**Title:** Web Development: Addressing Process Differences

**Length:** 3221 (target: 2500-3000 words)

**Contact Author:** A/Prof David Lowe  
**Affiliation:** University of Technology, Sydney  
**Street Address:** No. 1 Broadway, Ultimo  
**City, State:** Sydney, NSW  
**Postal/Zip Code:** 2007  
**Country:** Australia  
**Telephone:** +61 2 9514 2526  
**Fax:** +61 2 9514 2611  
**Email:** david.lowe@uts.edu.au

**Additional Author:** Prof Brian Henderson-Sellers  
**Affiliation:** University of Technology, Sydney  
**Email:** brian@it.uts.edu.au

**Topic Keywords:**  
- Web development  
- OPEN  
- process framework  
- process adaptation
Web Development: Addressing Process Differences
by David Lowe and Brian Henderson-Sellers

It barely needs to be stated that getting Web systems right is important. Not only have they come to play an increasingly important role in supporting critical business functions, but organisations are spending increasing amounts on these systems. A recent International Data Corp report predicted that U.S. expenditure on Web-based initiatives would grow from US$12 billion in 1999 to US$43.6 billion in 2002! And the ramifications of getting these systems wrong can be much worse than for conventional IT systems – they tend in many cases to be the direct interface between an organisation and their customers, business partners, and/or clients. A failure (be it a technical failure or a business failure) in these systems is a failure that is publicly visible, and hence publicly disastrous!

But so what, you may ask! We have been developing mission critical systems for years, and have built up a huge body of expertise in getting IT systems right – at least when we really need to. So what is different? The key difference is that developing Web systems is not like developing conventional IT systems.

YES VIRGINIA, WEB DEVELOPMENT IS DIFFERENT…

We can’t just apply conventional wisdom to the development of Web systems. It is easy to assume that since they are largely software-based (“so what if they have a few extra creative design components?”) and focus on managing information and information flows (“We’ve been doing that for years!”) that there is not really anything new. To take this view is a very quick path to disaster. Certainly, the development is largely conventional IT development, but it overlooks some of the fundamental differences that make Web development unique.

This issue can be explained using a simple example – understanding client requirements in Web projects. Conventional software processes tend to assume that requirements are understood by clients, and simply need to be elicited and analysed. A typical process will be to elicit requirements (which are documented in the User Requirements Definition), and then analyse these requirements to construct a Requirements Specification, iterating to refine the specification as necessary. One characteristic of this paradigm is that it presumes that the client either understands their requirements, or, at the very least, understands the problem that is being addressed. Even when the client is not able to articulate their requirements precisely, they are at least likely to understand their own business needs, and are able to understand whether a given design will address their needs. For example, approaches such as eXtreme Programming (XP – see www.extremeprogramming.org), utilise an iterative or incremental approach to development and use designs in gaining feedback from the client about to support the resolution of requirements.
This is problematic for Web projects, where the rapid technological change, and more importantly the fundamentally changing business models being initiated by Web systems, leads to many clients not only having a poor understanding of their requirements, but also having a poor understanding of the problems being addressed by the new system and its impacts on their business. In these circumstances, simply using a design to clarify whether it addresses the problem will be insufficient, as the problem itself is only poorly understood. Conventional development processes fail to provide support for this level of uncertainty in the system requirements. In effect, conventional software engineering processes see the requirements as preceding and driving the design process. Even where an incremental approach (such as XP), or an iterative approach (involving multiple feedback loops) is adopted the design is viewed as a way of assisting in the identification and validation of requirements, but rarely does it help the client to actually formulate their needs. In Web development, the situation is fundamentally different. The design process not only helps developers and clients articulate the needs, but also helps clients understand the system domain and formulate their understanding of their needs.

So what might processes that address issues such as these look like? A logical place to start is to look at the differences between Web development and conventional IT development, and then, for each difference, consider the implications for the development process. To illustrate this, we describe the recently incorporated support for web development found in the well-known OPEN process framework.

| THE OPEN PROCESS FRAMEWORK |

OPEN (Object-oriented Process, Environment, and Notation) is a process-focussed methodological approach to software-intensive systems development useful for both object-oriented and Component-Based Development (CBD). It is the longest established of the third-generation OO approaches and covers the full development lifecycle, including business and hardware issues as well as software. OPEN was developed and is maintained by the not-for-profit OPEN Consortium, an international group of over 35 methodologists, academics, CASE tool vendors and developers. OPEN was initially created by the merger of earlier methods: MOSES, SOMA, Firesmith, Synthesis and more recently enhanced by state of the art ideas from BON, Ooram, UML etc. It is documented in a series of books and in many journal articles, particularly in the journal JOOP. Appropriate references, and many of the shorter articles are to be found on the OPEN website at http://www.open.org.au.

