Annals of Software Engineering 13, 163–201, 2002

OPEN Process Support for Web Development

BRIAN HENDERSON-SELLERS
Faculty of Information Technology, University of Technology, Sydney, PO Box 123 Broadway, NSW 2007 Australia

DAVID LOWE and BRENDAN HAIRE david.lowe@uts.edu.au
Faculty of Engineering, University of Technology, Sydney, PO Box 123 Broadway, NSW 2007 Australia

Abstract. We evaluate the efficacy of an established OO/CBD development process (OPEN) in web development and propose new and amended Activities, Tasks, Techniques and Roles that should be included in OPEN in order to fully support the new demands of website construction and the delivery of business value on the web. Sixteen new Tasks are identified together with one new Activity. Four subtasks of particular relevance to the interface based on Usage Centered Design are also advocated. Seven Techniques and ten Roles (some modified rather than new) are also added to the OPEN repository and some example tailoring matrices introduced as examples of process instantiation and tailoring for B2C and B2B.

1. Introduction

Web development often appears not to use, nor to require, any formal process. But increasingly it is being realized that a smart idea, a new `dot.com` company and a blind faith in prototyping and "webtime" development will *not* provide a business environment that can be sustained. For serious software developments to support a successful and growing commercial enterprise, the *same* rigour of business and software development process is required in webtime development as for any other commercial-strength information system.

In this paper, we investigate the applicability of (and required adaptations to) an existing object-oriented (OO) process to the support of web development. The process selected is OPEN; which stands for Object-oriented Process, Environment, and Notation. It is a process-focussed methodological approach to software-intensive systems development useful for both OO and CBD (Component-Based Development) systems development (and, indeed, also useful for business modelling). It is the longest established of the third-generation OO approaches and covers the full lifecycle. 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 (e.g., [Firesmith and Henderson-Sellers 2002; Firesmith *et al.* 1998; Graham *et al.* 1997; Henderson-Sellers *et al.* 1998; Henderson-Sellers and Unhelkar 2000]) and in many journal articles, particularly, in the

journal *JOOP*. Many of these shorter articles are to be found on the OPEN website at `http://www.open.org.au`.

Here we formulate the necessary web extensions for OPEN and, in so doing, create a "dialect" of OPEN to be known as Web OPEN. The extensions are derived primarily from a combination of an analysis of current literature on web development processes and an analysis of two case studies [Haire 2000] undertaken in web-focussed software development companies, one in the commercial domain and one in the standards domain.

2. Web development literature

It is often claimed that web development is inherently different from standard applications software development [Bieber and Isakowitz 1995, Burdman 1999, Overmyer 2000]. Yet web development in its current incarnation goes far beyond the "promotional brochures" and "eye candy" of the first generation of websites and is concomitant with normal software development in a business environment *plus* a number of issues relating to usability (users can rapidly switch to a competitor's site if your website is too arcane), bandwidth (high volume of concurrent users) and graphic artistry (at least in the field of B2C). Web pages are often read in much the same way as brochures, usually scanned for important information and rarely completely read by the user. Web development projects create forms of consumer media with videos, sound clips and sometimes entire movies. In addition to this, there is also the traditional software aspect to web development with websites quite often containing sophisticated back-end systems that help sort, organise and maintain the site. Timescales for website development are also often short and site contents extremely malleable.

Web projects tend to be very visible in nature; systems that face the outside world have no room for error. The consequences of errors and downtime in web systems that interface to customers or suppliers are often major and simply cannot be tolerated. This results in the need for systems and upgrades to be right first time, every time. Possibly even more significant, from a development perspective, is the lack of certainty in the system domain and the volatility in the requirements of the system – which invariably evolve considerably as the system design emerges [Lowe 2000]. Indeed, for many commercial developments, the requirements process can be viewed as design-driven requirements management. In other words, the design process is explicitly used to reduce requirements volatility. This has a fundamental impact on the overall process that is adopted.

2.1. Architecture

The architecture of a web project is extremely important to its long-term success. The architecture typically merges a number of separate aspects. Specifically, it covers both an information architecture and a technical architecture. The information architecture covers aspects such as the underlying content, the way this content is structured and managed, together with a link between the information and the business model that is

being supported. The technical architecture of web systems typically has a thin, highly-customised client front-end, a substantially component-based middleware layer linked together with appropriate "glue" code and a customised back-end that links the system together with legacy systems.

One feature that must be highlighted in the project's architecture is that it must be adaptable. Technology within the web development field is changing so quickly that the architecture must be designed in such a way that it can easily change with technology. The importance of a project's architecture is often overlooked and many people assume that, since the system exists and has been built, then it must have an architecture. This is not true. Recently the re-use of system architectures within the web development community have been gaining support. Large companies such as IBM, Sun, HP and Microsoft have begun long-term projects that deal specifically with web projects and their architectures [Butler 2000].

2.2. Component based development

The software industry is approaching a stage within its development where software packages, called components, can be used to assemble systems, similar to the way you would put electronic components together on a printed circuit board. Generally, some custom coding still needs to be completed in order for the components to interact with each other. These components are continually becoming more advanced and the amount of coding needed is becoming more limited. This component-based development is also very apparent within web development projects. Web developers can assemble applications using a combination of remote services and local services. The nature of such component-based development differs slightly from traditional OO development and so new processes for development must be sought. The adoption of a component-based approach in web development is also reflected in the emerging web design notations such as WebML [Ceri *et al.* 2000] and Conallen's work on the adaptation of the UML [Conallen 2000].

2.3. Content management

Central to the idea of web development is the idea of an information architecture. A key aspect of this is content management. The rate of change within traditional software projects does not compare with the rate of change within today's web projects (particularly change in content). Websites where the content is updated several times an hour are commonly found. Website architectures must be adept at handling this ever-increasing rate of change within systems. Although various content design approaches have been emerging (such as OOHDM [Schwabe and Rossi 1995] and WebML [Ceri *et al.* 2000]) these support only very limited consideration of the way in which the information architecture relates to the technical architecture of a system. Typically, these are developed and represented using quite disparate approaches and the understanding of how to link these is only just beginning to emerge. Without this understanding, it becomes difficult

to ensure that aspects such as the content management system are effectively embedded within the overall systems architecture.

2.4. Interfaces

The interface wars have now entered the web. The level of communication between projects is increasing as the Internet becomes faster and more reliable. Businesses are using the Internet to rid themselves of paper-based processes and to improve their systems. The emergence of a new power player, XML, within the Internet has emphasized the importance of a standardised communication language. The value of data is known and more time is being invested in its ability to adapt. Many organisations have been through the painful and expensive process of converting or interfacing to legacy systems and do not want to repeat the process. This extra attention to the data within a system affects the process used when developing web projects.

2.5. Requirements engineering and high level design

The development of prototypes or white sites seems to be common practice within a number of web development organisations. At the recent Object World 2000 conference in Sydney, a number of companies described their production of white sites in the requirements elicitation phase of development, estimating the requirements phase to represent 15–20% of the total effort expended on a project.

What is important is that the architecture is developed during the requirements phase. This is the solution to the fact that requirements within web development projects are extremely volatile. This can often be because the client is unsure of what can be achieved, changing technology, or any range of situations. This is not a surprising attribute of any project, although it is more apparent in web development. Perhaps this is because the Internet is a relatively new technology, even within the context of software development. The production of a white site, and therefore the high level architecture, solves this problem by leaving the final contract of the systems specification to later in the development cycle. Figure 1 depicts what commonly happens in web development projects. What is interesting about this diagram is that there is no separate design phase that is presented to the customer. Rather it has been broken into two: high level design concerned with the architectural structure of the solution and lower level detailed design concerned with the design of the architectural modules. The first of these two, the high level architectural design, has been incorporated into the requirements elicitation or analysis phase of development. The latter of the two, detailed design, has been moved into the production or build phase of development. This makes the distinction between analysis and design hard to identify within web projects. The majority of analysis techniques used today in web development tend to point towards design decisions. Use cases are considered an analysis tool, yet there are a number of design decisions made while using them, as pointed out by Constantine [1995].

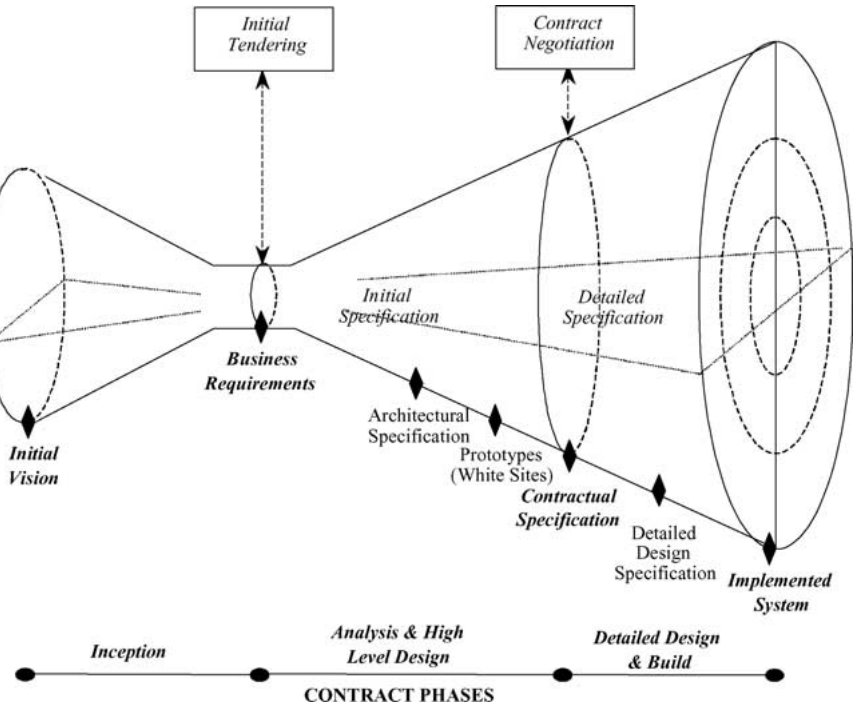

Figure 1. Contractual phases against the development process.

