8. Emergent Knowledge in Web Development

David Lowe

University of Technology, Sydney
PO Box 123, Broadway, NSW, 2007, Australia
email: david.lowe@uts.edu.au

ABSTRACT: Although Web development can be considered a derivative of software engineering, it exemplifies a class of development projects that has some unique characteristics that lead to changes in the development approach. Amongst other factors, there is substantial volatility in clients' articulation of their requirements, particularly as their understanding evolves of the way in which the systems under development might affect their client and stakeholder interactions, business processes, and ultimately their business model. We discuss these differences and the impact that they have on the development processes that are adopted for commercial Web systems. Specifically, we look at the ways in which client knowledge (and understanding) emerges progressively during the development process, often as a consequence of the design process, and the ways in which this results in a design-driven requirements process.

KEYWORDS: Web development; process; design; requirements

8.1 Introduction

Web systems were originally (i.e. in the early to mid 1990's) characterized by a strong emphasis on content and information provision. As such, they were often viewed not as software systems but as information systems. This characterization was evidenced in the focus of most of the early Web design methods (such as RMM [24] and OOHDM [41]) that emerged out of the hypertext community and emphasized content modeling and information structuring.

As Web technologies matured and became more sophisticated, the systems being developed exhibited increasingly complex functionality and consequently more complex underlying software. Again, this was typified by the emergence of web design methods that aligned more closely with mainstream software design approaches (such as a plethora of approaches based on UML – see [2, 8, 22, 25, 30] for examples) and an increasing debate over whether "Web engineering" can be viewed as a particular class of software engineering (see [38, ch. 29] for a discussion of this issue).

Whilst it is true to a limited extent that Web system development is primarily the creation of software systems, there is a growing recognition that Web systems – or rather that category of applications for which Web systems are an exemplar –

have various unique characteristics that are only poorly addressed by conventional development practices [31]. Amongst other factors, there is substantial uncertainty in clients' understanding of the ways in which the systems under development might affect their client and stakeholder interactions, business processes, and ultimately their business model. This, in turn, has some major implications for the ways in which, and particularly when, clients' are able to articulate their requirements during the development process.

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.

In this chapter we begin by investigating some of main differences between Web systems and other software systems. We then move on to explore the implications of the key differences for the ways in which client's knowledge evolves during the development process and how this should be addressed. We will, in particular, look at the role that the design process plays in this evolving understanding.

Before starting to look at Web systems in more detail one point of clarification is worth raising. Whilst we use the term *web system* in this paper for simplicity, we see these systems (i.e. those that have an architecture based on the utilization of Web technologies and protocols) as being exemplars of a much broader category of applications. This broader category can be understood by looking at the characteristics discussed in the next section, but can probably be best defined by one key characteristic – that the system under development changes the nature of the interaction with external stakeholders (such as clients, customers and business partners) and hence potentially triggers changes in business processes and ultimately business models. In other words, the solution under development inherently changes the nature of the problem which it was addressing. This can be described as the *problem domain and the solution domain being mutually constituted* – a concept that is well understood in the social informatics literature! We will discuss this is much more depth later in the chapter, but at this point it is simply worth noting that where we refer to *web systems*, this broader interpretation will often be applicable.

8.2 Web System Characteristics and Implications

There is a growing body of research [5, 13, 35] that is attempting to understand the differences between Web systems and more conventional software systems (i.e. given the above comments at the end of the introduction, we describe as conventional systems those which have minimal impact on the fundamental nature of the interactions with external stakeholders and/or the nature of the problem being addressed). In general, we can draw a distinction between the unique characteristics of Web 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 organizational (i.e. related to the ways in which organizations make use of these systems).

It is also worth noting that 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 utilized – 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.

Further, improving the modeling 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 organize this work.

We shall look at the various unique characteristics of Web systems, and investigate the impacts on both what we may wish to represent and potential changes to the development process.

8.2.1 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 technical 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 [39]. The linkage between the business architecture and the technical design of the system is much tighter than for conventional software systems (i.e. the technology is more visible to users and influences an organizations interaction with its stakeholders very significantly). 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.

