capital. For confidentiality reasons, we use a fictitious name for the company "Cap-Match".

Cap-Match specializes in matching potential investors and capital seekers by posting the investment proposals of the seekers on its website for a fee. Seeking capital in Australia is regulated by the corporations law of Australia. In addition, the Australian Securities and Investment Commission (ASIC) issued a number of exemptions for raising small capital from rich individuals. Cap-Match relied on ASIC's exemptions for its operation. At the time between 1995-2000, there were no clear rules as to the jurisdiction over online businesses. Suddenly in 2000, the company received a fax from the Securities and Exchange Commission (SEC) of the United States of America alleging the company was breach of the laws governing the capital raising in USA. At that time, the US government decided to take a more proactive stance to the capital raising activities on the internet. In effect, the SEC deems all the websites conducting capital raising activities that can be viewed by American residents as conducting capital raising activities in the USA and therefore subject to the laws of the USA. The fax outlined the laws governing capital raising in the USA and demanded that changes be made to the web system within ten days so that the operations of Cap-Match comply with the US laws in relation to capital raising. Cap-Match faces two options, to completely ignore SEC's notice or make changes to comply with SEC demands within a very short period of time. The first option could be a very expensive option if SEC decides to prosecute the company. Cap-Match decided to adopt the latter option. After reviewing the relevant legislations, the company came up with a set of changes to achieve the critical features that will make the operations comply with the US laws. Nevertheless, due to the very limited time, it is not possible to fully assess the impact of the changes to the whole system. It was clearly, however, some functionalities will have to be disabled for the period until the impact of the changes on the system has been fully tested. After ten days' intense effort, Cap-Match had completed changes to its web system to comply with the relevant laws in the USA. Contacts were made with the person from SEC in charge of the matter who informed the company the changes did appear to have satisfied their demands.

Having satisfied SEC, the company then spent another few weeks on restoring key functions that were disabled when making the changes. Patches were developed and tested on development server to make sure the functions are restored without compromising other functionalities. Once the key functionalities have been restored, the web system is back on the normal development and maintenance cycle. The only difference from the cycles before the changes is that the new development and maintenance need to be developed and tested against the changed system.

The above case demonstrates that the existing methodologies are inadequate for dealing with situations where urgent changes need to be made with the possibility of compromising existing functionalities. This paper hereby proposes a lightweight methodology that deals with emergency maintenance and integrates the functionality restoration period after the emergency period.

Critical Feature Method (CFM)

We begin by introducing the core values of the method. The principles and practices that underpin this methodology are described.

Core Values

The core values of this lightweight method are

- Communication between the business executives and the developers
- Minimal cost in communication and development
- Rapid development
- Simple documentation

To support the values, the following practices are applied:

- Partition maintenance to three states to manage emergency maintenance
- Close collaboration between the business executives and the developers
- Merge business modeling, requirements gathering and analysis, and design into one pair-prototyping phase
- Agile design process using paper prototyping and telephone prototyping
- Replace formal documentation with lightweight matrices

Fire fighting in an emergency situation, there is no time to follow hefty documentation requirements of the traditional methodologies. Instead, the focus should be on developing workable solutions with minimum disruption to the existing functionalities of the web system within tight time constraints.

Conventional software development processes typically divide key roles into client/users management, project management, systems analysis, systems design, programming, testing, deployment, quality assurance, document writing and configuration management. These roles are in turn assigned to project team members. Each member may be assigned one or multiple roles. Clear role definition helps to set the context for communication among team members and the coordination of tasks. Any change request needs to follow a procedure and needs to be signed off by people assigned with relevant roles such as testing and configuration management. These people in turn need to conduct reviews on the proposed changes. This rigorous change control process is necessary to maintain the quality of a system. However, when all the key roles are vested in a few people, such as the business owner and the chief developer in a SME, such a process becomes onerous because two persons could decide on the solutions or changes through face-to-face meeting, tele conference, paper prototyping and telephone prototyping (introduced below) without resorting to such process. The division of roles and document-based communication unnecessarily increases the cost of change in such a situation.