The unique aspect of OPEN is that it is not merely a process but a configurable family of processes, defined in terms of a metamodel (also known as a process framework: the OPEN Process Framework or OPF). This metamodel contains a number of major elements (Figure 1) - work units (such as activities and tasks); work products; and producers - that can be multiply instantiated. From these instances of the process fragments (stored in the OPF repository), organizationally-specific processes can be readily constructed. The way these elements are put together is also the decision of the organization or development team – thereby supporting the construction of highly customised development processes. The component-based
nature of OPEN permits appropriate extensions to support development in new domains – such as the new extensions for Web development, called Web OPEN – by adding appropriate activities, tasks, techniques, roles, etc. (see Haire et al. (2001) for a more complete treatment of Web OPEN). These extensions were derived primarily from an analysis of the documented differences between web development and conventional development, and validated using case studies of commercial Web development projects. It is worth noting, however, that much of the development remains unchanged. In particular, many of the original (i.e. non web) activities, tasks, techniques and roles in OPEN are still relevant to Web development. For example, the tasks relevant to the activities of Project Initiation, Implementation Planning and Project Planning will remain relatively unchanged. Activities such as requirements engineering and system build will be most affected, since this is where the project domain affects the process most noticeably.

**Figure 1** Some of the key elements in the OPEN Process Framework

**PROCESS SUPPORT FOR WEB DEVELOPMENT**

So – what process components are required to effectively support the Web development process? We will attack this question by looking at specific differences between Web development and conventional development, and then, for each difference, describe appropriate process changes.

**Short delivery timeframes and subsequent fine-grained maintenance**

Web development projects often have delivery schedules that are much shorter than for conventional IT projects. This is partly a consequence of the rapid pace of
technological development, and partly related to the rapid uptake of Web systems. They also support a much finer-grained ongoing maintenance process that is somewhat different from the more conventional maintenance-release cycle. The ability to make changes that are immediately accessible to all users without their intervention means that the nature of the maintenance process changes. Rather than a conventional product maintenance / release cycle, we typically have an ongoing process of content updating, editorial changes, interface tuning, etc. The result is a much more organic evolution.

To support these changes we have the OPEN Activity: Website Management. Website management brings together all the issues regarding the development, maintenance and management of a corporate website. The objectives of the website management activity include creating a high quality website; keeping the website up to date; and ensuring that site standards are met as the website evolves.

The website management activity also involves a number of new OPEN tasks and a number of more general management tasks relate to defining standards and strategies. As part of managing a website, it is important that the acceptance criteria for delivery to the client be clearly established, particularly since websites are such malleable entities (OPEN Task: Define Acceptance Criteria for Website); although there are several other management tasks relating to quality.

**Increased importance of quality attributes**

Web systems represent an increase in mission critical applications that are often directly accessible to external users and customers. Flaws in applications (be they usability, performance or robustness) are therefore likely to be much more visible to business clients and users, and hence will be much more problematic. To address these, supporting appropriate QA mechanisms becomes critical. Quality levels must be established, particularly in terms of how the website will actually be tested against client requirements (OPEN Task: Define website testing strategy). To end users, consistency is highly sought together with confidence that the site is understandable and will function the same way on repeat visits (OPEN Task: Define website standards).

**Highly variable client understanding and rapidly changing technologies**

It is extremely common for clients' understanding of their own needs, as well as the system capabilities, to be extremely poor at the commencement of a project, and to evolve and improve over the duration of the project. This can intensify problems with initial system definition and subsequent “requirements creep” and make developing a product that satisfies user expectations extremely difficult. These problems are also exacerbated by the rapid rate of change in the technology that underpins most web systems. This has several consequences. The first is that it increases the importance of creating flexible solutions that can be updated and migrated to new technologies with minimal effort. For example, the need for reusable data formats (such as XML) increases substantially. A second consequence is that developers’ understanding of these technologies is often restricted, increasing project risks.
To address these issues, several new techniques have been introduced. Two general techniques, the first primarily applicable in web developments but the second more generally useful, are: Development spikes and Field trips. Development spikes are rapidly developed partial solutions addressing a specific requirement and are intended to help users and developers understand the system better rather than being used as part of final solution. They can be thought of as a form of research and serve to minimise risk. Development spikes are particularly useful in web development due to the rapidly changing nature of technology within this field and the poor client understanding of their own needs. Field trips are used to examining the current business environment and final place of deployment of the system. They serve the same purpose as school field trips or field trips in the natural sciences or the engineering professions. By actually visiting a site, a greater overall understanding of the problem is gained. This technique is useful in isolating implied (or assumed) user requirements. It is more effective when coupled with techniques such as user focus groups.