2.6. Web development processes

Although there has been an emerging consideration of specific design approaches for web systems as well as certain aspects of the process such as handling of requirements, the overall process has barely been considered. In particular, there has been little attention given to how conventional approaches need to be adapted to suit web projects. For example, although there has been some consideration given to using lightweight development processes for web development [Angelique 1999; Fournier 1999; Haggard 1998] the consideration of web-specific issues has been superficial. One of the approaches receiving the most attention is the use of XP (eXtreme Programming) [Beck 2000]. When used in conventional software development, it has been claimed that XP is particularly effective for projects that are initially ill-defined – a characteristic of many web projects. As a result, many of the proponents of XP and similar approaches see it is an ideal approach to be adopted for web development. There are, however, certain problems that restrict approaches such as these to web projects. The first is that a number of studies (see, for example, [Martin 2000; Siddiqi 2000]) have shown that approaches such as XP only work effectively for projects that have cohesive development teams – something often not true with web teams due to their multi-disciplinary background which combines both technical and creative design elements. XP can also result in a brittle architecture and poor documentation, which makes ongoing evolution of the system difficult – something that is important for web systems. Finally, and perhaps most fundamentally,

XP utilises partial solutions to resolve uncertainty in requirements but does not inherently handle subsequent changes in these requirements (i.e., requirements volatility) as the system evolves. This creates problems for web systems, where the emerging design results in an evolving client understanding of their needs – and hence volatile requirements [Lowe 2000].

3. A starting point for a web development methodology: OPEN

3.1. OPEN's current architecture

The unique aspect of OPEN is that it is not 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 2) which can be multiply instantiated. From these instances of the process fragments (stored in the OPF repository), organizationally-specific processes can be readily constructed.

Although there are many metaclasses in the OPF, they mostly cluster into five groups: Work Units (with major subtypes of Activity, Task and Technique), Work Products and Producers with support from Stages and Languages (figure 2). Instances of these form a component library for OPEN (figure 3) from which individual instances are selected and put together, constructor set fashion, to create a specific instance of OPEN. This is called process construction. Since the development organization has selected these components, they are readily seen to conform *fully* to that organization's requirements. In addition, the way these elements *are* put together is also the decision of the

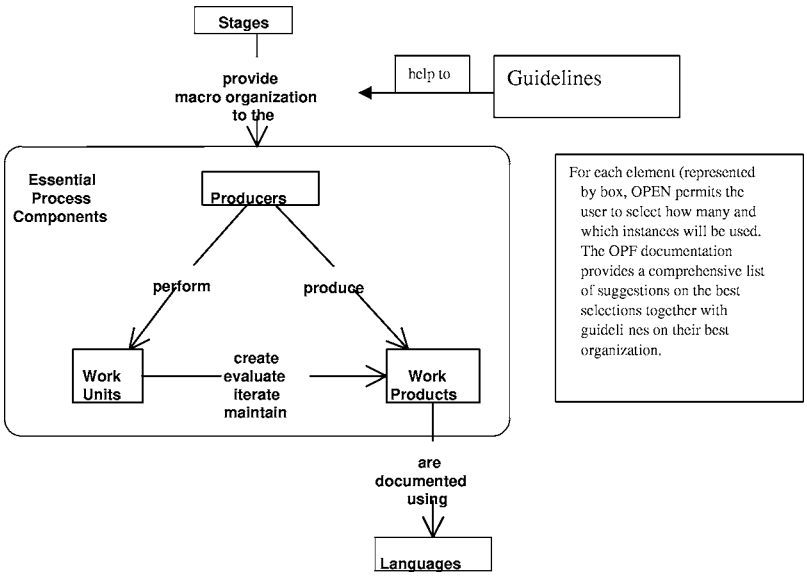

Figure 2. The components of the OPEN process framework (after [Firesmith and Henderson-Sellers 2002]).

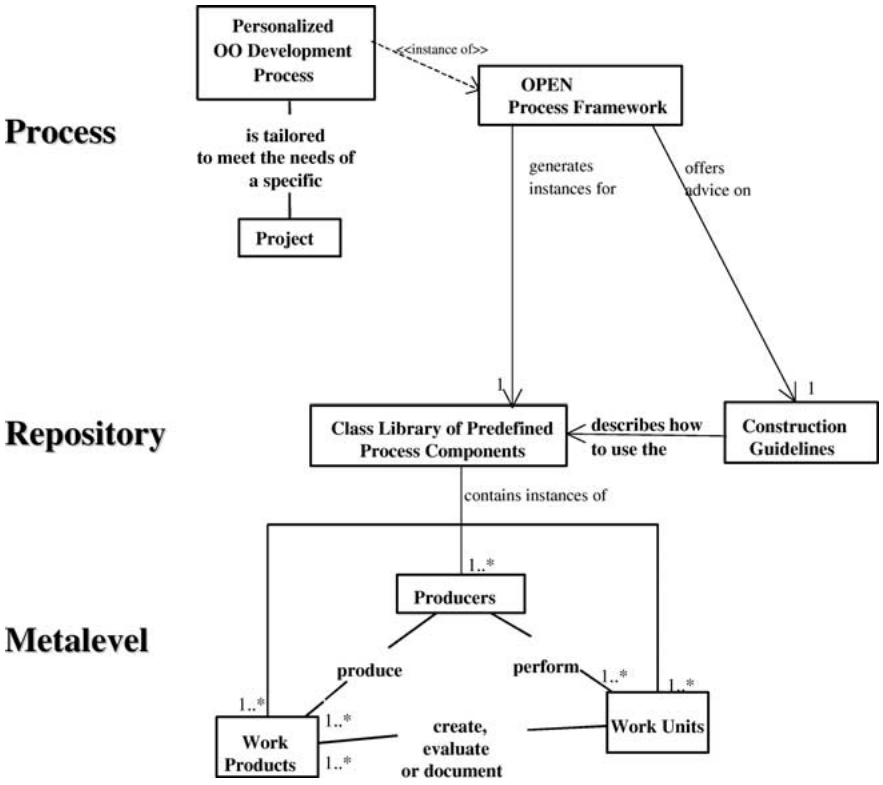

Figure 3. OPEN is a framework defined in terms of a metamodel plus a library or repository containing instances generated from that metamodel (modified from [Firesmith and Henderson-Sellers 2002]).

organization or development team. For example, to adopt and use a fully iterative, incremental and parallel (IIP) lifecycle model, the supplied Assertions are used to sequence elements of the process to express the needs of IIP. On the other hand, if a waterfall approach is preferred, then this too is easily constructed using more stringent constraints on the ordering of what are known as Activities (representing goals to be achieved) within the OPF. Activities, along with Tasks and Techniques, are all kinds of Work Units defined by the OPF. Activities state what needs to be done but not how. Tasks are similarly aimed at the "what" not the "how" but in this case are finer grained. They can be accomplished by a single developer or a small development team in a relatively short time and can be adjudged by the project manager to be complete or not complete. Making these goals (of Tasks and Activities) achievable requires actions of an individual or a team, referred to as a Producer in the metamodel of the OPF. Producers may be human or non-human and may be individuals or teams. They work utilizing one or more Tasks with the Techniques in order to produce the Work Products.

Thus, the focus in OPEN, as shown in figure 2, is the synergistic interaction between Producers (typically people), the things they do (Work Units) and the things they produce (Work Products). For a software development organization, it is generally the

last, Work Products, for which they get paid! – although this would not be possible without the Producers and some knowledge of how to build software (as described in the various Work Units). The OPF defines a useful set of Work Product types, each of which has a number of instantiations in the OPEN library or repository (figure 3). While some are diagrams documented with a modelling language like UML or OML, a large number are textual.

Outside the main trio, Stages and Languages (figure 2) provide additional support. There are various kinds of stages, such as phase, life cycle and milestone, which are used to give largescale organization (often in time) to the development process. On the other hand, languages, be they natural languages, modelling languages or coding languages, are needed as "tools" by which to help document many of the work products. OPEN supports both the UML notation, the OML notation and any other good OO notation of your choice in order to document the work products that the OPEN process produces.

The componentized nature (afforded by the metamodel) of OPEN thus permits the scope of the approach to be extended whenever new technologies arise – or rather, whenever the development context changes, thereby requiring changes to the development approach. Two such examples of changes are the emergence of component-based development and web engineering (although there is in fact significant overlap). Extensions to OPEN to support CBD are given in [Henderson-Sellers 2001]. In this paper, we focus instead on adding support for web-based developments, firstly by asking what support already exists in OPEN and then what is missing and therefore needs to be identified (or created), described and defined for addition into the OPEN framework.

3.2. Existing support in OPEN for web development

We have discussed the differences that exist between web development and traditional software development, but there is also a lot of commonality between the two fields. Therefore, many of the Activities, Tasks and Techniques in the OPEN framework, are still relevant to web development. Here, we evaluate existing, new and modified Activities and Tasks needed to create Web OPEN as a web-enabled "dialect" or instance of OPEN.

If we consider the similarities between regular and web development at the granularity of OPEN's activities, the tasks relevant to the activities of *Project Initiation*, *Implementation Planning* and *Project Planning* will remain relatively unchanged. These activities and tasks are the same for any project. Business approval must be obtained, feasibility studies must be undertaken and other general tasks must be completed. Activities such as Requirements Engineering and Build will be most affected, since this is where the project domain affects the process. Illustrated in figure 4 is OPEN's coverage of lifecycle issues. As OPEN is a full life cycle process model it takes into account business, training and personnel issues. The Activities, Tasks and Techniques associated with these issues may vary but will not be considered in this paper. Rather we will focus on the technical aspects of the process.

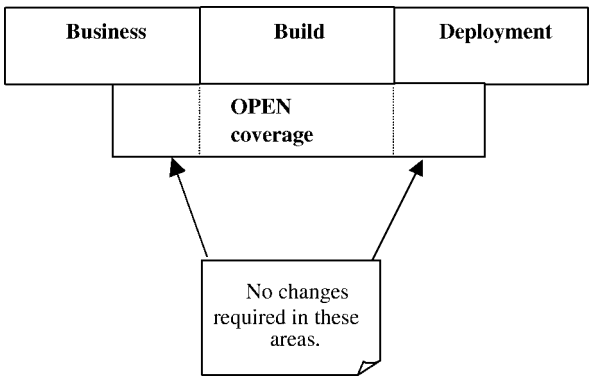

Figure 4. OPEN's coverage of lifecycle issues.