The impact 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 [28]:

> "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 emphasize functionality, with little consideration of how to represent the information architecture. In particular, aspects such as content modeling, information viewpoints etc. are not addressed.

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. Even more significantly, there is little understanding of the impacts of a given architecture back on the business processes and models. 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 mentioned above. 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 modularized architectures

Related to the above point is the emphasis that is typically placed on open and modularized 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.

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

Given this component-based development, 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. 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) [19].

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.

The work on detailed design notations for representing certain aspects of Web systems 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 dependent on specific technologies, and hence more likely to be able to be adapted to changes.

Content is king

Of notable significant 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. This 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 modeling and structuring.

Early approaches in this area evolved out of work on data modeling (such as Entity-Relationship models) and applied this to modeling 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 [24]) claims to provide a structured design model for hypermedia applications. In reality, the focus is very much on modeling the underlying content, the user viewpoints onto this content and the navigational structures that interlink the content. OOHDM (Object-Oriented Hypermedia Design Model [42]) is a similar approach, though somewhat richer in terms of the information representations and based on object-oriented software modeling approaches. Other similar examples include EORM [26] and work by Lee [27]. WSDM [11] 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 modeling information in the context of the Web, or have been adapted to this domain.

More recently, work on both WebML (Web modeling Language [6]) and the adaptation of UML (Unified modeling Language [34] – an emerging industry standard for modeling object-oriented systems – see for example [3]) has begun to amalgamate these concepts into a richer modeling language for describing Web applications. However, despite aims to support comprehensive descriptions, the

focus (as with the above techniques) is very much on content modeling rather than describing the functionality that is a key element of most current commercial Web systems. This leads on to the next point.

Even less consideration has been given to process related issues in terms of dealing with content. Approaches such as Usage Centered Design [9] provide some indications of suitable activities – though typically not as part of a broader framework. 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 [18] – 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 recognize this explicitly.

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, it becomes much more critical to engage users and provide much more evident satisfaction of users' needs and achievement of their objectives. The result is an increased emphasis on the user interface and its associated functionality. This is even more significant when it is recognized that many direct users of the systems are external rather than internal stakeholders.

A little more subtly, the emergence of authoring tools has 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.

A key element of user interfaces is the functionality that they provide. A few attempts have been made to integrate information modeling concepts with system functionality [8, 45] though in general these approaches are still rather simplistic, lack scalability and focus on low-level design representations. Conallen's [8] work in particular is interesting insofar as it attempt to link a user'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 modeling the way in which systems are utilized. For example, Guell et al [20] extend OOHDM to include tools such as user scenarios and use cases. Vilain et al. [47] has adapted UML to representing user interactions. Other researchers have investigated the use of formal methods for representing navigational requirements [17] or timing constraints [36] – 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 centered design [9], although a rigorous analysis of the application of these techniques to Web development has yet to be carried out.

The development process for user interface also raises numerous issues. Effectively this brings together content authoring and software development or, more precisely, creative design and technical development. It is worth noting that this highlights the difficulties that occur when combining two different cultures together within the same project.

Increased importance of quality attributes

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

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

In terms of development processes, there is a need to 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 [12]. 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 [37]. 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. 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. 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 [7].

8.2.2 Organizational Differences

In addition to the technical differences, and possibly more important than them, are a number of organizational characteristics that are either unique or heightened in Web systems [5]. One of the key ones is the issue of client uncertainty. This however relates strongly to how client and developer knowledge emerges during the project, and so will be discussed in the following section. Various other issues are worth briefly considering.

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. This is an issue that has yet to be considered in any substantive way in terms of how it impacts on Web design models and notations.

In terms of processes, the shorter development timeframes increases the importance of incremental development 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).

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.

As with many other aspects, this has yet to be considered in any substantial detail. It is worth pointing out, however, that one aspect of modeling that actively inhibits effective Web system maintenance is the lack of a cohesive architectural modeling language that actively links the information architecture with the technical architecture [19]. Conversely, the information models (such as OOHDM [42] and WebML [6]) actively support a much clearer understanding of the impacts of changes to various aspects of the underlying content, viewpoints or navigational structures.

