

Characteristics of Web Development Processes

David Lowe, Brian Henderson-Sellers

Abstract— **The nature of Web system development is significantly different from conventional software development. Amongst other factors, there is substantial uncertainty in both clients' understanding of their needs and developers' understanding of the system domain. We discuss these differences and the impact that they have on the development processes that are adopted for commercial Web systems.**

Index Terms— **Software Engineering, Software requirements and specifications, Software development management**

I. INTRODUCTION

WEB systems are often viewed as simply a form of software systems. Therefore, development approaches that have been established and refined for software systems development can readily be applied to the development of Web systems. Whilst this is true to a limited extent, there is a growing recognition that Web systems have various unique characteristics that are only poorly addressed by conventional development practices. Development practices from related domains (software engineering, graphic design, marketing etc.) do not typically address these differences particularly well. Despite this there has been little consideration within the research literature of the implications of these characteristics on the development process. This is in spite of the obvious growth in importance of these systems to business success. For example, a recent International Data Corp report predicted that U.S. expenditure on Web-based initiatives would grow from US\$12billion in 1999 to US\$43.6 billion in 2002. The systems being developed leverage the infrastructure of the Internet and an increasingly complex set of Web standards, protocols and technologies to provide sophisticated business solutions that merge Web-based front-ends with complex back-end software.

A/Prof David Lowe is with the Faculty of Engineering, University of Technology, Sydney. E-mail: david.lowe@uts.edu.au

Prof Brian Henderson-Sellers is with the Faculty of Information Technology, University of Technology, Sydney. E-mail: brian@it.uts.edu.au.

These solutions cover the spectrum from electronic commerce to information provision and management to business-to-business support systems.

In this paper we consider the differences between Web systems and conventional software systems, and explore the implications of these differences for system modelling, development practices and techniques, and overall development processes. We begin (in Section II) by discussing the differences between conventional software systems and Web systems. In particular, we look at those aspects that are most likely to impact on the way in which we develop these systems. We then move on to look at the specific impacts on the approach to development. In Section III, we look at the need for different modelling approaches and, in Section IV, we consider the impacts on the broader process — particularly the tasks and activities that need to be carried out. Finally, in Section V, we speculate a little about the areas requiring most immediate attention and then conclude the paper.

II. DIFFERENCES BETWEEN WEB SYSTEMS AND CONVENTIONAL SYSTEMS

There is a growing body of research that is attempting to understand the differences between Web systems and conventional software systems [1], [2], [3]. In general, a distinction is made between the unique characteristics of Internet-enabled systems that are technical (i.e. related to the specific technologies that are used and how these impact on the structure of the application) and those that are organisational (i.e. related to the ways in which organisations make use of these systems).

A. Technical Differences

There are obvious technical differences between Web systems and more conventional software and IT systems. The most significant of these are as follows:

Link between business model and architecture

Possibly the most obvious difference between web and traditional software development is seen in regard to the specific technologies that are used and the ways in which these are interconnected. For example, the technical structure of Web systems merges a sophisticated business architecture (that usually implies significant changes to the business model of the client) with both a complex information architecture and a highly component-based technical architecture [4]. The linkage between the business architecture and the technical design of the system is much tighter than for conventional software systems. Similarly, the information architecture (which covers aspects such as the content viewpoints, interface metaphors and navigational structures) is substantially more sophisticated than conventional software systems.

Open modularised architectures

Related to the above point is the emphasis that is typically placed on open and modularised architectures for web systems. Though not unique to Web systems, it is often more pronounced. Web systems are often constructed from multiple COTS (commercial off-the-shelf) components that are adapted and integrated together — particularly for the system back end middleware layers. This implies that strong integration skills become much more critical in most Web projects.

Rapidly changing technologies

The technology that underpins most web systems is changing very rapidly. This has several consequences. The first is that it increases the importance of creating flexible solutions that can be updated and migrated to new technologies with minimal effort. For example, the need for reusable data formats (such as XML) increases substantially. A second consequence is that developer's understanding of these technologies is often restricted, thus increasing project risks.

Content is king

Of notable significance is the importance of content. Irrespective of the sophistication of the functionality and the creativity of the interface, a site is likely to fail without appropriate, substantial and up-to-date content. This implies both an effective information design as well as suitable content management.

Increased emphasis on user interface

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

A little more subtly, the emergence of authoring tools have focused on supporting rapid development and on visual design rather than functionality. This in turn has promoted a greater use of designs as a part of a specification — and thus allows a more interactive process between gathering requirements and building solutions.

Increased importance of quality attributes

Web systems represent an increase in mission critical applications that are often directly accessible to external users and customers. Flaws in applications (be they usability, performance or robustness) are therefore typically more visible externally and hence more problematic.

B. Organisational Differences

In addition to the technical differences, and possibly more important than them, are a number of organisational characteristics that are either unique or heightened in Web systems [1].

Client uncertainty

With Internet-based systems, the technology, development skills, business models and competing systems are changing so rapidly that the domain is often not only poorly understood, but also constantly evolving [5]. This provides a substantial degree of uncertainty in the project context, and consequently makes resolving requirements very problematic. Indeed, clients often have problems not only articulating their needs, but also in understanding whether a particular design will satisfy their needs — as they have a poor understanding of their own needs with respect to the Web. It is also worth noting that many web projects are vision-driven rather than needs-driven leading to an initial lack of clarity, consequently

increasing the importance of incremental and prototyping approaches.