There is no need to create a new set of Activities and Tasks to mimic the ones already in OPEN. Most of the Activities, Tasks and Techniques are generic enough to be used in web development, e.g., Code can be used to signify the coding of objects or the actual writing of HTML pages. Some activities become less critical in web development, or more critical (such as configuration management [Dart 2000]), but the *existence* of the Activities is not affected. Changes could, however, be made to make the Activities, Tasks and Techniques more specific to web development. For example, *Undertake Architectural Design* could be renamed *Undertake Web Architectural Design* for a web environment. However, since this renaming does not really add value to the Task, we consider it to be unnecessary.

The Tasks and Techniques relating to the development of objects will still be useful within the Web OPEN framework. Their importance may be reduced, however, as a component-based development process becomes more prominent.

4. Extending OPEN support for web development

In this section, we outline the various Activities and Tasks (section 4.1) that are proposed as additions and modifications to the OPEN framework to better facilitate web projects. The full details of each of these Activities, Tasks and Techniques are given in the figures. We also discuss affiliated roles (section 4.2). These proposed additions are based on the literature analysis described in section 2 and validated using two case studies [Haire 2000] of web development projects. We analysed the extent to which the process that was followed contained activities, tasks or techniques that would not have been appropriate in a non-web project.

Many web development techniques are isomorphic to those used in regular applications development. Consequently, many of the relevant techniques are not unique to development for the web. Some are generic techniques that were just never included in the OPEN framework, "Checklists" is a good example of this, being added to OPEN more recently [Henderson-Sellers 2001]. The new techniques proposed here for addition to the OPEN framework are: "Branding"; "Development Spikes"; "Field Trip";

"Reuse of Graphical Components"; "System Metaphors"; "Web Metrics"; and "Web Templates". These are discussed in the relevant subsections below.

4.1. New Activity, Tasks and Techniques

In this subsection we discuss the various new activities, tasks and techniques which we have identified (from both the literature and our industry case studies) as being necessary additions to the process component repository of OPEN (figure 3). Each activity/task/technique thus identified is defined in a separate figure following the standard format defined in the OPEN books [Graham *et al.* 1997, Henderson-Sellers *et al.* 1998].

4.1.1. New Activity: Website Management

The whole area of web engineering requires a significant new focus in the form of an activity that is proposed as "Website Management". Website management brings together all the issues regarding the development, maintenance and management of a corporate website which may or may not include access to back-end transaction processing systems. 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 involves a number of new OPEN Tasks, which are introduced in the subsections below. 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 (figure 5), particularly since website are so malleable entities. Quality levels must also be established, particularly in terms of how the website will actually be tested against client requirements (figure 6).

To end users, consistency is highly sought together with confidence that the site is understandable and will function the same way on repeat visits. Thus an important task is "Define website standards" (figure 7).

Focus: Quality. Predetermining criteria for evaluation of website.
Typical supportive techniques: Critical success factors; Envisioning; Acceptance testing; Usability testing.
Explanation: Websites are often created for the client by third party companies. In delivering the completed website to the client, it is important to know on what basis it will or will not be deemed acceptable. This task focuses on agreeing the acceptance criteria early in the creation of the website so that they are unambiguous and easy to interpret. They will also guide the early stages of the site design.
The acceptance criteria will typically describe the underlying business model and outcomes and the key tasks to be supported by the site.
Producers for this task include Project manager, web designer.
Post-conditions for this task might include an agreed set of acceptance criteria.

Figure 5. Define acceptance criteria for website.

Focus: Testing.
Typical supportive techniques: Beta testing; Package and subsystem testing; Regression testing; Unit testing; Usability testing; Web metrics.
Explanation: Unlike most (though certainly not all) conventional software systems, web systems are usually directly accessible to users from outside the client organisation. Indeed they often become the primary interface between the client and their customers. Hence, it is critical that the system be fully operational from initial release and not suffer performance or usability problems. This will often be complicated by the fact that the user base is only poorly known and potentially huge, as well as from diverse social and cultural backgrounds. This means there will be a multitude of assumptions in the users' minds as they use the site; and many unforeseen navigational paths through the site are likely to be realized.
It is therefore critically important to thoroughly test the website before release. The testing must be rigorous and as complete as feasible. To do that, standards for testing must be pre-determined and an overall testing strategy devised in much the same way as such a strategy would be created for non-website developments.
Producers for this task include System tester, Project manager.
Post-conditions for this task might include a defined strategy for testing.

Figure 6. Define website testing strategy.

Focus: Quality and consistency.
Typical supportive techniques: Web metrics; Usability testing.
Explanation: Users like a website that has consistency – consistency within the website and consistency with other websites; consistent user functionality such as they might expect from buttons, scroll bars, etc. both in terms of their location and functionality. Since there are no internationally recognized standards for websites, it is important that a website owner consolidate a set of site-specific standards and *stick with them*. That way users can come to know what to expect and thus are more likely to keep returning to the site.
Producers for this task include web designers, System/site administrators.
Post-conditions for this task might include a set of standards for both the usability aspects (UI) and functional aspects of the proposed website.

Figure 7. Define website standards.

As mentioned above, there are also some new Techniques introduced here for supporting the use of OPEN in web developments. Two general techniques, the first primarily applicable in web developments but the second more generally useful, are Development Spikes (figure 8) and Field Trips (figure 9).

4.1.2. Components and frameworks
Web projects tend to have at least one level of their architecture that is component-based. The OPEN framework detailed in "The OPEN Process Specification" [Graham *et al.* 1997] does not include adequate support for component-based development (CBD). There does, however, exist an extension to the OPEN framework that allows for CBD. This extension can be found in more detail in [Henderson-Sellers 2001] and will not,

Focus: Minimising risk, solving unknowns.
Typical tasks for which this is needed: Develop and implement resource allocation plan; Develop software development context plans and strategies.
Technique description: A development spike can be thought of as research. The aim of the technique is to minimize areas of high risk by diving right in and starting development within a certain technology. The idea is that once development has begun, a greater understanding of the problem will be obtained and risk and time assessment will be more accurate. The development done within development spikes can be used as a reference but should never be used in the actual final development of the project. 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.
Technique usage: Areas of high risk are identified and code is quickly created (hacked) to gain better knowledge of a certain technology and/or problem. Development spikes are ideally suited to obtain quick answers regarding specific technologies. A typical development spike might start with the question, "Can I connect to a MySQL database using ASP?" As stated in the description, the work produced within a development spike should not be part of the final solution. eXtreme Programming – or XP [Beck 2000] – provides a good example of the use of development spikes.
Deliverables and outputs (post-condition): A small working example that demonstrates an answer to the particular question posed.

Figure 8. Development spikes.

Focus: Examining the current business environment and final place of deployment of the system.
Typical tasks for which this is needed: all tasks associated with requirements engineering.
Technique description: This technique is really quite self-descriptive. It serves 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.
Technique usage: A time should be arranged for the development team to go to the physical site of business operations and be given a tour.
Deliverables and outputs (post-condition): No explicit deliverables, although this technique helps to identify implied user requirements. A list of these may form the post-condition.

Figure 9. Field trip.

therefore, be discussed any further here, while noting that the nature of web development projects, and their component-based architecture, means this extension would be useful when implementing a Web OPEN framework.

4.1.3. Content management and personalization

Another major factor within web development is both the idea of content management and personalization. These both represent functionality that must be included in the majority of web projects today. It can be debated as to the level of which these factors should be represented in the OPEN framework. Since these issues can be considered to

play an intricate part in the architecture of the solution, three new Tasks are proposed. Two of these are at the planning stage: "Design and implement content management strategy" (figure 10); and "Design and implement personalization strategy" (figure 11); and one at the enactment stage of project management: "Undertake content management" (figure 12). These Tasks can often be component based and so could often run parallel to the new activity of "Component selection".

4.1.4. Architecture and architectural patterns

In recent times there has been much discussion on architecture and patterns within software development. This is also the case in the web development community. Creating a solid architecture is seen by many as the most crucial component of a successful systems

---

Focus: Incorporating a content management system into the project's architecture.
Typical supportive techniques: Review; System metaphors; Project planning.
Explanation: The power of most web application and web projects lies in the ability of users to be able to access up-to-date information quickly. Without a mechanism for keeping the information within a web project up-to-date this advantage is lost.
The purpose of this task is to design and incorporate a system to allow the content within a project to be updated both easily and quickly. The extent to which this affects the overall architecture of the solution depends on the type and extent of content to be updated and the method used to update it.
Producers for this task include Requirements modellers, web designers, System architects and Editors.
Post-conditions for this task include a documented content management strategy.

Figure 10. Design and implement content management strategy.

Focus: Incorporating a personalization system into the project's architecture.
Typical supportive techniques: Project planning; Impact analysis; Impact estimation table.
Explanation: As web projects can often deal with a large range of target users, it makes the task of designing a usable interface difficult. Each user has different expectations about the size, the color, the shape and where particular bits of information should be, as well as functionality that will support them in achieving their goals. To overcome this problem, the notion of user adaptation or personalization has been created. Details about the user (or class of users) are stored and the system adapts the content, structure, presentation and/or functionality to suit the specific user. This concept also covers the idea of portals.
This task looks at defining the level of personalization allowed for the system. It looks at the type of user profile information that will be stored and the method used to do so. The extent to which this affects the overall architecture of the solution depends on the type and extent of personalization and the method used.
Producers for this task include Requirements modellers, web designers and System architects.
Post-conditions for this task might include documentation for personalization (technical and business aspects).

Figure 11. Design and implement personalization strategy.

Focus: Managing the actual content (rather than the form) of material on the website.
Typical supportive techniques: Configuration management.
Explanation: Content on a website can change rapidly, sometimes daily. Managing this content is a change management challenge. Because pages are hyperlinked together and often to other sites (to and from) nothing is more frustrating than to try to follow a link that ends in a cul-de-sac or results in stale out-of-date information. Good content management would obviate such problems.
Producers for this task include Editor, Project manager.
Post-conditions for this task include a stable process for maintaining the quality and integrity of the content on the website.