One interesting avenue of work has been that related to Configuration Management (CM). Dart [10] 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 given 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 garden-

ing [30]. 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.

8.3 Evolving Project Knowledge

The above discussion has highlighted various aspects that characterize Web development. Few, if any of these characteristics, are unique to Web projects. When taken as a whole they tend, however, to characterize these projects.

There is however a characteristic that was skimmed over, but is much more significant in the overall impact that it is likely to have on the development process. This characteristic is the impact that a developed system has on the nature of the problem being addressed and how this relates to client uncertainty and emerging knowledge. As we stated in the introduction, the solution being developed inherently changes the nature of the problem which it is addressing – i.e. the *problem domain and the solution domain are mutually constituted and interdependent*! This will affect not only the way in which the solution is developed, but more fundamentally the way in which the problem itself is understood (and indeed, how this understanding changes over time).

Whilst there has been substantial work on using the Web to manage knowledge whilst carrying out development projects, there has been very little consideration given to how knowledge *about* web systems emerges and is managed during development. To understand this a little better, we begin by considering the issue of client uncertainty and requirements volatility.

8.3.1 Client uncertainty

It is often argued that with internet and Web-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 [43]. This can lead to a client not understanding their needs. Specifically, clients often have difficulty not only articulating their needs, but also in understanding whether a particular design will satisfy their needs. This is typically a result of a poor understanding of the consequences of the given solution. It is also worth noting that many web projects are vision-driven rather than needs-driven leading to an initial lack of clarity.

This interpretation is, however, a little simplistic. More commonly, clients will have sound knowledge about their own (current) business models, contexts, processes – and hence the problem to which they are seeking a solution. Whilst it is true that they may have difficulties in articulating this knowledge, there is a pleth-

ora of work in the requirements engineering domain about how this particular challenge can be addressed. Where a greater challenge arises is the situation where a client does not initially comprehend that a given problem definition will result in a solution that has impacts beyond the confines of the problem as defined – i.e. a possible solution that adequately addresses the problem as defined by the client will change or impact on other elements of the clients business model, processes or context. In this situation, the client's knowledge of the solution impacts only emerges progressively as possible designs are created by the developer and jointly explored [44].

An alternative way of conceptualizing this is that the underpinning technology that enables the solution implies certain linkages between different aspects of the solution, and so when one of these aspects is addressed by a solution, the other elements are also affected. This can possibly be clarified with a simple example. Consider an existing company that does event promotion by regularly collecting information from event venues and using this to construct promotional posters for distribution – with advertising space available to generate an income stream. Developing a web-based system to support distribution of the event information may seem like a relatively straightforward extension of existing business models and processes, but the interaction with the customer base (i.e. event patrons) and advertisers is changed by the nature of the Web. Specifically, it is likely that the patrons will have new expectations regarding the ability to dynamically provide feedback on events, which in turn will change the value of this information. Advertisers will perceive differing value in a transient online presence as compared to more permanent hardcopy advertising material. In other words, the solution that is constructed will change the value chains that exist in the business and possible even ultimately the business model itself. The client's knowledge regarding these changes will only develop once the system itself takes form and can be used to gain feedback.

8.3.2 Addressing Client Uncertainty and Understanding Requirements

So, client uncertainty largely arises from a lack of understanding of the likely broader impact on business problems of addressing a given set of business needs, and client knowledge about their evolving needs emerges progressively during the development. How is this issue addressed by current approaches? A useful place to start in understanding this issue is to look at how requirements are handled in Web projects. Stated rather simplistically, conventional development tends to assume that requirements are known to clients, and simply need to be elicited and analyzed. Requirements processes usually differentiate (at least conceptually, if not in the way they are represented) between user requirements that capture the user understanding of their needs and the system specification that represents the system that will meet these needs. The user requirements are often elicited and formalized in a URD (or User Requirements Definition) and then analyzed to construct the system requirements which are formalized in a SRS (or system require-

ments specification). In effect, the two documents are different representations of the same concepts.

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 and can be led through a process of articulating their needs. 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.