Changing business requirements

Related to the previous point is the volatility in requirements. Specifically, given that clients' understanding of not only the technology's capabilities but also the potential impacts on their businesses change dramatically during a project, it is not surprising that the project scope and focus will often evolve considerably during the course of a project. This is also coupled with business models that are evolving rapidly as organisations migrate to an increased reliance on Internet technologies [6]. These issues are exacerbated by the lack of effective design tools, as well as the next two characteristics.

Short time frames for initial delivery

Web development projects often have delivery schedules that are much shorter than for conventional IT projects — often in the range of 1-3 months. This is partly a consequence of the rapid pace of technological development and partly related to the rapid uptake of Web systems.

Highly competitive

Web projects tend to be highly competitive. This is, of course, not new — being typical of the IT industry in general. The nature of the competitiveness is, however, somewhat different. There is regularly a perception that with simple Web authoring tools anyone can create an effective site. This creates inappropriate expectations from clients coupled with numerous small start-up companies claiming to be doing effective Web design but in reality offering little more than HTML skills and rudimentary graphic design. The result is a highly uninformed competitiveness.

Fine-grained evolution and maintenance

Web sites typically evolve in a much finer-grained manner than conventional IT applications. The ability to make changes that are immediately accessible to all users without their intervention means that the nature of the maintenance process changes. Rather than a conventional product maintenance / release cycle, we typically have an ongoing process of content updating, editorial changes, interface tuning etc. The result is a much more organic evolution.

Given these unique characteristics of Web sys-

tems, we can investigate desirable changes in the development methods and processes. For each of the different areas impacted, we will look at each characteristic and consider how it is being addressed. We will begin by looking at changes in the models and notations that are used to support the development of Web systems.

III. CHANGES TO MODELS AND NOTATIONS

Although Web systems can be viewed as software systems, this does not automatically imply that existing representations of various aspects of these systems will be able to be directly applied. Indeed, to blindly apply existing models to the representation of Web systems would encourage developers to overlook the peculiarities of these Web systems, and hence not address these peculiarities, leading to inappropriate solutions.

This is not to say that existing models should not be utilised — simply that we need to do so with an awareness of their limitations with respect to the aspects of Web systems that we wish to understand and document. We also need to understand how these limitations may be circumvented by appropriately supplementing (or replacing where necessary) the models. Let us look at the unique characteristics of Web systems, including those discussed above, and consider the impact of each, in turn, on what we may wish to represent.

Link between business model and architecture

The impacts that Web systems have on business models implies that there is a need to be able to understand (and document) the link between business models and system architectures. This has typically been only implicitly addressed in traditional development as the business models are well established and understood. This is less true for Web projects and, as a result we see a growing body of work — largely emerging from large technology vendors such as IBM, Sun and Microsoft — that considers how to represent supported business functions and the technical architectures required to support these. The most mature of these approaches is the patterns for e-Business work being developed by IBM (see <http://www.ibm.com/framework/patterns/>). This work provides a framework for identifying common patterns of business models. As stated in [7]:

The paths to creating e-businesses are repeatable. Many companies assume that they are unique and that therefore every creation of an e-business has to be learned as you go. In fact, there are lessons and architectural paths or patterns that can be discerned from all these engagements.

For each business pattern, a number of logical architectures (or topologies) are defined. These topologies provide a mechanism for fulfilling a particular business need. In effect, these models provide a direct link between the business models that underpin the systems being developed and the technical architecture that supports these business models. One problem with these current approaches is that the architectural models tend to emphasise functionality, with little consideration of how to represent the information architecture. In particular, aspects such as content modelling, information viewpoints etc. are not addressed.

Open modularised architectures

Although there is significant attention on modelling of open and component-based systems, little attention has yet been applied to considering the modelling of these systems in the context of the Web.

Rapidly changing technologies

Except where addressed peripherally, the rapid pace of technological change is only poorly considered. Indeed, the work on detailed design notations for representing certain aspects of Web systems (discussed in the next few points) may actually create problems in terms of the portability of designs into new technologies. Alternatively, work on architectures and, more broadly, on information models tends to create designs that are less dependant on specific technologies, and hence more likely to be able to be adapted to changes.

Content is king

The importance of content within Web sites implies a need to at least consider how we understand and represent the informational elements of a Web system. It is not surprising therefore that that much of the earliest work on Web development models focused on information modelling and structuring.

Early approaches in this area evolved out of work on data modelling (such as Entity-Relationship models) and applied this to mod-

elling the information domain associated with applications. Indeed, much of this work predates the Web and focused on hypermedia design. For example RMM (Relationship Management Methodology [8]) claims to provide a structured design model for hypermedia applications. In reality, the focus is very much on modelling the underlying content, the user viewpoints onto this content and the navigational structures that interlink the content. OOHDM (Object-Oriented Hypermedia Design Model [9]) is a similar approach, though somewhat richer in terms of the information representations and based on object-oriented software modelling approaches. Other similar examples include EORM [10] and work by Lee [11]. WSDM [12] attempts to model slightly different characteristics — beginning more explicitly from user requirements, but these are only addressed in a very rudimentary fashion. In general, these notations were either developed explicitly for modelling information in the context of the Web, or have been adapted to this domain.