Figure 12. Undertake content management.

Focus: Creating an architecture for the website that will last.
Typical supportive techniques: Concept maps.
Explanation: Websites tend to evolve constantly during their lifecycle. This evolution is often very incremental and fine-grained. Without a solid architecture, it is likely that this evolution will result in a system that rapidly deteriorates in quality. This task focuses on creating such an architecture including considerations of security, usability and functionality. While websites are continually changing as owners update both content and functionality, it is important to permit this flexibility within an overall strategy (the architecture).
It is also worth noting that the architecture needs to encompass not only the technical structure, but also the information architecture – how the content will be managed, organised, etc. At present, architectural modelling approaches do not handle well the integration of these various aspects.
Producers for this task include System architects. Post-conditions for this task includes a documented architecture for the proposed website.

Figure 13. Design website architecture.

development. In the case of web developments, we introduce specifically a task called "Design website architecture" (figure 13).

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., Sun's initiative, Java 2 Platform, Enterprise Edition Blueprint focuses on the development and deployment of applications using the J2EE platform); 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 [Butler 2000]. 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 a new task "Choose architectural pattern for website" (figure 14) has been created.

Focus: Choosing an appropriate architectural pattern for the project domain.
Typical supportive techniques: Domain analysis.
Explanation: The majority of projects that are undertaken have some relation to projects that have already been completed. Projects can be classified into domains depending on their particular characteristics (B2B, B2C, etc.). Good use can be made here of previous experience, perhaps encapsulated in patterns. In order for patterns to be of use, a certain degree of domain analysis must have taken place. Domains can usually only be identified after a number of projects have been completed, these projects reflected on and grouped roughly according to their characteristics. Web development is currently reaching the stage in its evolution where this is possible.
There are a number of organisations that have completed patterns analysis emerging in the field of web development. Some of these include Sun's Java 2 Platform Enterprise Edition (J2EE), Microsoft's work revolving around the Duwamish Books and IBM's work on "Patterns for e-business." IBM's work currently stands out as it focuses on separating the actual architecture of the solution from the technology used to implement the solution.
Currently IBM has a number of separate architectural patterns in the following 6 business areas: User to Business; User to Online Buying; Business to Business; User to User; User to Data; and Application Integration [Butler 2000]. Further detail on these patterns can be found at the IBM website for the application framework for e-business (`http://www.ibm.com/framework`).
The purpose of this task is to minimize both design time and risk by selecting and adapting an architectural pattern that has already been developed and tested. This pattern can then act as a starting point for further development. An unfortunate problem with selecting an architecture is that it is often restricted by the choice of a component framework. For example, it would be more time consuming to implement anything else but a Microsoft architecture on a .NET framework. It is for this reason that it is recommended that this task runs parallel with the task "Choose appropriate component framework".
Producers involved in this task include Requirements modellers and System architects.
The pre-condition of this task is a domain analysis of the project. The post-condition is an architectural framework to use as a starting point for the architectural design of the system.

Figure 14. Choose architectural pattern (subtask of "Create a system architecture").

An interesting technique, borrowed from XP [Beck 2000] is that of system metaphor (figure 15). Here the developers try to find a useful analogue for their architecture, often in the business world. For instance, on an e-commerce site, we might design the architecture in terms of one to support a suupermarket analogy or metaphor with shopping trolleys and checkouts.

4.1.5. Content development

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

Focus: Conveying the system architecture in an understandable non-technical language.
Typical tasks for which this is needed: Identify user requirements; Undertake the architectural design.
Technique description: A technique originally used in the Extreme Programming (XP) process [Beck 2000] for naming classes and methods. It was designed in order to keep the entire team thinking along the same lines when it comes to naming. However, this is also an important technique when discussing an architecture of a system. It is important for the entire development team as well as the client to have an overall understanding of the architecture of the system being produced. As web development teams tend to have a wide range of producers from varying disciplines, the use of a system metaphor is a good tool to communicate across these platforms.
Note that this use of the idea of metaphors is different from the way the concept is used in user-interface development. In this latter case, a metaphor is often used as the basis for the design so that users have a known (and consistent) model of interaction. For example, the "supermarket" concept of searching for goods and then adding them to a shopping cart that is then "checked out" is widely used. An alternative metaphor used for e-commerce applications may be an "exclusive boutique" where items are presented to a user for consideration, rather than the user having to search for them.
In effect, the first use of metaphors (the focus of this technique) is to provide a basis for developers to understand the system during development. The second use is to provide a basis for users to understand how to interact with the system.
Technique usage: Choose a system of names for your project that everyone can relate to. Ideally it should be related to the business area for which the system is being designed (i.e., not the system itself); decide on the style of commercial interactions (transaction processing) that is desirable on the website. The example used in the XP process was for the Ford Car Sales system where the naming was structured as a bill of materials. Naming conventions within this system contained metaphors like "production line".
There is also a metaphor known as a naïve metaphor that is based on the system itself. The XP example of this is a system to control a coffee maker. As everyone is familiar with a coffee maker, they decided to use a naïve metaphor based on the system itself. A naïve metaphor should not be used unless it is very simple.
Deliverables and outputs (post-condition): A naming scheme for the architecture.

Figure 15. System metaphors.

print. The preparation of images, editing and layout of text, and obtaining copyright clearances all must be completed. There is no existing Task that deals with these kinds of things within the standard OPEN framework. The addition of such a task is essential for a Web OPEN framework. The name chosen for this new OPEN Task is "Create content (on website)" (figure 16). In addition, much reuse can be made in creating content. Thus the Technique of "Reuse of graphical components" (figure 17) is also introduced. One particular form of reuse is that provided by "Web templates" (figure 18) which is also a highly useful technique.

After this content has been carefully prepared, it must then be combined with the user interface. This is an ongoing task that must be done to bring together the worlds

Focus: Reviewing content. Editing and formatting.
Typical supportive techniques: Review; Reuse of graphical components; System metaphors.
Explanation: The purpose of this task is to finalize the content that will eventually be incorporated into the user interface. It parallels what an editor might do in the production of any print media. Copyright clearances should be obtained for any information to be displayed. The content should be edited to maintain a consistent feel. The overall relevance of the information should be reviewed for fitness for purpose.
In many case the content will not just be textual but involve multiple media forms. As such this task may also involve aspects such as audio and video editing, image manipulation (cropping, resizing, etc.), construction of animations. It will also involve ensuring the overall consistency of this content.
The producers for this task can vary depending on the scale of the web development. In largescale web projects there may be a separate editor while in smaller operations the majority of this work can be done by the actual customer. In general, the producer for this task must have a good overall feel for the project's higher business purpose as well as excellent communication skills. Other producers involved include Graphical designers and web designers, as well as specialists in managing non-textual media.
Post-conditions for this task include content that has been formally edited and reviewed. All copyright clearances and legal issues relating to the content should have been resolved.

Figure 16. Create content (on website).

Focus: Generating re-usable graphical components.
Typical tasks for which this is needed: Prepare content; Integrate content with user interface; Optimize reuse ('with reuse'); Optimize the design.
Technique description: This technique is more good practice than an actual technique. It is based around the concept that browsers cache pictures and therefore reusing a number of pictures will improve a site's performance and therefore its quality. This technique also focuses on minimizing the size of graphics without losing a significant amount of picture quality. What signifies a significant amount of picture quality depends on the purpose and use of the graphic.
Technique usage: Identify common graphical components within a system. Focus on re-using these graphical components where possible. Identify large graphical images within the site and experiment with different file formats (GIF, JPG, etc.) as well as picture resolutions to minimize the file size.
Deliverables and outputs (post-condition): A library of optimized graphical components to be used within the user interface.

Figure 17. Reuse of graphical components.

of print media and software development. On one side there is a team of creative type people 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. A new OPEN Task of "Integrate content with user interface" (figure 19) is thus proposed. This task is responsible for combining the content with the method being used to present that content to the user. This task is also

Focus: Generating standardised web templates for common user interface pages.
Typical tasks for which this is needed: Prepare Content; Integrate Content with user interface; Optimize reuse ('with reuse'); Optimize the design.
Technique description: This technique focuses on isolating common areas of content so they can be displayed in a consistent format. It also assists in the maintenance of a system by providing appropriate templates to use when adding new content to the site. The technology used to implement web templates can vary. Templates generally come with some kind of validation system that verifies that all the appropriate content has been entered. Some commonly used methods include Microsoft Word documents with embedded Visual Basic to validate the content, or online Active Server Pages (ASP) that allow administrators to update and validate new content online. The main advantage of using web templates is that it allows people with a lower level of technical experience with the system to do the majority of the work to create new content. This reduces the effect of any bottleneck that may occur with the site administrator. Web templates can also ensure that standards are met by forcing the collection of information such as metadata for indexing.
Technique usage: Divide the content into logically grouped areas that are to be displayed in the same style. Determine what aspects of the content need to be validated and any additional information that needs to be collected (e.g., metadata for search indexing). Design the structure of the web templates and a procedure for using them. Note that this step will be influenced by architectural and design decisions.
Deliverables and outputs (post-condition): A variety of web templates and a procedure for using them.

Figure 18. Web templates.

important within the Web OPEN framework as it highlights the difficulties that occur when combining two different cultures together within the same project.

4.1.6. User interface

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 already has a task named "Design user interface". This task needs to be somewhat more emphasized for web development projects. It does not warrant being labelled as an activity under the OPEN framework, yet deserves 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 [Norman and Draper 1986] given that it is often not possible to conduct effective user centred design. "Usage-Centered Design focuses on the work that users are trying to accomplish and on what the software will need to supply via the user interface to help them accomplish it" [Constantine and Lockwood 1999]. 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.

Focus: Combining the prepared content to be displayed by the presentation layer of the system.
Typical supportive techniques: Robustness analysis.
Explanation: This is the task of bringing two varying worlds of different disciplines together. The creative design team involves graphical designers and editors, whereas the software engineering team consists of web developers, software architects and programmers. The varying traits of each team can be summarized as follows:

Creative Design Team:
- Based around intuition not a formal process;
- Artistic in nature.

Software Engineering Team:
- Architecturally bound;
- Logical in nature;
- Focus on functionality; and
- Process orientated.