Given the characteristics of Web projects that have been outlined, this will problematic. A fundamental problem arises out of the evolving client knowledge about the changes to the problem domain – and the fact that this evolving knowledge is actually triggered by the system designs, prototypes and implementations.

Turning this around, we can see that it becomes impractical to resolve the requirements (which in essence are an articulation of what needs to be done to address the problem domain) without an understanding of the proposed solution domain. In our research work we have been referring to this is a design-driven requirements process [32]. An interesting analogy is found in the area of social informatics [40], which encompasses the concept that technology and the use of that technology are mutually constituted – i.e. the desired use defines the desired technological solution, but the actual solution changes the usage. Web systems could be described as an exemplar of that class of systems where the system and the problem domain are mutually constituted.

Whilst there has been little work addressing this specific issue, some of the techniques mentioned above that focus on modeling the way in which systems are utilized [20, 47] may help reduce client uncertainty and allow clients to obtain a clearer view of potential changes to their businesses.

One avenue being pursued by the authors is the investigation of a characterization model that represents the key aspects that need to be woven into an evolving specification of a Web system [29] (see Table 1 for an example). The complete form of the 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.

8.3.3 Development Processes

So what development approach can be used to address this "design-driven requirements" process and assist clients in constructing knowledge about the impacts of the solutions being developed? We can begin by considering the increasing use of lightweight development processes for software projects [1, 15].

Table 1. Acceptance Criteria Framework

Dimension	Possible Representations	Example Elements
Client/User		
Client problem statement	(Natural language)	
Product vision	(Natural language)	Client needs and business objectives
Users	(Natural language)	User descriptions and models
Application		
Content modeling	Structured language, hypermedia / information modeling languages (OOHDM, HDM, entity modeling, etc.)	Existing content structure, Information views, Navigational structures, Required content
User interaction	Modified TAM	Usability and usefulness metrics
	Structured language, hypermedia modeling, HCI models, etc	Access mechanisms, user control behaviour, user orientation, search requirements, security control
Development Constraints	Natural language, standards	Adherence to corporate policies, Resource availability
Non-functional requirements	Natural language, quality metrics, adherence to standards	Reliability of content, Copyright constraints
Application Evolution		
Evolution directions	(Natural language)	Expected content changes
Client adoption/ integration of Web	Business Process Reengineering	Information dissemination paths, Workflow changes
Maintenance processes	Natural language, process models	Content maint. responsibility, Web management cycles

One of the approaches receiving the most attention is the use of XP (eXtreme Programming) [4]. XP is based on the incremental development of partial solutions that address component requirements. These partial 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 (arguably) proven to be effective for projects that are initially ill-defined – a characteristic of many web projects. This is possibly because it allows a client to see the emerging solution early in the development when further clarification of the requirements is still possible. As a result, many of the proponents of XP and similar approaches see them as ideal to be adopted for Web development [46]. In effect, the emerging solution will facilitate the development of client knowledge about the impacts of the solutions, and allow the refinement of the system definition early in the development.

It can be argued, however, that there are certain problems that restrict the applicability of approaches such as these to Web projects (see, for example [33]). The first is that a number of studies 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 utilizes 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. In other words, the incremental development implicit in XP can be viewed as a form of prototyping that aims to either consider the applicability of a given design to a known problem, or to assist the developers in ensuring that they have understood the clients' problem. The prototyping in Web development however aims to help a client develop an understanding of how different solutions may impact on the nature of the problem being addressed.

A useful divergence at this point is to consider a comparison with the approach that is often referred to as "Ready-Fire-Aim" [23]. This essentially is referring to approaches where the design is commenced prior to a full understanding of the requirements (or coding commenced prior to a full design, depending on the interpretation) as a way of informing clients in the presence of uncertainty. In contrast, commercial Web development is typically about developing prototype solutions as a way not of resolving initial uncertainty, but rather to understand the impact of a given solution. This is a little bit like saying "Well, if we fire there, then it will have this impact, but if we fire there it will have that impact". i.e. Possible solutions are jointly investigated by the developer and client (typically, through a design prototyping approach – but *prior* to committing to a specific solution) in terms of their impact on the problem domain and hence the requirements, with the ultimate result that a solution is identified that matches a problem that has been changed by that solution.