More recently, work on both WebML (Web modelling Language [13]) and the adaptation of UML (Unified modelling Language [14] — an emerging industry standard for modelling object-oriented systems — see for example [15]) has begun to amalgamate these concepts into a richer modelling language for describing Web applications. However, despite aims to support comprehensive descriptions, the focus (as with the above techniques) is very much on content modelling rather than describing the functionality that is a key element of most current commercial Web systems. This leads on to the next point.

Increased emphasis on user interface

A key element of user interfaces is the functionality that they provide. A few attempts have been made to integrate information modelling concepts with system functionality [16], [17] though in general these approaches are still rather simplistic, lack scalability and focus on low-level design representations. Conallen's [17] work in particular is interesting insofar as it attempt to link a users's view of the system (as seen through the interaction with Web pages) to the back-end processes that support this interaction.

Other researchers have looked at modelling the way in which systems are utilised. For example

Guell et al. [18] extend OOHDMD to include tools such as user scenarios and use cases. Vilain et al. [19] has adapted UML to representing user interactions. Other researchers have investigated the use of formal methods for representing navigational requirements [20] or timing constraints [21] — though these tend to focus on ensuring consistency rather than directly addressing the quality of the user interface.

Possibly the most fruitful work in this area is usage centred design [22], although a rigorous analysis of the application of these techniques to Web development has yet to be carried out.

Increased importance of quality attributes

As with some other aspects, this has not been directly addressed at a modelling level, except insofar as developing effective architectures that support characteristics such as robustness, scalability and reliability. Certainly these elements have not been effectively woven into the detailed Web requirements or design models.

Client uncertainty

Client uncertainty largely arises from a lack of understanding of the business problems being addressed by the Web systems and, as such, design prototypes on their own can play only a limited role. Nevertheless, some of the techniques mentioned above that focus on modelling the way in which systems are utilised [18], [19] may help reduce client uncertainty.

One avenue being pursued by the authors is the investigation of a characterisation model that represents the key aspects that need to be woven into an evolving specification of a Web system [23]. This model highlights the links between the various characteristics, especially including the link between the business architecture and the technical and information architectures. The intention is that it be used to guide the formulation and evaluation of project acceptance criteria, user requirements and detailed contractual specifications.

Changing business requirements

The points made in the above paragraph apply here as well. Furthermore, there is a belief held by some authors that configuration management must play a substantially increased role [24] — leading to some consideration of configuration management models for Web systems.

Short time frames for initial delivery

This is an issue that has yet to be effectively addressed in terms of how it impacts on Web design models and notations.

Fine-grained evolution and maintenance

Again, this has yet to be considered in any substantial detail. It is worth pointing out, however, that one aspect of modelling that actively inhibits effective Web system maintenance is the lack of a cohesive architectural modelling language that actively links the information architecture with the technical architecture [25]. Conversely, the information models (such as OOHDMD [9] and WebML [13]) actively support a much clearer understanding of the impacts of changes to various aspects of the underlying content, viewpoints or navigational structures.

IV. CHANGES TO THE PROCESS

Improving the modelling support for the unique characteristics of Web systems is a useful first step — but, on its own, it is not sufficient. We also need to consider how we actually carry out the development. This includes both the specific activities and tasks that are desirable, as well as broader process issues related to how we organise this work. We begin this section by looking briefly at our previous work that has directly addressed exactly this issue.

A. *Web OPEN*

OPEN (Object-oriented Process, Environment, and Notation) is a process-focused methodological approach to software-intensive systems development useful for both object-oriented and Component-Based Development (CBD). It is the longest established of the third-generation OO approaches and covers the full development lifecycle, including business as well as software issues. OPEN was developed and is maintained by the not-for-profit OPEN Consortium, an international group of methodologists, CASE tool vendors and developers. OPEN was initially created by the merger of earlier methods: MOSES, SOMA, Fire-smith, Synthesis and enhanced by state of the art ideas from BON, Ooram, UML etc. It is documented in a series of books (e.g. [26], [27], [28], [29], [30]) and in many journal articles, particularly in the journal JOOP. Many of these shorter

articles are to be found on the OPEN web site at <http://www.open.org.au>.

A unique aspect of OPEN is that it is not merely a process but a configurable family of processes, defined in terms of a metamodel (known as the OPEN Process Framework or OPF). This metamodel contains a number of major elements (Figure 1) that can be multiply instantiated: work units (such as activities and tasks); work products; and producers. From these instances of the process fragments, organisationally-specific processes can be readily constructed. The way these elements are put together is also the decision of the organisation or development team — thereby supporting the construction of highly customised development processes.

The component-based nature of OPEN permits appropriate extensions to support development in new domains. One such set of extensions is those for Web development, called Web OPEN (see Haire et al. [31] for a more complete treatment of Web OPEN). These extensions were derived primarily from an analysis of the documented differences between web development and conventional development, and validated using case studies of commercial Web development projects. It is worth noting that many of the original (i.e. non web) activities, tasks, techniques and roles in OPEN are still relevant to Web development. For example, the tasks relevant to the activities of Project Initiation, Implementation Planning and Project Planning will remain relatively unchanged. Activities such as Requirements Engineering and System Build will be most affected, since this is where the project domain affects the process most noticeably. During the remainder of the paper we shall use Web OPEN to illustrate many of the ways in which the development process can be adapted to suit Web development.