During this task, an agreement must be reached by both teams as to the content that is to be displayed and the method used to display it. In effect, there will be a trade-off between what is desirable from a creative perspective and what is technically feasible.
Producers involved in this task would be the web designers, Graphical designers, Editors and backend System developers.
Post-conditions for this task include a prototype and functional website.

Figure 19. Integrate content with user interface.

The three sub-tasks that have been added to supplement the original "Design user interface" task are "Create the UCD role model" (figure 20); "Create the UCD task model" (figure 21); and "Create the UCD content model" (figure 22). The last of these subtasks links in well with the new task "Integrate content with user interface" as it starts to identify the relationships between the content and the user interface including navigation maps (new Task: "Create navigation map for website" (figure 23)). 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.

Once the interface has been designed and perhaps partly constructed, a new task of "Prototype the human interface" (figure 24) should be executed in order to get early user feedback and thus improve the ongoing interface development.

4.1.7. White sites (the web prototype)

A task often completed in web development projects is the construction of what is termed in the industry as a "white site". A white site consists of no rich graphical content and usually only represents a portion of the entire website. In web development, the white site is responsible for a number of key factors that lead to a successful project:

- it provides valuable client confirmation that the developer has interpreted the systems requirements correctly;

Focus: Modeling the various roles that will use the system.
Typical supportive techniques: Active listening; Brainstorming; Workshops; Questionnaires.
Explanation: The purpose of this task is to identify whom, in terms of roles, will be using the system. A user role is an abstract collection of needs, interests, expectations, behaviors and responsibilities characterizing a relationship between a class or kind of users and a system [Wirfs-Brock 1993]. User roles generally relate to humans or more specifically to the roles that the humans assume while using the system.
A user roles list must be created during this task that identifies the needs, interests, expectations, behaviors and responsibilities that characterize and distinguish the different roles. Some appropriate questions to consider while constructing the role model include [Constantine and Lockwood 1999]:

- Who would or could use the system?
- What is the general class or group to which thay belong?
- What distinguishes how they would or could use the system?
- What characterize their relationship to the software?
- What do they typically need from the software?
- How do they behave in relation to the software, and how do they expect the software to behave?

The development of the user role model also includes identifying focal roles and creating a user role map.
Producers for this task include the Prototype developer and the Graphic designer.
The post-conditions for this task include a list of User Roles and a User Role Map.

Figure 20. Create the UCD role model (subtask of "Design user interface").

Focus: Modeling the various tasks that the system will be required to complete.
Typical supportive techniques: Active listening; Brainstorming; Workshops; Questionnaires; Videotaping; Essential use cases.
Explanation: The purpose of this task is to identify the nature of the work that is to be completed by the system. OPEN techniques such as videotaping are useful for this task since users will often tell you what they are meant to be doing rather than what they are actually doing.
The creation of essential use cases and a use case map are central to this task. Constantine and Lockwood [1999, chapter 5] present significant further detail on the process of creating a UCD task model.
The producer for this task is the System tester.
Post-conditions for this task might include a set of essential use cases and/or a use case map.

Figure 21. Create the UCD task model (subtask of "Design user interface").

- it integrates the change management strategy, the usage and the navigation of the site all together in a visible working solution;
- it works as a communication tool to display the site architecture to the client and the development team.

Strictly speaking, creating a white site is a form of prototyping, which is already covered in the OPEN framework. However, the usefulness of white sites, in the design

Focus: Modeling the various contexts that work takes place in.
Typical supportive techniques: Active listening; Brainstorming; Workshops; Questionnaires.
Explanation: Efficient user performance takes place in a context in which the necessary tools and materials are immediately available [Constantine and Lockwood 1999, p. 125]. Thus, in order for a system to be effective and user friendly, the appropriate tools and information must be available in the correct areas. The identification of these areas is dependent on the roles the user is playing (the role model) and the type of work that is being performed (the task model). To present a scenario to understand this concept better, you would go into your garage acting as a mechanic when you wanted to fix your car. In this scenario the role would be the mechanic, the task would be to fix the car and the context of the work would be the garage. This therefore makes the garage an obvious place to store all the tools you require to fix a car for ease of accessibility and improved efficiency. This scenario highlights the need to identify these interaction spaces within the user interface in order to improve efficiency and usability.
The content model is an abstract representation of the contents of the various interaction spaces for a system and their interconnections [Constantine and Lockwood 1999, p. 126]. To communicate this a content model with supporting navigation maps is used.
The producer for this task is the System tester.
Post-conditions for this task might include a content model which satisfies UCD principles.

Figure 22. Create the UCD content model (subtask of "Design user interface").

Focus: Finding way around a website.
Typical supportive techniques: Concept maps; Complexity measurement; web templates; System metaphors.
Explanation: Many of the earliest websites just grew without any planning. One immediate consequence is that the user rapidly gets lost and disoriented when using the site because there is no navigational logic to the site. In this task, we purposefully design the navigational structure of the site. A representation of this is often accessible within websites via a button labelled something like "site map".
This task will often be complicated by the existence of dynamically changing content, the appearance of the same data in multiple places, the complexity of the "work-flows" that may be supported by the site, and the diversity of users for a system. A number of design languages (such as WebML) and methods (such as OOHDM – Object-Oriented Hypermedia Design Model) have appeared that directly support navigational design.
Producers for this task include System architects, Systems modellers, web designers, System/site administrators.
Post-conditions for this task would include documentation describing the navigational structure of the website plus an endorsement of its high quality.

Figure 23. Create navigation map for website.

and requirements engineering phases of web development, warrants the existence of a separate Task. In addition, we note that there exist a number of various Techniques relating to how to create white sites, which points to it being a Task rather than a Technique. The name chosen to represent this new Task in OPEN is "Build white site" (figure 25).

Focus: Usability.
Typical supportive techniques: Simulation; Throwaway prototyping; Usability testing; Dialogue design in UI; Reuse of graphical components; System metaphors.
Explanation: Since the value of most websites is in terms of attracting users and then getting them to return to the same website later, usability is a key issue for website designers. The human interface is thus critical and, as part of building a white site, an important task is to trial or prototype the human interface.
Producers involved in this task would include Requirements modellers, Graphic designers, Prototype developers and System testers.
Post-conditions for the task include a user-interface prototype, an evaluation of this prototype and knowledge on how to improve the usability of the website in the next iteration.

Figure 24. Prototype the human interface.

Focus: The production of a visible solution incorporating the architecture and user interface of the system.
Typical supportive techniques: Use Cases; Robustness Analysis; Throwaway prototyping.
Explanation: This task is intended to provide feedback on the requirements of the web project. As Internet technology is constantly changing, it is often the case that either the client or developer (or both) are unfamiliar with the technology at hand. The early production of a prototype allows for increased estimation and risk assessment for the remainder of the project. The prototype most often developed in the web industry consists of a series of web pages with rudimentary content (graphical imagery is usually left out or remains in a very early stage of development) that sits on top of a simulation of the final architectural design. This type of prototype has been labelled a "white site" within industry.
A white site incorporates all of the major architectural components of the design without worrying about the lower level details. It typically would include a rough change management solution (or at least an example of how it would work) as well as sample pages of content incorporated with the navigation maps. The site will typically not provide full functionality, but will at least indicate what this functionality will be.
Pre-conditions for this task include draft navigation maps, a set of user requirements, and a proposed architectural solution.
As "white sites" are often used as a tool to help distinguish the system requirements the production of a white site is best served in the very early phases of development. Most web development organisations will charge the client for a definition phase in which the white site is created. This effectively moves the analysis and high level design of the problem into the same invoice.
The white site once developed may or may not be maintained for the duration of the project depending on the nature of the project and the client's requirements.
Producers for this task include the Web Designer, Graphic Designer, System Architect and Requirements Modeler.
Post-conditions for this task include a visible working prototype of the solution and an evaluation of this prototype. The white site should include all aspects of the solution that are considered to contain high risk factors. The prototype can then be used to further freeze user requirements as well as acting as an architectural base for the rest of the development team.

Figure 25. Build white site.

Focus: Developing standards for the data within the system.
Typical supportive techniques: Web metrics.
Explanation: The importance of the data within a project cannot be underestimated. The problems apparent with many of today's systems result from old legacy databases that are not adaptable enough to be able to handle changes and modifications. We cannot afford to make this mistake in the future and so the data collected and used within web projects must be extensible, scalable and preferably follow an appropriate standard in order to be compatible. There are industry standards that are being developed that can act as an excellent starting point. Some of these include the W3 Consortium's industry standards for XML tags. A well thought out data standard ensures that the system can be replaced with changing technology without affecting the data within the system.
A subsidiary aspect of this task is the adoption of suitable meta-data standards and formats, as well as procedures for ensuring their maintenance. Meta-data can be critical in supporting effective indexing, searching and control and management of the data.
Producers for this task would typically include Requirements modellers, System architects, System designers and Editors.
As the web develops further we can expect to see further system to system projects that provide services. This idea expands on the idea of web portals.
Post-conditions for this task include an agreed data standard for the website.

Figure 26. Develop data standards.

4.1.8. Standards

At present, the rapid pace of technological change and the growing complexity of web systems is leading to significant difficulties with regard to interfaces between various system components, web systems, legacy systems and related business systems. Although the technology (and in particular the communications protocols and data formats) supporting these interfaces will stabilise – led in part by the move to XML – other aspects will continue to evolve. As interfaces stabilise, changing knowledge representations will become a major focus, so that as these stabilise changes in agent brokers may become the focus. In other words, changing technology has become a constant factor within the web environment. Nevertheless, the development of data standards is still a critical aspect of web development, given the strong focus on content and the way in which it is managed. Consequently, an appropriate new OPEN Task is introduced: "Develop data standard" (figure 26). There has been a lot of work done on developing industry standard XML tags by the W3 Consortium and this would often provide an excellent starting point for this task. The importance of this task is more noticeable in large B2B projects than B2C projects.