**Open and modularised architectures**

Although not unique to web applications, it is still worth mentioning the emphasis that is typically placed on open and modularised architectures for web systems. They are often constructed from multiple COTS (commercial off-the-shelf) components that are adapted and integrated together. Indeed, strong integration skills become much more critical in most Web projects. The original OPEN framework did not include adequate support for component-based development (CBD) – this was recently added (Henderson-Sellers, 2001). There does, however, exist an extension to the OPEN framework that allows for CBD. The nature of web development projects, and their component-based architecture, means that this extension is also useful when implementing a Web OPEN framework.

Further, many see creating a solid architecture as the most crucial component of a successful Web systems development. In the case of web developments, Web OPEN specifically defines a task called **Design website architecture**.

A number of large organisations have done a significant amount of work in detailing architectural patterns that emerge for different web applications. Some of these initiatives link in closely with their corresponding component frameworks (e.g. the Java 2 Platform, Enterprise Edition Blueprints). Others, however, focus more on the architectural side of the solution and separate this from the implementation layer of development. The most notable of these is the work done by IBM in their “Patterns for e-Business” project (see [www.ibm.com/framework/patterns/](http://www.ibm.com/framework/patterns/)). This all points to the need for choosing an architectural pattern depending on the type of web development. These architectural patterns are really a form of domain modeling in the OPEN framework. To facilitate this choice within the OPEN framework, the Task: **Choose architectural pattern for website** is available.

**Increased emphasis on content**

Web systems are about content! A website that looks fantastic, and has sophisticated and powerful functionality will still be a failure if it does not have adequate and appropriate content, together with mechanisms for ensuring that this
content is both accessible and maintained effectively. Certainly conventional systems rely on their ability to manage data, but with Web systems this data needs to be made easily accessible to users, rather than requiring the users to work to gain access.

This raises the issue, in web development, of the ideas of both content management and information personalization. These both represent functionality that must be included in the majority of web projects today. Since these issues can be considered to play an intricate part in the architecture of the solution, three Tasks are specifically included in Web OPEN. Two of these relate to the planning stage: **Design and implement content management strategy** and **Design and implement personalization strategy**, and one refers to the enactment stage of project management: **Undertake content management**.

Within web projects there is a large amount of rich content that exists as part of the user interface. In traditional software development, the user interface contained a number of various simple controls such as combo boxes and edit bars. The user interface with web development projects consists of almost anything imaginable, from rich text, streaming audio, to even actual applications within the user interface. This content must be carefully prepared much the same way that an editor reviews a newspaper before its final print. The preparation of images, editing and layout of text, and obtaining copyright clearances all must be completed. This is covered in the Web OPEN Task: **Create content (on website)**. In addition, much reuse can be made in creating content. Thus the technique of **Reuse of graphical components** is also available. One particular form of reuse is that provided by **Web templates** which is also a highly useful technique.

**Increased emphasis on user interface**

With conventional software systems users must make an (often considerable) investment in time and effort to install and learn to use an application. With web applications however, users can very quickly switch from one web site to another with minimal effort. As such the need to engage users and provide much more evident satisfaction of users’ needs and achievement of their objectives becomes critical. The result is an increased emphasis on the user interface and its associated functionality.

Beginning with the content – we need to consider how it will be combined with the user interface. This is an ongoing task that must be done to bring together the worlds of print media and software development. On one side, there is a team of creative designers coming up with all sorts of new ideas. On the other side are technical people who must facilitate a method to integrate these ideas with the current navigation, usage and content management of the site. The Web OPEN Task: **Integrate content with user interface** is responsible for combining the content with the method being used to present that content to the user. This task is also important within the Web OPEN framework as it highlights the difficulties that occur when combining two different cultures together within the same project.

This raises the issue of the actual interface design. The user interface within a web project constitutes a large portion of the overall project. It is vital in determining the
success or failure of the project. OPEN has always had a task named “Design user interface”. This task needs somewhat more emphasis for web development projects. While it does not warrant being labelled as an activity under the OPEN framework, it does deserve a number of relevant subtasks. These subtasks have been taken from Constantine and Lockwood’s (1999) work on Usage Centered Design (UCD), which is more appropriate than the significantly different User Centered/Centric Design. Usage-Centered Design focuses on the work that users are trying to accomplish and how the software will support this. It is also important to recall the comments made in the introduction about the role of design within Web development. In particular, design-driven requirement elicitation is significantly different from conventional design. This highlights the significance of UCD, which allows designers to focus on potential patterns of utilisation and therefore helps resolve the uncertainty in the requirements.

The three sub-tasks to the “Design user interface” Task that were introduced in Web OPEN are Create the UCD role model, create the UCD task model, and Create the UCD content model. The last of these subtasks links in well with the Task Integrate content with user interface discussed above as it starts to identify the relationships between the content and the user interface including navigation maps (Task: Create navigation map for website). All three subtasks identify how the site is to be used (hence the name Usage Centered Design) and also help to tie the user interface to the web projects requirements.