B. Required changes to the process

Again, we will consider each of the issues raised previously and investigate how this might be addressed by appropriate changes to the development process.

Link between business model and architecture

Although the relationship between the business model and the system architecture is beginning to be addressed at a notational level, there is little

work in this area in terms of processes that support the interpretation of business requirements and the relationship that these have to the architecture. The work that does exist tends to focus on the design of architectures (see the next point). One of the few exceptions is the IBM work on patterns for e-Business that was mentioned previously. Although not providing a formal process, it does suggest an implicit process whereby the broad business needs are used to select a suitable business pattern, and then to use this to guide the selection of suitable architectures.

Open modularised architectures

Web systems are often constructed from multiple COTS (commercial off-the-shelf) components that are adapted and integrated together. Indeed, strong integration skills become much more critical in most Web projects. The importance of a strong architectural design is also increased. Indeed, many see creating a solid architecture as the most crucial component of a successful Web systems development.

Within Web OPEN, a specific task called *Design web site architecture* has been introduced. This can be supported, as indicated by the work on patterns, by the selection of suitable architectural pattern — formalised in a Web OPEN task *Choose architectural pattern for web site*. The Component Based Development (CBD) additions to OPEN [32] provide further useful support.

One aspect that is yet to be effectively addressed is appropriate support (either as tasks or suitable techniques) for the linking of the various disparate elements of the architecture (i.e. informational and technical to the business architecture).

Rapidly changing technologies

Apart from indirect support through aspects such as the design of evolvable component-based architectures and technology-independent data representations, little consideration has been explicitly given to the issue of coping with rapidly changing technology.

Content is king

As has been pointed out, Web systems are largely about content! A web site that looks fantastic, and has sophisticated and powerful functionality will still be a failure if it does not have adequate and appropriate content, together with

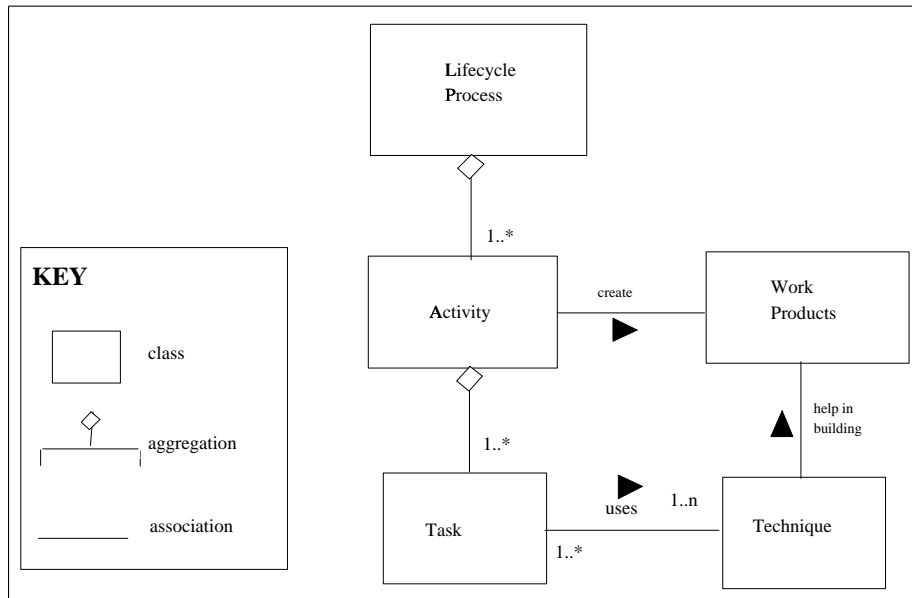


Fig. 1

SOME OF THE KEY ELEMENTS IN THE OPEN PROCESS FRAMEWORK.

mechanisms for ensuring that this content is both accessible and maintained effectively.

This raises the issue, in web development, of the ideas of both content management and information personalisation. These both represent functionality that must be included in the majority of web projects today. Since these issues can be considered to play an intricate part in the architecture of the solution, they should be actively addressed within the development process.

Approaches such as Usage Centred Design [22] provide some indications of suitable activities — though typically not as part of a broader framework. Web OPEN also addresses these issues through three tasks. Two of these relate to the planning stage: *Design and implement content management strategy* and *Design and implement personalisation strategy*, and one refers to the enactment stage of project management: *Undertake content management*.

The actual authoring of the content itself is also a significant development issue that is often overlooked. With conventional software development the population of the system with data is largely viewed as an operational issue (or at best, part of deployment). With Web development, then generation of “data” (i.e. content authoring) is fundamentally part of the development process [33]

— involving significant editing and layout of text, preparation of images and other media, obtaining copyright clearances etc. The development processes that underpin some of the information management approaches discussed earlier recognise this explicitly.

We need to ensure that content authoring is appropriately integrated into the development process. Web OPEN addresses this through the task *Create content (on web site)*. In addition, much content reuse is possible. Thus the technique *Reuse of graphical components* is also available. One particularly useful form of reuse is that provided by Web templates.

Increased emphasis on user interface

The development of the user interface raises numerous issues. Let us begin by considering how content will be combined with the interface. Effectively this brings together content authoring and software development or, more precisely, creative design and technical development. Web OPEN addresses this through the task *Integrate content with user interface*. It is worth noting that this highlights the difficulties that occur when combining two different cultures together within the same project.