The lightweight process proposed here emphasizes collaboration between the chief developer and the business executives; minimal feedback loop; close involvement of business executives; and rapid design focusing on critical features. Some of these core values are borrowed from existing methodologies. For example, eXtreme Programming utilizes an on-site customer to minimize the feedback loop in development, Joint Application Development (JAD) and Participatory Design (PD) stress greater user involvement and user participation in the development of systems (Bjerknes et al.1987, August 1991, Carmel et al. 1993). Here, because of the extreme time pressure, there is no time for user feedback, instead, business executives who understand users' needs are seen as surrogates for the users.

Agile methodologies (Agile Alliance 2001, Martin 2002) and WebMutuality (Lowe 2003c) highlight the need for clients and developers to work closely together throughout the project. Such close collaboration is based on the premise that neither the clients nor the developers know clearly about user/client needs at the beginning. Close collaboration not only helps the developers developing features that satisfy business needs, but also helps the clients to change the business domain (Lowe et al. 2003b, Lowe 2003c). As suggested by MacCormack (2001), "daily incorporation of new software code and rapid feedback on design changes" and "a team with broad-based experience of shipping multiple projects" are critical to delivering software projects at internet time.

The nature of SMEs makes the collaboration between the chief developer and the business executives easier. They are often co-located and work closely on daily basis. In addition to scheduled meetings, they could easily reach each other through informal means outside office hours which is critical during an emergency. Such close inter relationship could be utilized fully to reduce feedback time lag, increase interaction between the two and ultimately develop solutions to the emergency rapidly.

In this lightweight web maintenance methodology, project roles for SMEs are assumed to be vested in two per persons, a business executive and a chief developer. The business executive runs the business and understands user needs. The chief developer is in charge of the web system who looks after technical issues and usually works closely with the business executive. In many cases, the chief developer is also one of the founders of the business and therefore knows the business well. The chief developer and the business executive usually have frequent meetings at informal occasions such as tea break, lunch, home or parties. During these meetings, new business requirements and designs of the system can be discussed and debated between the business executive and the developers. These informal meetings are invaluable to resolving the emergency situation for a SME because both the business executive and the chief developer still have routine tasks and obligations that can not be fully set aside for this emergency situation only.

Three Maintenance States

As demonstrated in the Cap-Match case, the maintenance process can be partitioned into three distinct states (See Figure 1): the Emergency Maintenance State, the Post Emergency State and the Normal Evolution State.

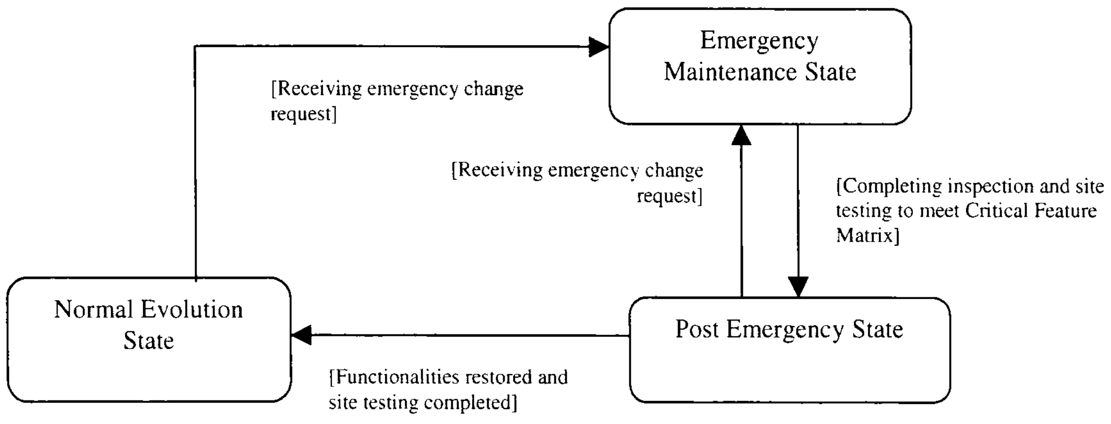
Emergency Maintenance State
[Receiving emergency change request]
[Receiving emergency change request]
[Completing inspection and site testing to meet Critical Feature Matrix]
Normal Evolution State
Post Emergency State
[Functionalities restored and site testing completed]