<table>
<thead>
<tr>
<th>(Included as side bar)</th>
</tr>
</thead>
</table>

**FULL LIST OF WEB OPEN TASKS**

The following is the full list of activities, tasks and subtasks that have been added to OPEN to specifically address the particular characteristics of Web systems.

**New Activity**
Website Management

**Tasks**
Build White Site
Create content (on website)
Create navigation map for website
Define acceptance criteria for website
Define website testing strategy
Design and Implement content management strategy
Design and Implement personalization strategy
Design website architecture
Design website standards
Develop a brand identity
Develop Data Standard
Integrate Content with User Interface
Prototye the human interface
Undertake content management
Undertake market analysis
Undertake testing of website
Subtasks
Choose Architectural Pattern for website (subtask of “Create a System Architecture”)
Create the UCD role model (subtask of “Design User Interface”)
Create the UCD task model (subtask of “Design User Interface”)
Create the UCD content model (subtask of “Design User Interface”)

Relevant Component-Based Development Tasks
Choose appropriate component framework
Evaluate the potential component frameworks
Integrate Components
Screen the candidate list of component frameworks

CONCLUSIONS

There are various ways in which the development of Web systems (be they e-commerce applications or B2B support for mission critical business processes) differs from more usual IT development. To ignore these differences and adopt a conventional approach to the development is a recipe for disaster. This is not say, of course, that we should throw away the vast body of knowledge and expertise that we have built up – simply that we need to apply and extend it carefully.

We have not described the full details of Web OPEN – to do so is beyond the scope of this article. Rather, we have looked at a range of the specific characteristics of Web systems and their development, and some of the tasks, activities and techniques that were recently added to the development process to address these characteristics. The full set of extensions to OPEN includes numerous other extensions – such as: consideration of the definition of system metaphors and the role they play in ensuring consistency and understandability; reuse of graphical components to improve system performance; the building of white-sites and how they can help garner client feedback, the development of data standards to ensure portability and effective evolution of the site; development of Web metrics to help manage QA aspects; and the development of a brand identity for the site.

Although the IT community is still learning how to manage the development of Web system (something that will be an ongoing process, as the nature of Web systems continues to evolve) we can take some relatively simple steps to improving current practices. Most of what we have described should be relatively obvious. Unfortunately, it is often the obvious that is overlooked.

ACKNOWLEDGEMENTS

The authors wish to thank Brendan Haire for his valuable contributions in developing many of the Web OPEN extensions described in this article.

This is Contribution number 01/09 of the Centre for Object Technology Applications and Research.
REFERENCES


BIOGRAPHICAL SKETCH

Associate Professor David Lowe is the Director of Undergraduate Programs in the Faculty of Engineering, and a Co-Director of the Centre for Object Technology, Applications and Research (COTAR) at the University of Technology, Sydney. He has active research interests in the areas of Web development and technologies, hypermedia, and software engineering. In particular he focuses on Web development processes and web project specification and scoping, and information contextualisation. He has published widely in the area, including several texts (Lowe and Hall, Hypermedia and the Web: An Engineering Approach, Wiley, 1999 and Wilde and Lowe, Transcluding the Web: Linking and XML, Addison-Wesley (currently in preparation)). In the last 7 years he has published over 40 refereed papers and attracted over $900,000 in funding, including a recent grant for research into Web project specifications.

He is on numerous Web conference committees and is the information management theme editor for the Journal of Digital Information. He has undertaken numerous consultancies related to software evaluation, Web development (especially project planning and evaluation) and Web technologies.

A/Prof Lowe can be reached at The University of Technology, Sydney; P.O. Box 123, Broadway, NSW 2007, Australia.

Telephone +61-2-95142526; Email: david.lowe@uts.edu.au.

Professor Brian Henderson-Sellers was the first Director of COTAR (1994-6, 1999-present) and is Professor of Information Systems at the University of Technology, Sydney. His interests are mainly in OO methodologies and process, metrics, project management and company migration to OO. He is involved in several metrics projects, leads the OPEN Consortium and is involved, through the Object Management Group, with the ongoing changes to UML and the new initiative towards a Software Process Engineering Model (SPEM) standard. He has published a significant number of papers and books under the auspices of COTAR as well as
making a large number of international presentations. He is a columnist for the large
circulation OO journal JOOP and a frequently invited speaker at industry
conferences.

Prof Henderson-Sellers can be reached at The University of Technology, Sydney;
P.O. Box 123, Broadway, NSW 2007, Australia.
Telephone +61-2-9514-1687; Email: brian@it.uts.edu.au.