We also need to consider the actual interface design. The user interface within a web project

constitutes a large portion of the overall project. It is vital in determining the success or failure of the project. OPEN has always had a task named *Design user interface*. This task however has been expanded within Web OPEN to incorporate a number of relevant subtasks. These subtasks have been taken from Constantine and Lockwood's work on Usage Centred Design (UCD) [22], which is more appropriate than the significantly different User Centred/Centric Design. UCD focuses on the work that users are trying to accomplish and how the software will support this.

The three sub-tasks to the *Design user interface* Task that were introduced in Web OPEN [31] are *Create the UCD role model*, *create the UCD task model*, and *Create the UCD content model*. The last of these subtasks links in well with the Task *Integrate content with user interface* discussed above, as it starts to identify the relationships between the content and the user interface including navigation maps (Task: *Create navigation map for web site*). All three subtasks identify how the site is to be used (hence the name Usage Centred Design) and also help to tie the user interface to the web projects requirements.

Increased importance of quality attributes

As has been mentioned previously, Web systems represent an increase in mission critical applications that are often directly accessible to external users and customers. To address these, the process needs to explicitly address quality assurance (QA) issues. Some work has been carried out looking explicitly at quality assurance issues in Web development — though in general this has been restricted to specific domains such as educational applications [34].

One key element of effective QA is evaluation. Indeed, it has been claimed that the quality of multimedia projects are directly determined by the effort put into evaluation [35]. For effective evaluation we need to establish suitable quality criteria — particularly in terms of how the Web system will be actually tested against client requirements (OPEN Task: *Define web site testing strategy*). This also implies the need to actually understand client requirements — an issue that we discuss further shortly.

Another important issue is the establishment of suitable standards in order to ensure consistency

— both from a usability perspective and from a development perspective. With this in mind Web OPEN includes a task *Define web site standards*. It is worth noting that considerable attention is beginning to focus on usability standards and, in particular, accessibility standards such as the World Wide Web Consortium's (W3C) Accessibility Initiative [36], [37].

Client uncertainty and Changing business requirements

These two issues are so interrelated at a process level that we discuss them together. We begin by looking at how conventional software processes handle requirements. Stated rather simplistically, they tend to assume that requirements are known to clients, and simply need to be elicited and analysed. These processes usually differentiate between user requirements (often formalised in a URD — or User Requirements Definition) that capture the user understanding of their needs, and the system specification (SRS — or system requirements specification) that represents the system that will meet these needs. In effect, the two documents are different representations of the same concepts. A typical process will be to elicit requirements (which are documented in the URD) and then analyse these requirements in order to construct the SRS, iterating to refine the URD as necessary.

One significant difficulty with this paradigm is that it presumes that clients either understand their requirements, or at the very least understand the problem that is being addressed. Even when clients are not able to articulate their requirements precisely, they are at least able to understand whether a given design will address their needs. In cases such as these, the design may commence prior to full resolution of requirements. The design will then be used to ascertain (from client feedback) whether the proposed solution addresses the identified need.

This is problematic for Web projects, where many clients not only have a poor understanding of their requirements but also have a poor understanding of the problems being addressed by the proposed system. In these circumstances, simply using a design to clarify whether it addresses the problem will be insufficient, since the problem itself is only poorly understood.

As an illustrative example of these issues, consider the increasing use of lightweight development processes for software projects [38], [39]. One of the approaches receiving the most attention is the use of XP (eXtreme Programming) [40]. XP is based on the incremental development of partial (or ‘spike’) solutions that are used to subsequently resolve specific requirements. These spike solutions are then integrated into the evolving system through refactoring of the current solution to incorporate these components. When used in conventional software development XP has proven to be 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 [41].

There are, however, certain problems that restrict the applicability of approaches such as these to Web projects. The first is that a number of studies (see, for example [42]) have shown that approaches such as XP only work effectively for projects that have cohesive development teams. This is often not the case with Web projects, which often lack cohesiveness between the technical development and the creative design as a result of the disparate disciplinary backgrounds of the development team members. 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.

In effect, conventional software engineering processes see requirements as preceding and driving the design process. Even where an incremental approach (such as XP) or an iterative approach (involving multiple feedback loops) is adopted, the design is viewed as a way of assisting in the identification and validation of requirements; yet rarely does it help the client to actually formulate their needs. In Web development, the situation is fun-

damentally different. The design process not only helps developers and clients articulate their needs, but also helps clients understand the system domain and therefore their needs.

In effect, the design partially drives the requirements process. We begin with a poor client understanding of their own needs (as well as system capabilities) and during the course of the project this understanding evolves and matures. This has several consequences. The first is that it increases the importance of creating flexible solutions that can be updated and migrated to new technologies with minimal effort. For example, the need for reusable data formats (such as XML) increases substantially. A second consequence is that developers’ understanding of these technologies is often restricted, increasing project risks. To address these issues, Web OPEN includes the technique *Development spikes* — though it needs to be recognised that this is utilised somewhat differently from the spikes in XP. Specifically, the spikes are used to help clients develop an understanding of the technology and support client understanding of the problem domain. Web OPEN also includes the technique *Field trips*, which are used to examine the current business environment and final place of deployment of the system — again with the intention of improving both client and developer understanding of the system context.

It is also worth looking briefly at the hourglass model of Web development [23] (see Figure 2). This model 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.