4.1.9. Performance testing

Performance testing has been elevated to a new level on the web. We now see systems that must deal with tens of thousands of concurrent users on-line. The performance of a web project can often be determined by its ability to deal with these large loads. This links into the fact that users can quickly become frustrated from an unresponsive website

Focus: The performance of the project and associated processes under high usage.
Typical supportive techniques: Beta testing; Package and subsystem testing; Regression testing; Unit testing; Usability testing; web metrics.
Explanation: This subtask has a two-fold purpose. The first is to test the hardware configuration and its ability to handle an appropriate level of traffic or 'hits' as they are commonly referred to. The second and perhaps more important aspect of this subtask is to test the underlying processes and how they are affected by the expected maximum load on the system. There is no point in having a e-commerce web project that can handle 1 million hits a day if the fulfilment procedure responsible for delivering goods can only handle 100,000 orders a day.
The producer for this task is the System tester.
Post-conditions for this task include a suite of test results and their analysis together with a list of consequent action items.

Figure 27. Undertake testing of website.

Focus: Collection of metric data relating to web development.
Typical tasks for which this is needed: Evaluate quality; Undertake post-implementation review.
Technique description: One of OPEN's strong characteristics is its attention to metrics. While further work is needed in order to statistically verify what the most appropriate metrics are, an initial proposal should focus on:

- Interface complexity: at a simple level this can be estimated through a page count, though this can be misleading, as a single server-side page may contain scripting that results in many different client-side manifestations. A more effective metric may be interaction counts or something similar, though there has, to date, been little work in this area.
- Performance: this can be estimated initially through number of hits per page per unit time – determines the general usage of a page and indicates where optimization would best be served.
- Access: total size of pages (including graphics) – a useful quality measure in terms of speed to load.
- Maintenance: rate of change of content – a metric useful for indicating when a site or page has become stagnant.

Some of the metrics are relevant to the development process and some related to the maintenance aspect of web projects. In addition to these extra metrics supporting web development, the original metric techniques within the OPEN framework are still relevant.

Figure 28. Web metrics.

and quickly move on to another. The OPEN repository does not currently have a task relating to this kind of performance testing. The inclusion of a new Task "Undertake testing of website" (figure 27) highlights the importance of performance testing within web development projects. (Tasks and Techniques already exist in OPEN for usability and interface testing and evaluation, e.g., Technique: Usability testing and the newly proposed Task: "Define website testing strategy" (figure 6). An allied technique here could be elements of the newly proposed Technique: "Web metrics" (figure 28).)

Focus: Determining market description.
Typical supportive techniques: Active listening; Workshops; Questionnaires.
Explanation: The purpose of this task is to determine market share and sector characteristics. Market research is a huge industry and, if any significant research needs to be undertaken, it is recommended to outsource this particular task.
This task is useful in requirements engineering as it helps identify the usage of the system. It should provide information that better refines the requirements as well as affecting the overall design of the system. Things that should be looked into include average hardware profile of users, software most used for browsing, as well as non-technical issues such as the most common reason users might visit the site.
If this task is not being outsourced it is best performed by the Requirements modeller and/or anyone in the team with market research skills.
Post-conditions for this task include data and analysis of current and potential users and/or clients.

Figure 29. Undertake market analysis.

4.1.10. Market research

There are many tasks that are borderline between being classified as dealing with the engineering side of software development and the business side of software development. In web development this tends to be somewhat more evident. A new OPEN Task of "Undertake market analysis" (figure 29) is heavily associated with the upper business level, since there must be some understanding of the target audience before the organization decides to embark on a web development project. Unfortunately, with the state of affairs recently, this does not seem to be the case in practice. Many organisations simply want a website, almost for its own sake or "for appearances", and they will find out who they are targetting later. Also there is a significant amount of information that web developers need to know about their target audience with which the upper business levels would not be concerned. For example, an initial market research project may cover the disposable income of web users, their age range, etc. Web developers will want to know quite different things such as "What system are they using?" or "How fast is their connection?" The importance of this task is more noticeable in web projects that fall into categories that deal with customers (e.g., B2C). It is still important in other web projects (such as B2B, or B2E) but the information is generally easier to obtain and does not always warrant a separate task.

As part of market research, it is likely that the creation of a brand identity will be a major element. For Web OPEN we therefore propose a new Task: "Develop a brand identify" (figure 30) supported by a new Technique: "Branding" (figure 31).

4.2. Roles

Essential to the OPEN metamodel is the existence of work products and producers that are responsible for creating these work products. So, when creating Web OPEN, it is important to discuss the producers that exist within the field of web development. In

Focus: Image.
Typical supportive techniques: SWOT analysis; Branding.
Explanation: Branding can often be highly lucrative. A product with a brand can sell for significantly more than an equivalent unbranded product (and vice versa). In terms of websites, branding the products and the website so that it is immediately recognizable worldwide takes patience but is an important part of many dot.coms these days. Websites such as `www.amazon.com` have entered the general parlance and thus can be counted as successful branding forays.
Producers for this task include Project manager, Marketer, Graphic designer.
Post-conditions for this task include both a strategy to establish a brand *and* an identifiable branding or logo.

Figure 30. Develop a brand identity.

Focus: Creation of a brand identity.
Typical tasks for which this is needed: Develop brand identity; Communicate the brand identity.
Technique description: Web development is increasingly less about software development and more about marketing and developing a market identity or brand. This is a combination of the product and the way that the product is portrayed on the website (and other media – which are of less interest to us here). Brand strategies must be developed, an overall artistic and marketing agreement made as to what constitutes the "brand" and ways to attain widespread brand recognition. An example here in early web commerce was `www.amazon.com`. A number of high-profile e-commerce failures (such as `Boo.com` and ValueAmerica) have resulted, at least in part, from a lack of effective branding.
Technique usage: Identify the product to be branded; evaluate the website possibilities; investigate new logos or new ways of using old logos; register the website on as many search engines as possible.
Deliverables and outputs (post-condition): Recognition (as demonstrable by editorials in the media); identified relationship to existing branding strategies and competing brands.

Figure 31. Branding.

the following subsections, each of the relevant producer roles is briefly described, as identified during two industrial case studies [Haire 2000].

4.2.1. Requirements modeller

The requirements modeller is responsible for the collection and maintenance of system requirements. They play an intricate role in the business domain modelling of the system. They are concerned with "what" is to be built and not so much with "how" it is to be done. Ideally, they would only be concerned with the "what", although in practice the "how" often slips into the systems requirements. Requirement modellers need good communication and modelling skills. A general technical knowledge of systems is also an advantage. One significant difference that typically exists between a web requirements modeller and a modeller for a conventional system is the relationship to the design process. Most web development takes place within the context of rapidly evolv-

ing technology, a poor client understanding of the consequent changes to their business model and an overall lack of understanding of their own requirements. Typically, the design artifacts, or even partial solutions, are used to drive the elicitation of requirements – leading to an increased need for requirements modellers to understand how to utilise the design activities to facilitate clarification of requirements.

4.2.2. System architect

The system architect constructs the structure on which to build the entire web project. Without a good architecture it will be difficult to expand the system in the future. Everyone within the team should have a good understanding of the system architecture, but the system architect should have an excellent understanding as well as in depth knowledge as to why the architecture is the way that it is. The strength of a system's architecture is a key element in the success or failure of a web project. Due to the speed of change on the Internet (few web projects remain unchanged for more than a couple of months), a sound architecture is needed to allow for further system development. The skills required for this role include excellent modeling skills as well as plenty of experience with various systems and their design. Developing a good system architecture can often be a tricky but important task – especially given the poor current level of understanding about how to integrate aspects of the business model with the information architecture and with the technical system architecture.

4.2.3. System developer

The system developer fills in holes and connects the framework together. They are responsible for integrating components, developing new components and conducting the detailed design of the system. This is the role that produces most of the final functionality of the system. System developers should be logical thinkers with strong programming skills.

4.2.4. Web designer

The web designer needs to have a general skill level in a wide variety of areas. Ideally, they have some artistic ability for creating things such as simple graphics as well as general programming experience. The web designer helps to bind the gap between the artistic world of print media and the programming world of software engineering. They work with the graphic designer as well as the system developers to make the proposed content and layout a reality. Skills in HTML, Java Applets, Javascript, XML and many other web-based technologies are a necessity.

4.2.5. Graphic designer

Due to the amount of rich content that goes into many web projects there is a need for this role of a graphic designer. Their responsibility is to help prepare the content and layout for the final system. This can include photographs, music clips, video clips and much more. The graphic designer needs to be artistic and imaginative and possess strong skills in creating and altering computer media.

4.2.6. Editor (optional)
This is an optional role for many web projects. It is often left as the responsibility of the client to ensure that the content provided has been correctly edited and reviewed. In many cases, this role is not explicitly named but the tasks associated with the role are divided amongst the team, in particular, between the web designer and graphic designer. In projects with a large amount of content, it is a good idea to assign someone to this role more permanently.

4.2.7. System tester
This is another generic role from software development. The system tester is responsible for verifying and validation that all the components of the system meet their requirements. They work closely with the requirement modeller and system developers. The system tester should be methodical and have an eye for detail.

4.2.8. System/site administrator
As web projects gain their strength from their ability to provide up-to-date information, there is a need for constant maintenance of the content (as distinct from maintenance of the system technical components). This role can be completed by the team building the system, in-house by one of the client's team, out-sourced to an external team or some combination of these. The skills required are highly dependent on the system that is to be administered. Skills usually required include a high level of computer literacy, with basic programming skills being an advantage but not a necessity.

4.2.9. Project manager
This is a standard role in any project. They are responsible for the organisation and coordination of the team, ensuring things get delivered on time and on budget. Skills required are those of any project manager including things like leadership, communication, organisation and so on.

4.2.10. Prototype developer
Although the role of prototype developer is useful for application developments, it is included in OPEN with a web focus. The *prototype developer* is the role played by the person who is responsible for creating and testing the white site, i.e., the prototype website which does not yet have the full rich textual content necessary in the completed website.

4.2.11. A quick analogy
The following analogy portrays the roles and responsibilities of web producers in a more familiar environment. To use an analogy of building a room:

- the requirement modeller decides what type of room you need;
- the system architect provides the foundation and framework;
- the system developer provides the brick walls;

- the web designer provides the plaster over the brick walls;
- the graphics designer paints the plastered walls;
- the system tester makes sure the room will not fall over and that it is the room that was originally asked for;
- the system/site administrator is responsible for changing the pictures hanging on the wall; and finally
- the project manager (a.k.a. foreman) can be thought of as the ceiling that oversees everything and makes sure it comes together in a square (unless of course you are building a round room).