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 fundamentally 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 drives the requirements process. We begin with a poor client understanding of their 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.
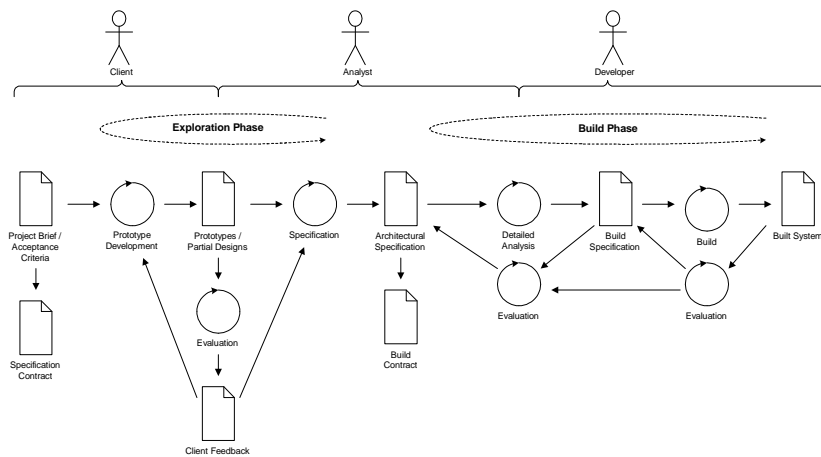

Fig. 8.1. Typical Web Development Process

Figure 1 shows a depiction of a development process for Web systems that incorporates this understanding. In this figure, the first cycle iterates around a series of exploratory design prototypes, including elements such as white sites and storyboards, with the aim of moving from an initial set of acceptance criteria to a clear specification of the system – but a specification that includes not only requirements but also the broad architectural design elements of the site [16, 21]. The second cycle covers the (usually fine-grained incremental) design and build process. In effect, the process (specifically the first of the two key cycles shown in Figure 1) is aimed at developing (or rather evolving) a joint understanding of the combined problem/solution domain.

Finally, it is worth noting that anecdotal evidence indicates that these issues are well understood and accepted within industry. Research has been limited to empirical work using scenario-based redesign of partially developed sites [14] though this work has at least recognized the importance of designs in assisting clarification of client needs:

> "We practice a revised method of scenario-based design inferred from a theoretical perspective which treats design as inquiry, inquiry as dialogue and dialogue as the source of all tools, including mental constructs. The result is a set of techniques for using structured dialogue between users and designers to increase designers' understanding of specific domains of users' work."

In commercial Web projects, these concepts (and particularly the mutual interdependence of requirements and design) are typically reflected in the absence of separate requirements and design documents. Rather, developers tend to create a hybrid specification that blends design and requirements (something that is usually viewed as anathema in conventional software engineering).

In other words, system design allows stakeholders to understand technical possibilities and limitations, and hence improve their understanding of the development context. The result is a vehicle for reducing the underlying uncertainty. For this to be effective, however, we need to develop a suitable model of the relationship between system design, client requirements, and uncertainty within these requirements. This uncertainty model can then be used to adapt the requirements engineering process – resulting in a design-driven requirements process. This is the focus of our ongoing research.

8.5 Future Trends and Conclusions

So what conclusions can we draw from the above discussions regarding how knowledge is managed in Web projects? The key insight is that the nature of Web projects implies that since the solution changes the nature of the problem we therefore need to acknowledge that a client will be inherently unable to define their problem in the absence of a possible solution. Different solutions (i.e. the Web systems to be developed) will fundamentally lead to differing impacts on the stakeholder interactions and business processes and hence to different problem domains. This in turn means that we need to recognize the importance of exploring a range of possible solutions – and to do so not only to determine the optimal design, but possibly to determine the optimal problem!

Further, it also indicates that client involvement in the design process becomes crucial (something that is often viewed as very dangerous). Without an understanding of the possible system designs, the client is unlikely to develop a clear understanding of the implications of a proposed solution. Thus design knowledge becomes a crucial enabling tool within Web projects.