Figure 1. State Diagram for Critical Feature Method

Normal Evolution State
A web site needs to be constantly updated and modified to suit business changes. This natural evolution state has been likened to web gardening (Lowe 2000a). Most of the time, a web site is in this routine evolution state - Normal Evolution State. In this state, the team could conduct the following activities evolutionarily to the existing live web system:

- Content management and archiving live web system.
- Corrective maintenance: correct pre-existing errors in the exploration, design and implementation in all three states.
- Adaptive maintenance: modify the system to adapt to environment changes such as changes in hardware, platform, language, database or upgrading to other new technologies.
- Perfective maintenance and refactoring: add or modify features in response to business changes and developers suggestion, refactoring the code.
- Reengineering: replan, redesign, rebuild the entire web system in response to changes in business, technologies, system architecture erosion and the creativity of developers.

Emergency Maintenance State
Once an emergency situation arises, the Emergency Maintenance State starts. All non-essential activities in other states are suspended immediately to give way to the emergency. Because of the tight coupling between the web system and the business domain, the business executives work closely with the chief developer to work out critical features that are needed to deal with the emergency. This communication should be done in a timely fashion without resorting to much formal documentation.

The time constraint is extremely tight in the emergency situation. The highest priority in this state is to accomplish the critical features first. The changes should be as simple as possible as long as the critical features can be achieved. Reactive and unstructured maintenance are allowed which may impact on other functionalities and therefore affect the quality of the system. Inspection and testing could be limited to the features that have been changed in this situation and those critical to the business. A "Critical Feature Matrix" has been proposed to identify and track the features to be tested in this state. The emergency changes should be recorded in a simple way such as hand writing. Formal documentation is not recommended in this state.

Emergency is by nature unpredictable and therefore could happen at any time. The critical activities during an emergency include making decision on the business changes, working out corresponding critical features for the systems, implement the critical features, inspect and test, and deploy the changed system online. During an emergency, reactive fire fighting replaces forward planning, timeboxing replaces schedule estimation. The critical features should be met as far as possible within timebox constraints. Some non-critical features of the system may need to be disabled temporarily or abandoned for the critical features to be implemented.

Post Emergency State
Once the emergency has been rectified, the maintenance enters the Post Emergency State. The main objective during this state is to restore the functionalities disabled or lost during the emergency state.

The web system is thoroughly tested to identify disabled or lost functionalities. The disabled or lost functionalities that are necessary for the business will be restored through further patching. Documentation should be kept to a minimum in this state. Configuration management may be necessary depending upon the size of the project. Once the functionalities of the system are back to normal, the system goes back to the Normal Evolution State.

Pair-prototyping for rapid exploration and design

Web systems are constantly changed to adapt to user needs and business changes. This month's web system is next month's legacy system. As discussed above, effective collaboration between the business executive and the chief developer enables rapid and simple design that could accelerate the pace of developing effective solutions in emergency. Such collaboration between the business executive and the chief developer facilitates the developer's understanding of business processes and the executive's understanding of the web system and its fit with desired business model. Research indicates that the web process is more design-driven. Design artifacts play a crucial role in the development of customers' understanding of their needs (Lowe et al. 2002b). As suggested by recent research, business modeling, requirement analysis and design are in one "exploration phase" in a typical web development process (Lowe et al. 2002b, Lowe 2003c).

This methodology proposes a rapid design process which explores technical design and business solutions through pair-prototyping approach. The design decision is a decision for both web system design and new business features and solution. The close collaborating relationship between the business executive and the chief developer is fully leveraged here to facilitate this rapid design process. The rapid prototyping include paper prototyping (Snyder, 2003) and telephone prototyping both of which can be applied in either the Emergency Maintenance State and the Post Emergency State. The two prototyping approaches can also be used in the Normal Evolution State in conjunction with white site prototyping and other design methods.