5. Some example OPEN process instances

The relationship and timing between activities forms the basis of the process for an organisation. The structure of this is dependent on both the organisation's business structure and the project at hand. The number and type of intercommunication paths between individual pairs of activities follows smoothly from the specification of contracts in terms of preconditions and postconditions on the activity objects.

Here, we examine in more detail the way that these activity objects are linked through to more fine-grained Tasks and Techniques. This is accomplished by a pair of deontic matrices which define the possibility values of each pair of Activity/Task and Task/Technique. For each of these two matrices (figure 32 illustrates schematically the one for Activities versus Tasks), possibility values are entered *for a specifically constructed and configured process instance*. In the OPEN approach [Graham *et al.* 1997; Henderson-Sellers *et al.* 1998; Henderson-Sellers and Unhelkar 2000] these are selected from one of five values: M = mandatory; R = recommended; O = optional; D = discouraged; and F = forbidden. The inclusion of optionality in the choice of Tasks and Techniques permits an organization or a project to select tasks and techniques specially suited to local conditions – including skills sets of project members, resource availability, level of criticality/safety requirements and so on. Having said that, however, we anticipate that for most organizations, these specific characteristics will remain static for the duration of the project (and often longer) so that the majority of the values in both deontic matrics are likely (our conjecture only) to tend towards becoming bimodal, i.e., using only M and F values.

Here, rather than the five alphabetically-encoded values, we use a simpler scale of (0–1) with an incremental value within the range of 0.1. A value of 0 is equivalent to a forbidden task and a value of 1 is equivalent to a mandatory task. A value of 0.5 implies that this task is likely to be carried out but with a lesser emphasis (it is considered to be optional). We also do not attempt to tailor these deontic matrices for a specific organization or a specific project but rather to offer guidelines and "rules of thumb" for specific categories of web developments; notably small/medium business-to-customer (B2C), large business-to-customer and business-to-business (B2B).

	Activity A	Activity B	Activity C	Activity D	Activity E
Task 1	M	D	F	F	F
Task 2	D	D	F	F	D
Task 3	D	D	O	O	D
Task 4	F	O	O	O	F
Task 5	F	M	O	D	F
Task 6	R	R	M	R	O
Task 7	D	R	F	M	O
Task 8	R	R	D	R	R
Task 9	O	D	O	O	R
Task 10	F	M	O	F	D

Figure 32. Schematic deontic matrix linking OPEN Activities and Tasks. Values in the matrix indicate possibility values (here labelled with one of five values – see text for details).

5.1. B2C

Within Business-to-Customer web projects there is more of an emphasis placed on the market research aspect of the requirements engineering than in business-to-business projects. Small to medium projects tend to have less of an in-depth analysis and design phase. However, this adds to the overhead of the project which then constitutes a large percentage of the overall cost if the project budget is small. Larger B2C projects, on the other hand, are more concerned with analysis and design phases, since in complex systems mistakes or errors early in the system are costly. Larger projects also tend to try and optimize reuse more as it can significantly reduce the development costs. These B2C considerations are reflected in the deontic matrix between Activities and Tasks. The tables associating activities to tasks for small/medium business-to-customer projects can be found in figure 33. These values were derived from two in-depth case studies at commercial sites in Australia in late 2000 [Haire 2000].

5.2. B2B

Within Business-to-Business web projects there is more of an emphasis placed on the business processes of the two organisations and how they will interact. Requirements engineering is more concerned with defining what needs to be communicated in order to improve and add value to the business processes. The process also tends to be more document-oriented since the system must be designed to work across two organisations. This is reflected in the deontic matrix between Activities and Tasks (figure 34).

6. Summary

6.1. Conclusions

As part of a research project to extend process support for web development, we have utilized the OO/CBD (Object-Oriented/Component Based Development) process known

	Small /Med	Large
Project Initiation		
Obtain business approval	0.4	0.8
Undertake feasibility study	0	0.4
Requirements Engineering		
Develop BOM	0.6	0.8
Identify context	0.7	0.7
Identify source(s) of requirements	0	0.3
Conduct Market Research	0.4	0.9
Create White Site	0.5	0.7
Develop Data Standard	0	0.4
Identify user requirements	1.0	1.0
Define problem and establish mission and objectives	0.6	0.5
Establish user requirements for distributed systems	0	0.5
Establish user DB requirements	0.3	0.8
Analysis and model refinement		
Analyze user requirements	0.9	0.9
Create White Site	0.5	0.7
Develop BOM	0.7	0.7
Undertake architectural design	0.6	1.0
Choose Architectural Pattern	0.3	0.7
Design & Incorporate content management strategy	0.8	1.0
Design & Incorporate personalization strategy	0.5	0.8
Develop layer design	0.6	0.8
Establish distributed systems strategy	0	0.5
Select database/storage strategy	0.6	0.8
Project Planning		
Develop and implement resource allocation plan	0.8	0.8
Obtain business approval	0.5	0.5
Component Selection		
Screen the candidate list of component frameworks	0.1	0.7
Evaluate the potential component frameworks	0.1	0.7
Choose appropriate component framework	0.1	0.7
Screen the candidate list of components	0.3	0.4
Evaluate the potential components	0.3	0.4
Choose appropriate components	0.3	0.4
Build		
Construct the object model	0.2	0.2
Create and/or identify reusable components	0.3	0.8

Figure 33. Deontic matrix values for Activities and Tasks for B2C. Tasks which do not appear are not required, i.e., their possibility value is zero.

	Small /Med	Large
Construct Frameworks	0.3	0.7
Optimize for reuse	0.4	0.7
Optimize reuse ('with reuse')	0.5	0.5
Build – Evolutionary development		
Prepare content	0.6	0.9
Code	1.0	1.0
Develop Data Standard	0	0.4
Integrate Components	0.8	1.0
Design and implement physical database	0.7	0.7
Distribution/replication design	0	0.4
Operational and performance design	0.5	0.8
Performance evaluation	0.5	0.8
Design user interface	1.0	1.0
Create the role model	0.8	0.9
Create the task model	0.8	0.9
Create the content model	0.8	0.9
Integrate Content with User Interface	0.8	0.9
Identify CIRTs	0.1	0.3
Determine initial class list	0.1	0.3
Identify persistent classes	0.1	0.3
Identify roles	0.1	0.3
Refine class list	0.1	0.3
Map logical database schema	0.7	0.7
Map roles on to classes	0.3	0.3
Test	0.9	0.9
Perform acceptance testing	0.9	0.9
Perform class testing	0.8	0.8
Perform package/cluster testing	0.8	0.8
Perform regression testing	0.8	0.8
Performance Testing	0.8	0.9
Undertake usability design	0.8	0.8
Build – User review		
Evaluate quality	1.0	1.0
Analyze metrics data	0.1	0.7
Evaluate usability	0.9	0.9
Review documentation	0.5	0.3
Build – Consolidation		
Optimize the design	0.5	1.0

Figure 33. (Continued).

	Small /Med	Large
Evaluation		
Evaluate quality	0.9	0.9
Maintain trace between requirements and design	0.1	0.4
Optimize the design	0.4	0.4
Write manuals and prepare other documentation	0.1	0.2
Programme planning		
Model and re-engineer business process(es)	0.9	0.9
Build context (i.e., business process) model	0.6	0.6
Build task object model	0.4	0.6
Convert task object model to business object model	0.4	0.6
Do user training	0.8	0.6
Prepare ITT	0.5	0.6
Optimize reuse (‘with reuse’)	0.5	0.5
Undertake feasibility study	0.3	0.3
Resource planning		
Develop and implement resource allocation plan	0.7	0.9
Choose hardware	0.1	0.6
Choose project team	0.1	0.6
Choose toolset	0.1	0.6
Decompose programs into project	0.1	0.7
Develop education and training plan	0.1	0.7
Develop iteration plan	0.6	0.9
Develop timebox plan	0.7	0.9
Identify project roles and responsibilites	0.6	0.9
Manage packages/subsystems	0.1	0.9
Set up metrics collection program	0.7	0.8
Specify individual goals	0	0.6
Specify quality goals	0.4	0.8
Use dependencies in the BOM to generate first cut project plan	0.4	0.9
Develop software development context plans & strategies	0.9	0.9
Develop capacity plan	0.7	0.8
Develop contingency plan	0.7	0.9
Develop security plan	0.7	0.9
Establish change management strategy	0.5	0.9
Establish data take-on strategy	0	0.9
Integrate with existing, non-OO systems	0.5	0.8
Tailor the lifecycle process	0.8	1.0
Model and re-engineer business process(es)	0.8	0.8

Figure 33. (Continued).

	Small /Med	Large
Build context (i.e., business process) model	0.7	0.7
Build task object model	0.4	0.7
Convert task object model to business object model	0.4	0.7
Do user training	0.4	0.7
Prepare ITT	0.4	0.7
Domain modeling		
Create and/or identify reusable components	0.6	0.9
Undertake architectural design	0.4	0.7
Other projects		
Manage library of reusable components	0.4	0.7
Optimize reuse ('with reuse')	0.3	0.6
Optimize the design	0.3	0.6
Use of system		
Deliver product to customer	1.0	1.0
Undertake in-process review	0.9	0.8
Undertake post-implementation review	0.7	0.7
Write manuals and prepare other documentation	0.1	0.3

Figure 33. (Continued).

as OPEN. OPEN is defined by a meta-level architecture or framework that contains several metaclasses (figure 2). Instances are then created of these metaclasses and, either directly or from the repository (figure 3), are selected and configured.

Web projects must be developed with the emphasis being on how services can be improved and not on the technology involved. Success on the Internet is more than having the fastest computers and the biggest databases; it is about complete fulfilment of the business processes. This includes issues such as customer query handling, product delivery and tracking, as well as service guarantees and speed of connections. It is also important to note that these business processes themselves are changed by the introduction of the new systems – so we end up with a system that by its very nature will modify its context and thereby require further changes in order to remain effective.