Short time frames for initial delivery

The shorter development timeframe of Web systems has a number of implications. Firstly, it increases the importance of incremental develop-

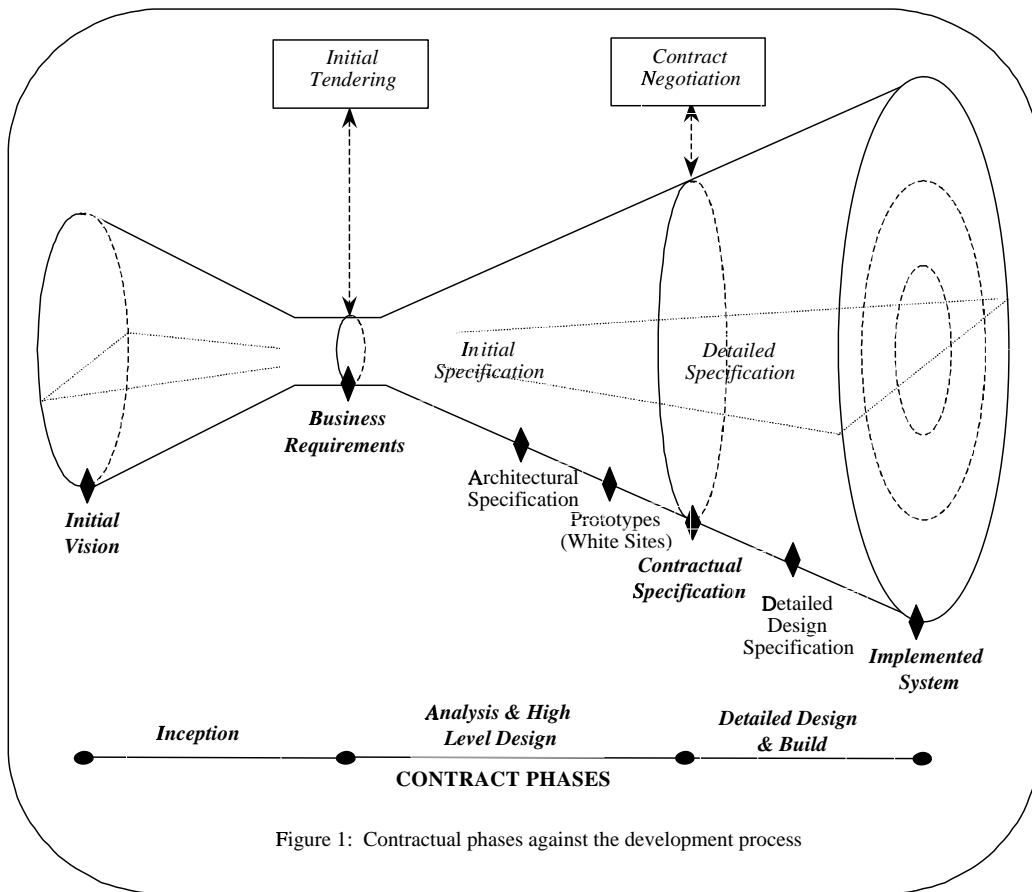


Fig. 2

THE HOURGLASS MODEL OF WEB SYSTEM DEVELOPMENT

ment approaches and consequently also increases (as discussed above) the reliance on flexible system architectures (particularly with respect to the user interface and the way in which information is managed within the site). Web OPEN addresses these issues through the introduction of the activity *Web site Management*. This brings together all the issues regarding the development, maintenance and management of a corporate web site. The objectives of the web site management activity includes creating a high quality web site; keeping the web site up to date; and ensuring that site standards are met as the web site evolves.

Indeed, probably more importantly than the actual tasks in Web OPEN is the flexible way in which processes can be constructed from the pool of tasks, activities, techniques, roles etc. This approach allows processes to be highly customised to the specific characteristics of the project — something that is highly desirable when developmental

time pressures become a major issue.

Fine-grained evolution and maintenance

The unique characteristics of Web system maintenance — and its impacts on the supporting processes — has only been very peripherally addressed in the literature. Probably the most interesting avenue of work has been that related to Configuration Management (CM). Dart [24] argues that, because of the incremental nature of Web projects, and the fine-grained way in which they change, CM is even more important than for conventional projects. Only very rudimentary consideration is, however, given to the way in which CM is integrated into the broader development process.

It is also useful to note that a consequence of the emphasis on rapid development and fine-grained development is that there can tend to be less thought to formal evaluation as this is often

perceived as interrupting the build process.

One unusual area that has been used as an analogy for web development and may provide some useful insights into maintenance processes is landscape gardening [43]. Web site development is often about creating an infrastructure (laying out the garden) and then ‘tending’ the information that grows and blooms within this garden. Over time the garden (i.e. Web site) will continue to evolve, change and grow. A good initial architecture should allow this growth to occur in a controlled and consistent manner. This analogy has been discussed in terms of providing insights into how a site might be maintained.

V. CONCLUSIONS

In this paper we have analysed the differences between Web systems and more conventional software and IT systems, focusing on the implications of these differences for the development of Web systems. In particular, we have investigated both desirable changes in the models and notations, and in the development processes.