Ongoing work of the author and others has begun to explore exactly what level and form of design knowledge will best assist clients in developing a clear con-

ceptualization of the impact of possible designs. This work is, however, still too early to have provided concrete outcomes.

Another project that is only just commencing is looking at process modeling and project management tools that track the evolving process that accompanies the evolving product understanding. By monitoring the relationships between these models (often expressed as project plans) and the initial templates from which they were derived it is possible to identify the points at which the process deviated. Once this is identified, the developer can be interrogated as to the cause of the deviation, and this information then fed back into the underlying project templates to support future project planning. This approach becomes much more crucial in Web projects where the nature of the process is difficult to determine *a priori* because of the evolving system.

Ultimately, the insights explored in this paper are not only about Web projects, but rather about those systems where, as we mentioned, the *solution and the problem* are mutually constituted – i.e. neither can exist without the other, and they need to be jointly understood, developed, and evolved.

Acknowledgements

References

8.1.   Angelique, E. (1999), *A Lightweight Development Process for Implementing Business Functions on the Web*. in *WebNet'99*. Honolulu, Hawaii, USA.

8.2.   Baresi, L., F. Garzotto, and P. Paolini (2001), *Extending UML for Modeling Web Applications*. in *34th Hawaii International Conference on System Sciences*. Hawaii, USA.

8.3.   Baumeister, H., N. Koch, and L. Mandel (1999), *Towards a UML Extension for Hypermedia Design*. in *<<UML>> 1999: The Second International Conference on The Unified Modeling Language*. Fort Collins, Colorado, USA: IEEE.

8.4.   Beck, K. (1999), *Extreme Programming Explained*: Addison-Wesley.

8.5.   Burdman, J. (1999), *Collaborative Web Development*: Addison-Wesley.

8.6.   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.

8.7.   Chisholm, W., G. Vanderheiden, and I. Jacobs (1999), *Web Content Accessibility Guidelines 1.0*. World Wide Web Consortium.

8.8.   Conallen, J. (1999), *Building Web Applications with UML*. Addison Wesley Object Technology Series: Addison-Wesley. 336.

8.9.   Constantine, L.L. and L.A.D. Lockwood (1999), *Software For Use*: Addison-Wesley. 579.

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

8.11.   De Troyer, O. and C. Leune (1997), *WSDM: A user-centered design method for Web sites*. in *7th International World Wide Web Conference*. Brisbane, Aust: Elsevier.

8.12.   Eklund, J. and D. Lowe (2000), *A quality assurance methodology for technology-delivered education and training*. in *WebNet 2000: World Conference on the WWW and Internet*. San Antonio, Texas, USA: AACE: Association for Advancement of Computing in Education.

8.13.   England, E. and A. Finney (1999), *Managing Multimedia: Project Management for Interactive Media*. 2nd ed: Addison-Wesley.

8.14.   Erskine, L., D. Carter-Tod, and J. Burton (1997), *Dialogical techniques for the design of web sites.* International Journal of Human-Computer Studies, 47: p. 169-195.

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

8.16.   Gates, L. (2001), *Analysis and Design: Critical yet Complicated*, in *Application Development Trends*. p. 40-42.

8.17.   German, D.M. and D.D. Cowan (1999), *Formalizing the specification of Web applications.* Lecture Notes in Computer Science, Springer Verlag, 1727: p. 281–292.

8.18.   Ginige, A., D. Lowe, and J. Robertson (1995), *Hypermedia Authoring.* IEEE Multimedia, Winter 1995.

8.19.   Gu, A., D. Lowe, and B. Henderson-Sellers (2002), *Linking Modelling Capabilities and Abstraction Levels: The Key to Web System Architectural Integrity*. in *WWW'2002: The Eleventh International World Wide Web Conference*. Hawaii, USA: ACM Press.

8.20.   Guell, N., D. Schwabe, and P. Vilain (2000), *Modeling Interactions and Navigation in Web Applications*. in *World Wild Web and Conceptual Modeling'00 Workshop - ER'00 Conference*. Salt Lake City, USA.