Paper prototyping

Paper prototyping could be a face to face design approach with three states when the business executive is with the developers. Color paper sheets are cut into pieces to mimic web objects. Site architecture is designed by placing the paper pieces on a large paper sheet. Navigational model is designed by organizing the sequences of pages using these paper pieces. User interfaces are prototyped by arranging the color pieces into a screen-size paper sheet. Data structures are discussed in paper or storyboard. Procedural design is sketched on paper or whiteboard using simple diagrams that both the business executive and the chief developer understand such as flow chart or sequence diagram. Design alternatives are rapidly debated and chosen when manually arranging these paper pieces. Business rules raised by the business executive are analyzed and added into the design. The web system features are explored during the prototyping and debate. The prototyping solutions help the business executive to decide the system need. Usability data are gathered in this process. Once the decision is made, the color paper pieces are glued to large pieces of paper sheets. The identified web system features are manually recorded on paper or whiteboard. They will be recorded into a Critical Feature Matrix later.

The artifacts of this process could include paper prototypes of interfaces, paper navigational model, data design on paper sheet, sketched architectural model, sketched procedural design diagrams, web system features of this release upon the current change requests. Not all above activities and artifacts are needed in this prototyping process. The design is only upon the current change requests and critical features.

This process could frequently happen between the business executive and the chief developer after working hours in any place due to their special close relationship. Other low-tech prototyping such as "sand prototyping" and "shell prototyping" could replace the paper prototyping in park or beach in their weekends, holidays or even at their daily three meals and home. This reduces huge amount of cost in communication, technologies and manpower in design process. Hence supports the core values of this maintenance methodology. The business executive gradually learns and understands the technologies and their impacts to the business model in this daily collaboration process. In return, this makes the telephone prototyping possible if an emergency occurs.

Telephone prototyping

In Emergency Maintenance State and Post Emergency State, if the business executive is not approachable via face to face communication, telephone prototyping is an effective, simple and quick design approach. Through years of experience in running the business and collaboration with the chief developer, the executive should already have a mental model of the web system and its business features in his/her memory. In telephone prototyping, brain drawings replace pieces of paper used in the paper prototyping. The chief developer uses examples of templates, existing web objects and design patterns that the business executive is familiar with to prototype the web system via telephone communication with the business executive. The urgent business change requests, braindraw of design upon these requests, the change impacts on existing business process and mental simulation of the business process in the web system are exchanged and debated rapidly via telephone. Design decision is made via telephone. The design artifacts are recorded on paper or whiteboard.

White site prototyping and other design modeling
Developers typically enjoy cutting edge technologies and complex solutions. However, in Emergency Maintenance State and Post Emergency State, developers' love with technologies should give way to simple and easy solutions. New and complex technologies may actually slow down the pace and prolong the emergency state.

In Normal Evolution State, the evolution pace is back to normal when the developers could explore the use of new technologies and add more functions to the system. White site prototyping and other design modeling provide creative approaches to website development. In this state, the developers could have the luxury to explore new technologies. Various design methods and tools such as RMM (Isakowitz 1995), OOHDM (Schwabe 1995), UML (Baumeister et al.1999, Baresi et al. 2001), WebML (Ceri et al. 1995), e3-value (Gordijn et al. 2001) and WebML+ (Lowe et al. 2003a) can be used in this state depending on the nature of the tasks and project team's experience. Whatever the design tools and models, simple design should be the overriding design rule.

Critical Feature Matrix and Normal Feature Matrix replace formal documentation

The process models for conventional software development and maintenance define and recommend some formal documentation that are associated artifacts of each defined activities in software lifecycles. Due to the characteristics of web process, these documents are not suitable for web process. Researchers have been trying to clarify which aspects should be specified, what the documentation is meaningful to both clients and developers and where in the lifecycle should requirements be articulated (Lowe 2000c, Lowe 2001a, Lowe et al. 2001b). Some agile methodologies such as eXtreme programming do not recommend documentation. Throw-away cards of user stories replace formal requirements (Beck 2000, Wallace et al. 2003).

To effectively deal with the emergency situations in web maintenance identified above, here we propose a "Critical Feature Matrix" and a "Normal Feature Matrix" to replace formal documentations such as development plan, requirements specification and testing documentation.