This emphasis is reflected in Web OPEN with the addition of tasks relating to the analysis of the data and how they will be used within the final system. This emphasis also needs to be represented within the user interface for successful web projects. This has been completed in Web OPEN by the inclusion of tasks and techniques taken from Usage-Centered Design [Constantine and Lockwood 1999]. Usage-centered design focusses on how the user interface will be used in order to improve the underlying business need of the system, rather than on who will be using it, as would be the case in user-centred design.

A number of other smaller modifications have been made to the OPEN framework that are particular to web development, including testing, market analysis and the development and review of content. OPEN's strength in the field of metrics is maintained

Project Initiation	
Obtain business approval	0.3
Undertake feasibility study	0.8
Requirements Engineering	
Develop BOM	1.0
Identify context	0.7
Identify source(s) of requirements	0.2
Create White Site	0.7
Develop Data Standard	0.9
Identify user requirements	1.0
Define problem and establish mission and objectives	0.9
Establish user requirements for distributed systems	0.9
Establish user DB requirements	0.8
Analysis and model refinement	
Analyze user requirements	0.9
Create White Site	0.4
Develop BOM	0.9
Undertake architectural design	1.0
Choose Architectural Pattern	0.8
Design & Incorporate content management strategy	1.0
Design & Incorporate personalization strategy	0.3
Develop layer design	0.9
Establish distributed systems strategy	0.8
Select database/storage strategy	0.8
Project Planning	
Develop and implement resource allocation plan	0.8
Obtain business approval	0.5
Component Selection	
Screen the candidate list of component frameworks	0.7
Evaluate the potential component frameworks	0.7
Choose appropriate component framework	0.7
Screen the candidate list of components	0.4
Evaluate the potential components	0.4
Choose appropriate components	0.4
Build	
Construct the object model	0.4
Create and/or identify reusable components	0.8
Construct Frameworks	0.7
Optimize for reuse	0.7
Optimize reuse (‘with reuse’)	0.5

Figure 34. Deontic matrix values for Activities and Tasks for B2B. Tasks which do not appear are not required, i.e., their possibility value is zero.

Build – Evolutionary development	
Prepare content	0.4
Code	1.0
Develop Data Standard	1.0
Integrate Components	0.9
Design and implement physical database	0.9
Distribution/replication design	0.9
Operational and performance design	0.9
Performance evaluation	0.9
Design user interface	1.0
Create the role model	0.9
Create the task model	0.9
Create the content model	0.9
Integrate Content with User Interface	0.9
Identify CIRTs	0.5
Determine initial class list	0.5
Identify persistent classes	0.5
Identify roles	0.5
Refine class list	0.5
Map logical database schema	0.9
Map roles on to classes	0.6
Test	0.9
Perform acceptance testing	0.9
Perform class testing	0.8
Perform package/cluster testing	0.8
Perform regression testing	0.8
Performance Testing	0.9
Undertake usability design	0.7
Build – User review	
Evaluate quality	1.0
Analyze metrics data	0.9
Evaluate usability	0.9
Review documentation	0.6
Build – Consolidation	
Optimize the design	1.0
Evaluation	
Evaluate quality	0.9
Maintain trace between requirements and design	0.8
Optimize the design	0.8
Write manuals and prepare other documentation	0.8

Figure 34. (Continued).

Programme planning	
Model and re-engineer business process(es)	1.0
Build context (i.e., business process) model	0.8
Build task object model	0.8
Convert task object model to business object model	0.8
Do user training	0.8
Prepare ITT	0.8
Optimize reuse (‘with reuse’)	0.7
Undertake feasibility study	0.7
Resource planning	
Develop and implement resource allocation plan	0.9
Choose hardware	0.9
Choose project team	0.9
Choose toolset	0.9
Decompose programs into project	0.8
Develop education and training plan	0.8
Develop iteration plan	0.9
Develop timebox plan	0.9
Identify project roles and responsibilites	0.9
Manage packages/subsystems	0.9
Set up metrics collection program	0.9
Specify individual goals	0.7
Specify quality goals	0.8
Use dependencies in the BOM to generate first cut project plan	0.8
Develop software development context plans & strategies	0.9
Develop capacity plan	0.9
Develop contingency plan	0.9
Develop security plan	0.9
Establish change management strategy	0.9
Establish data take-on strategy	0.9
Integrate with existing, non-OO systems	0.9
Tailor the lifecycle process	0.9
Model and re-engineer business process(es)	0.8
Build context (i.e., business process) model	0.8
Build task object model	0.8
Convert task object model to business object model	0.8
Do user training	0.9
Prepare ITT	0.9
Domain modeling	
Create and/or identify reusable components	0.9
Undertake architectural design	0.8

Figure 34. (Continued).

Other projects	
Manage library of reusable components	0.8
Optimize reuse ('with reuse')	0.7
Optimize the design	0.7
Use of system	
Deliver product to customer	1.0
Undertake in-process review	0.8
Undertake post-implementation review	0.9
Write manuals and prepare other documentation	0.8

Figure 34. (Continued).

with the addition of tasks relating specifically to web metrics.

In this paper, we have described new and extended definitions for several instances (in the repository) of Activity, Task, Technique and Role. All instances were identified from two industry case studies, one in the standards domain and one in the commercial domain, both extensive users of website development approaches.

6.2. Further work

Web OPEN is a starting point for further research into web process development. Further work could be completed on the task *Integrate components*. The importance of this task will continue to escalate, as components become more developed and widely used. A significant amount of work on component integration has been completed by the Catalysis team [D'Souza and Wills 1998] and would prove beneficial to the Web OPEN framework. Component integration is complemented by component creation. There is no reason why organisations cannot produce their own custom components and include this within their development process. Further work towards merging, with OPEN, further ideas from usage-centered design is also under way.

The metrics that are specific to the web development process need to be further looked at, evaluated and statistically verified. Much the same way where good quality objects were shown to have less than 10 methods and 3–4 lines of code per method [Haynes and Henderson-Sellers 1996], similar web metrics must be obtained. Once these appropriate metrics have been produced, we will be able to better measure the quality of web projects.

There will always be constant need for review and modification of the Web OPEN framework as technology and the development environment changes. Further work could also incorporate varying development methodologies such as artificial intelligence or agent technologies.

Acknowledgements

This is Contribution number 01/08 of the Centre for Object Technology Applications and Research (COTAR).

References

Angelique, E. (1999), "A Lightweight Development Process for Implementing Business Functions on the Web," In *WebNet'99*, Honolulu, Hawaii, USA, pp. 262–267.

Beck, K. (2000), *Extreme Programming Explained*, Addison-Wesley, Reading, MA.

Bieber, M. and T. Isakowitz (1995), "Designing Hypermedia Applications," *CACM 38*, 8, 26–29.

Burdman, J. (1999), *Collaborative Web Development*, Addison-Wesley, Reading, MA.

Butler, M. (2000), "IBM's Patterns for E-Business," In *Object-World 2000*.

Ceri, S., P. Fraternali and A. Bongio (2000), "Web Modeling Language (WebML): A Modeling Language for Designing Web Sites," In *Proceedings of WWW9 Conference*, Amsterdam.

Conallen, J. (2000), *Building Web Applications with UML*, Addison-Wesley Object Technology Series, First Edition, Addison-Wesley, Reading, MA.

Constantine, L. (1995), "What Do User's Want? Engineering Usability into Software," *Windows Tech Journal 4*, 12, 30–39.

Constantine, L. and L. Lockwood (1999), *Software For Use*, Addison-Wesley, Reading, MA.

Dart, S. (2000), *Configuration Management: The Missing Link in Web Engineering*, Artech House, Boston, MA.

D'Souza, D. and A. Wills (1998), *Objects, Components, and Frameworks with UML: the Catalysis Approach*, Addison-Wesley, Reading, MA.

Firesmith, D. and B. Henderson-Sellers (2002), *The OPEN Process Framework. An Introduction*, Addison-Wesley, Harlow, UK.

Firesmith, D., G. Hendley, S. Krutsch and M. Stowe (1998), *Object-Oriented Development Using OPEN: A Complete Java Application*, Addison-Wesley, Harlow, UK.

Fournier, R. (1999), *Methodology for Client/Server and Web Application Development*, Yourdon Press.

Graham, I., B. Henderson-Sellers, and H. Younessi (1997), *The OPEN Process Specification*, Addison-Wesley, Harlow, UK.

Haggard, M. (1998), *Survival Guide to Web Site Development*, Microsoft Press.

Haire, B. (2000), "Web OPEN: An Extension to the OPEN Framework," Undergraduate Capstone Thesis, University of Technology, Sydney.

Haynes, P. and B. Henderson-Sellers (1996), "Cost Estimation of OO Projects: Empirical Observations, Practical Applications," *American Programmer 9*, 7, 35–41.

Henderson-Sellers, B. (2001), "An OPEN Process for Component-Based Development," In *Component-Based Software Engineering: Putting the Pieces Together*, G. Heineman and W. Councill, Eds., Addison-Wesley, Reading, MA.

Henderson-Sellers, B., A. Simons and H. Younessi (1998), *The OPEN Toolbox of Techniques*, Addison-Wesley, Harlow, UK.

Henderson-Sellers, B. and B. Unhelkar (2000), *OPEN Modeling with UML*, Addison-Wesley, Harlow, UK.

Lowe, D. (2000), "A Framework for Defining Acceptance Criteria for Web Development Projects, In *Second ICSE Workshop on Web Engineering*, S. Murugesan, Ed., Limerick, Ireland.

Martin, R. (2000), "A Case Study of XP Practices at Work," In *XP2000*, Cagliari, Italy.

Norman, D. and S. Draper (1986), *User-Centered Design*, Lawrence Erlbaum Assoc., Hillsdale, NJ.

Overmyer, S. (2000), "What's Different about Requirements Engineering for Web Sites?," *Requirements Engineerng Journal 5*, 1, 62–65.

Schwabe, D. and G. Rossi (1995), "The Object-Oriented Hypermedia Design Model," *Communications of the ACM 38*, 8, 45–46.

Siddiqi, J. (2000), *eXtreme Programming: Pros and Cons*, IEEE Computer Society.

Wirfs-Brock, R. (1993), "Designing Scenarios: Making the Case for a Use Case Framework," Smalltalk Report, November–December.