The analysis has emphasised that although there is an emerging body of research in this area, it is still relatively fragmented and considerable work still remains to be carried out. Several areas are worth highlighting. The first is the need for a design-driven requirements process that structures the way in which design activities can be linked to the clarification of requirements through an appropriate model of domain uncertainty. The second is an improved linkage between the representation of the architectural elements of a Web system. In particular, given the evolving role of designs, it is important that the business architecture be able to be linked to both the information architecture and the technical architecture of a system. Indeed a fruitful area for further investigation is to look at how the informational and functional aspects of the architecture can be coupled appropriately during the design.

One element driving both these research areas is our ongoing refinement of a characterisation model that captures the various elements that emerge in the specification and design of Web sites. This model provides a basis for understanding which elements should be clarified at which point in the process, and the linkages between these elements.

ACKNOWLEDGMENTS

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

The first author also wish to acknowledge the collaborative funding support from the Australian Research Council, Access Online Pty Ltd and Allette Systems Ltd. under grant no. C4991-7612 which partially supported this work. In particular we wish to thank Vassiliki Elliott, John Eklund, Ross Jeffery, Nick and Marcus Carr, Louise Scott, Lucila Carvalho, and John D’Ambra for their contributions to this research project.

This is Contribution number 01/10 of the Centre for Object Technology Applications and Research.

REFERENCES

- [1] J. Burdman, *Collaborative Web Development*, Addison-Wesley, 1999.
- [2] E. England and A. Finney, *Managing Multimedia: Project Management for Interactive Media*, Addison-Wesley, 2nd edition, 1999.
- [3] S. Overmyer, “What’s different about requirements engineering for web sites?,” *Requirements Engineering Journal*, vol. 5, no. 1, pp. 62–65, 2000.
- [4] P. Russell, “Infrastructure - make or break your e-business,” in *TOOLS-Pacific 2000: Technology of Object-Oriented Languages and Systems*, Sydney, Australia, 2000.
- [5] G. Sinha, “Build a component architecture for e-commerce,” *e-Business Advisor*, 1999.
- [6] L. Stein, “Profit, the prime directive,” *WebTechniques*, vol. 5, no. 11, pp. 14–17, 2000.
- [7] J. Lord, “Patterns for e-business: Lessons learned from building successful e-business applications,” June 2000.
- [8] T. Isakowitz, E. Stohr, and P. Balasubramanian, “Rmm: A methodology for structured hypermedia design,” *Communications of the ACM*, vol. 38, no. 8, pp. 34–44, 1995.
- [9] D. Schwabe and G. Rossi, “The object-oriented hypermedia design model,” *Communications of the ACM*, vol. 38, no. 8, pp. 45–46, 1995.
- [10] D. Lange, “An object-oriented design method for hypermedia information systems,” in *HICSS-27: Proc of the Twenty Seventh Hawaii International Conference on System Sciences*, Maui, Hawaii, 1994.
- [11] S. Lee, “A structured navigation design method for intranets,” in *Third Americas Conference on Information Systems, Association for Information Systems (AIS)*, Indianapolis, 1997.
- [12] O. De Troyer and C. Leune, “Wsdm: A user-centered design method for web sites,” in *7th International World Wide Web Conference*, Brisbane, Aust, 1997.
- [13] S. Ceri, P. Fraternali, and A. Bongio, “Web modeling language (webml): a modeling language for designing web sites,” in *Proceedings of WWW9 Conference*, Amsterdam, 2000.
- [14] “OMG unified modeling language specification, version 1.3,” OMG document 99-06-09, June 1999.
- [15] H. Baumeister, N. Koch, and L. Mandel, “Towards a uml extension for hypermedia design,” in *UML 1999*, 1999, pp. 614–629.