The Critical Feature Matrix contains the critical features and business constraints that the system should meet and should be tested or inspected in a business view. A feature could be informally defined as a system requirement, a user story, a use case or a change request. Each feature could be built within a few days. Urgent change requests are always the top critical features. Other critical features could be from some categories such as user identification, registration and access control; user monitoring; security; content and data source control; work flow control. Considering the system traceability, other supporting contents of the Critical Feature Matrix could include business constraints; not-yet-passed testing or inspection records; related resources; scheduling information; examples; revision history or comments. The selection of above columns in the matrix is flexible. Conventional planning, requirements specification, validation and verification documents are built into one matrix. It is a lightweight, informal, explicit, traceable and dynamic management matrix. Top rows are always related to the current business priorities. The matrix is dynamically updated by the business executives assisted by the chief developer.

Non-critical features will not be included in Critical Feature Matrix and would not be tested in emergency situation. Non-critical features should be included in Normal Feature Matrix to guide the development effort in Post Emergency State and Normal Evolution State. Examples of such non-critical features could include interface and aesthetic features, user input validation and recovery, response time performance, adaptability and extensibility. Technical constraints could be included in Normal Feature Matrix. The structure could be similar to that of Critical Feature Matrix.

Critical Feature Matrix and Normal Feature Matrix could be manually written on the whiteboard or on paper, or written on a card that could be stuck on or removed from the matrices. They could be later recorded once the emergency state has passed. The features could be a short written statement or a simple diagram that could be understood by the team. Although the features in both matrices could be categorized, we do not recommend categorizing activity in this lightweight methodology. In theory, these two matrices together cover the entire features of the web system. The Critical Feature Matrix focuses the attention of the team during the Emergency Maintenance State while the Normal Feature Matrix is used during the Post Emergency State and the Normal Evolution State.

Conclusion and Future Work

A lightweight web maintenance methodology has been proposed in this paper to effectively handle emergency situations that are common to web systems. The web maintenance states are analyzed and classified into three distinct states. The proposed process can be integrated with the normal evolution of web systems. The methodology is underpinned by a set of core values that emphasise collaboration between the chief developer and the business executives; minimal feedback loop; close involvement of business executives; and rapid design focusing on critical features. Two rapid prototyping approaches have proposed. The Critical Feature Matrix and Normal Feature Matrix are introduced as a simple tool to replace the onerous conventional documentation.

This methodology is developed based on the authors experience in web systems development. However, we believe the methodology could be extended to other types of software systems. Further research is needed to enhance the methodology and identify the applicability to other types of systems. Empirical studies on the effectiveness of this approach are needed.

References

Agile Alliance (2001) Principles: The Agile Alliance. URL http://www.agilealliance.org/principles.html Accessed August 8 2003

August, J. H. (1991) Joint Application Design The Group Session Approach to System Design, Englewood Cliffs, NJ: Yourdon Press.

Baumeister, H., Koch, N. and Mandel, L. (1999) Towards a UML extension for hypermedia design. In <<UML>>1999: The Second International Conference on The Unified Modeling Language, pp 614–629, Fort Collins, Colorado, USA, 1999.

Baresi, L., Garzotto, F. and Paolini, P. (2001) Extending UML for modeling web applications. In 34th Hawaii International Conference on System Sciences, Hawaii, USA, 2001.

Beck, K. (2000) Extreme programming explained: embrace change, Addison-Wesley, c2000

Behforooz, A. and Hudson, F.(1996), Software Engineering Fundamentals, Oxford Uni Press, 1996

Bjerknes, G., Ehn, P., and Kyng, M. (1987) Computer and Democracy: A Scandinavian Challenge, Aldershot, Avebury, Great Britain.

Carmel, E., Whitaker, R. D. and Geoge, J. F. (1993). PD and Joint Application Design: A transatlantic Comparison, Communications of ACM, Vol. 36, No. 4, June 1993.

Ceri, S., Fraternali, P. and Bongio, A. (1995) Web modeling language (WebML): a modeling language for designing web sites. In Proceedings of WWW9 Conference, Amsterdam, 1995.

Eckstein, J. (2003) Agile Software Development in the Large, Dorset House, 2003

Eljabiri, O. and Deek, F. (2001) Toward a comprehensive framework for software process modeling evolution, Computer Systems and Applications, ACS/IEEE, International Conference on. 2001, 25-29 June 2001, 488 -491

Gordijn, J. and J. Akkermans, J. (2001) e3-value: Design and evaluation of e-business models. IEEE Intelligent Systems, 16(4): 11–17, 2001.

Haire, B., Henderson-Sellers, B. and Lowe, D. (2001),Supporting web development in the OPEN process: additional tasks. COMPSAC'2001: International Computer Software and Applications Conference, Chicago, Illinois, USA, 2001, IEEE Computer Society Press.

Isakowitz, T., Stohr, E. and Balasubramanian, P. (1995) RMM: A methodology for structured hypermedia design. Communications of the ACM 38 (1995) 34–44

Kruchten, P. (2000) The Rational Unified Process: An Introduction (2nd Edition) Addison-Wesley, 2000

Lowe, D. (2000a) Web Engineering or Web Gardening, Column in WebNet Journal, Vol1, Issue 1, 2000

Lowe, D. (2000b) What makes Web development difficult? Column in WebNet Journal, Vol 1, Issue 2, 2000

Lowe, D. (2000c) Website Evaluation, Column in WebNet Journal, Vol 1, Issue 4, 2000

Lowe, D. (2001a) A Framework for Defining Acceptance Criteria for WebDevelopment Projects. in Murugesan, S. and Deshpande, Y. eds. Web Engineering: Managing Diversity and Complexity of Web Application Development. Springer-Verlag.

Lowe, D. and Eklund, J. (2001b) Development issues in specification of web systems, The sixth Australia workshop on requirements engineering, Sydney, Australia, November, 2001

Lowe, D. and Henderson-Sellers, B. (2001c) Impacts on the development process of differences between web systems and conventional software systems. SSGRR 2001: International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, L'Aquila, Italy, 2001.

Lowe, D. and Henderson-Sellers, B. (2002a) Characterising Web Systems: Merging Information and Functional Architectures, in Nansi, S. ed, Architectural Issues of Web-Enabled Electronic Business, Idea Group Publishing

Lowe, D and Eklund, J. (2002b) Client needs and the design progress in web projects, Journal of web engineering, Vol 1, No.1, 2002

Lowe, D. and Tongrungrojana, R. (2003a) WebML+ in a nutshell: Modelling architectural level information flows. In: WWW2003: 12th International World Wide Web Conference, Budapest, Hungary

Lowe, D., Yusop, N. and Zowghi, D. (2003b) Understanding business impacts of web system prototypes, the Ninth Australian World Wide Web Conference, Gold Coast, Australia, July, 2003

Lowe, D. (2003c) Web development process, WebTech'03, Sydney, Australia, July, 2003

MacCormack, A. (2001) Product-Development Practices That Work: How Internet Companies Build Software, MIT Sloan Management Review, Winter 2001

Martin, R. (2002) Agile Development: Principles, Patterns, and Process, URL http://www.agilealliance.org/, Accessed 8 August 2003

Palmer, S. and Felsing, J. (2002) Practical Guide to Feature-Driven Development, Prentice Hall, 2002

Poppendieck, M. and Poppendieck, T. (2003) Lean Software Development: An Agile Toolkit, Addison-Wesley, 2003

Pressman, R. (2000) Software Engineering - A practitioner's approach, McGraw Hill, 5th Edition.2000

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

Schwabe, D. and Rossi G. (1995) The object-oriented hypermedia design model. Communications of the ACM, 38(8): 45–46, 1995.

Snyder, C. (2003) Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces. Morgan Kaufmann Publishers, 2003.

Sommerville, J. (2001) Software Engineering, Addison Wesley, 6th Edition. 2001

Stapleton, J. (1997) DSDM, Dynamic Systems Development Method: The Method in Practice, Addison-Wesley, 1997

Telstra Corporation Limited (2003) 2003 Yellow Pages E-Business Report, The online experience of small and medium enterprises. July 2003

Wallace, D., Raggett, I. and Aufgang, J. (2003) Extreme Programming for Web Projects, Addison Wesley, 2003

COPYRIGHT