- [16] K. Takahashi and E. Liang, "Analysis and design of web-based information systems," in *7th International World Wide Web Conference*, Brisbane, Aust, 1997.
- [17] J. Conallen, *Building Web Applications with UML*, Addison Wesley Object Technology Series. Addison-Wesley, 1st edition, 1999.
- [18] N. Guell, D. Schwabe, and P. Vilain, "Modeling interactions and navigation in web applications," in *Proceedings of the World Wide Web and Conceptual Modeling'00 Workshop - ER'00 Conference*, Salt Lake City, USA, 2000.
- [19] P. Vilain, D. Schwabe, and C. S. d. Souza, "A diagrammatic tool for representing user interaction in uml," in *UML'2000*, York, U.K., 2000.
- [20] D. German and D. Cowan, "Formalizing the specification of web applications," *Lecture Notes in Computer Science*, Springer Verlag, vol. 1727, pp. 281292, 1999.
- [21] F. B. Paulo, M. Augusto, S. Turine, M. C. F. d. Oliveira, and P. C. Masiero, "Xhmbms: A formal model to support hypermedia specification," in *Ninth ACM Conference on Hypertext*, 1998, p. 161170.
- [22] L. Constantine and L. Lockwood, *Software For Use*, Addison-Wesley, 1999.
- [23] D. Lowe, "A framework for defining acceptance criteria for web development projects," in *Second ICSE Workshop on Web Engineering*, S. Murugesan, Ed., Limerick, Ireland, 2000.
- [24] S. Dart, *Configuration Management: The Missing Link in Web Engineering*, Artech House, 2000.
- [25] D. Lowe and V. V. Elliott, "Web requirements: An overview," *Journal of the American Society for Information Science and Technology (JASIST)*, Submitted.
- [26] I. Graham, B. Henderson-Sellers, and H. Younessi, *The OPEN Process Specification*, Addison-Wesley, 1997.
- [27] B. Henderson-Sellers, A. Simons, and H. Younessi, *The OPEN Toolbox of Techniques*, Addison-Wesley, UK, 1998.
- [28] D. Firesmith, G. Hendley, S. Krutsch, and M. Stowe, *Object-Oriented Development Using OPEN: A Complete Java Application*, Addison-Wesley, Harlow, UK, 1998.
- [29] B. Henderson-Sellers and B. Unhelkar, *OPEN Modeling with UML*, Addison-Wesley, Harlow, UK, 2000.
- [30] D. Firesmith and B. Henderson-Sellers, *The OPEN Process Framework. An Introduction*, Addison-Wesley, Harlow, UK, 2001.
- [31] B. Haire, B. Henderson-Sellers, and D. Lowe, "Supporting web development in the open process: additional tasks," in *COMPSAC'2001: International Computer Software and Applications Conference*, Chicago, Illinois, USA, Submitted, IEEE Computer Society.
- [32] B. Henderson-Sellers, "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, USA, 2001.
- [33] A. Ginige, D. Lowe, and J. Robertson, "Hypermedia authoring," *IEEE Multimedia*, 1995.
- [34] J. Eklund and D. Lowe, "A quality assurance methodology for technology-delivered education and training," in *WebNet 2000: World Conference on the WWW and Internet*, G. Davies and C. Owen, Eds., San Antonio, Texas, USA, 2000, pp. 162-169, AACE: Association for Advancement of Computing in Education.
- [35] R. Philips, *The developer's handbook to interactive multimedia*, Kogan Page. London., 1997.
- [36] J. White, W. Chisholm, and G. Vanderheiden, "Web content accessibility guidelines 2.0," World Wide Web Consortium, Workig Draft WD-WCAG20-20010125, Jan. 2001.
- [37] W. Chisholm, G. Vanderheiden, and I. Jacobs, "Techniques for web content accessibility guidelines 1.0," World Wide Web Consortium, Note WAI-WEBCONTENT-TECHS-19990505, May 1999.
- [38] E. Angelique, "A lightweight development process for implementing business functions on the web," in *WebNet'99*, Honolulu, Hawaii, USA, 1999, pp. 262-267.
- [39] R. Fournier, *Methodology for Client/Server and Web Application Development*, Yourdon Press, 1999.
- [40] K. Beck, *Extreme Programming Explained*, Addison-Wesley, 1999.
- [41] D. Thomas, "Managing software development in web time software," in *XP2000*, Cagliari, Italy, 2000.
- [42] R. Martin, "A case study of xp practices at work," in *XP2000*, Cagliari, Italy, 2000.
- [43] D. Lowe, "Web engineering or web gardening?," *WebNet Journal: Internet Technologies, Applications and Issues*, vol. 2, no. 1, Jan-Mar 2000.



Associate Professor David Lowe is the Director of Undergraduate Programs in the Faculty of Engineering, and a Co-Director of the Centre for Object Technology, Applications and Research (COTAR) at the University of Technology, Sydney. He has active research interests in the areas of Web development and technologies, hypermedia, and software engineering. In particular he focuses on Web development processes

and web project specification and scoping, and information contextualisation. He has published widely in the area, including several texts (Lowe and Hall, *Hypermedia and the Web: An Engineering Approach*, Wiley, 1999 and Wilde and Lowe, *Transcending the Web: Linking and XML*, Addison-Wesley (currently in preparation)). In the last 7 years he has published over 40 refereed papers and attracted over \$ 900,000 in funding, including a recent grant for research into Web project specifications. He is on numerous Web conference committees and is the information management theme editor for the *Journal of Digital Information*. He has undertaken numerous consultancies related to software evaluation, Web development (especially project planning and evaluation) and Web technologies. A/Prof Lowe can be reached at The University of Technology, Sydney; P.O. Box 123, Broadway, NSW 2007, Australia. Telephone +61-2-95142526; Email: david.lowe@uts.edu.au.



Professor Brian Henderson-Sellers was the first Director of COTAR (1994-6, 1999-present) and is Professor of Information Systems at the University of Technology, Sydney. His interests are mainly in OO methodologies and process, metrics, project management and company migration to OO. He is involved in several metrics projects, leads the OPEN Consortium and is involved, through the Object Management Group, with the ongoing changes to UML and the new initiative towards a Software Process Engineering Model (SPEM) standard. He has published a significant number of papers and books under the auspices of COTAR as well as making a large number of international presentations. He is a columnist for the large circulation OO journal JOOP and a frequently invited speaker at industry conferences. Professor Henderson-Sellers can be reached at The University of Technology, Sydney; P.O. Box 123, Broadway, NSW 2007, Australia. Telephone +61-2-9514-1687; Email: brian@it.uts.edu.au.

agement Group, with the ongoing changes to UML and the new initiative towards a Software Process Engineering Model (SPEM) standard. He has published a significant number of papers and books under the auspices of COTAR as well as making a large number of international presentations. He is a columnist for the large circulation OO journal JOOP and a frequently invited speaker at industry conferences. Professor Henderson-Sellers can be reached at The University of Technology, Sydney; P.O. Box 123, Broadway, NSW 2007, Australia. Telephone +61-2-9514-1687; Email: brian@it.uts.edu.au.