8.21.   Haggard, M. (1998), *Survival Guide to Web Site Development*: Microsoft Press. 225.

8.22.   Hennicker, R. and N. Koch (2001), *Systematic Design of Web Applications with UML*, in *Unified Modeling Language: Systems Analysis, Design and Development Issues*, K. Siau and T. Halpin, Editors. IDEA Group Publishing.

8.23. Holtzman, J.K. (1993), *Ready. Fire!! Aim???* in *Proceedings of the 11th annual international conference on Systems documentation*. Waterloo, Canada: ACM Press.

8.24. Isakowitz, T., E. Stohr, and P. Balasubramanian (1995), *RMM: A Methodology for Structured Hypermedia Design*. Communications of the ACM, 38(8): p. 34-44.

8.25. Koch, N. and A. Kraus (2002), *The expressive Power of UML-based Web Engineering*. in *Second International Workshop on Web-oriented Software Technology (IWWOST2)*. Malaga, Spain.

8.26. Lange, D. (1994), *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.

8.27. Lee, S.C. (1997), *A Structured Navagation Design Method For Intranets*. in *Third Americas Conference on Information Systems, Association for Information Systems (AIS)*. Indianapolis.

8.28. Lord, J. (2000), *Patterns for e-business: Lessons learned from building successful e-business applications*. IBM. p. 4.

8.29. Lowe, D. (2000), *A Framework for Defining Acceptance Criteria for Web Development Projects*. in *Second ICSE Workshop on Web Engineering*. Limerick, Ireland.

8.30. Lowe, D. (2000), *Web Engineering or Web Gardening?*, in *WebNet Journal: Internet Technologies, Applications and Issues*.

8.31. Lowe, D. and B. Henderson-Sellers (2001), *Web Development: Addressing Process Differences*. Cutter IT Journal.

8.32. Lowe, D. and J. Eklund (2002 (Accepted)), *Client Needs and the Design Process in Web Projects*. Journal of Web Engineering, 1(1).

8.33. Martin, R. (2000), *A Case study of XP practices at work*. in *XP2000*. Cagliari, Italy.

8.34. OMG (2000), *OMG Unified Modeling Language Specification, Version 1.3 (released to the general public as OMG document formal/00-03-01 in March 2000)*.

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

8.36. Paulo, F.B., et al. (1998), *XHMBS: A formal model to support hypermedia specification*. in *Ninth ACM Conference on Hypertext*: Association of Computing Machinery.

8.37. Philips, R. (1997), *The developer's handbook to interactive multimedia*: Kogan Page. London.

8.38. Pressman, R. (2001), *Software Engineering: A Practitioner's Approach*. 5th ed: McGraw Hill.

8.39. Russell, P. (2000), *Infrastructure - Make or Break your E-Business*. in *TOOLS-Pacific 2000: Technology of Object-Oriented Languages and Systems*. Sydney, Australia.

8.40. Sawyer, S. and H. Rosenbaum (2000), *Social Informatics in the Information Sciences: Current (2000)*. Informing Science, 3(2): p. 89-96.

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

8.42. Schwabe, D. and G. Rossi (1998), *Developing Hypermedia Applications using OOHDM*. in *Workshop on Hypermedia Development Processes, Methods and Models (Hypertext'98)*. Pittsburgh, USA.

8.43. Sinha, G. (1999), *Build a Component Architecture for E-Commerce.* e-Business Advisor, March.

8.44. Stein, L.D. (2000), *Profit, the Prime Directive.* WebTechniques, 5(11): p. 14-17.

8.45. Takahashi, K. and E. Liang (1997), *Analysis and Design of Web-based Information Systems*. in *7th International World Wide Web Conference*. Brisbane, Aust.

8.46. Thomas, D. (2000), *Managing Software Development in Web Time Software*. in *XP2000*. Cagliari, Italy.

8.47. Vilain, P., D. Schwabe, and C.S.d. Souza (2000), *A Diagrammatic Tool for Representing User Interaction in UML*. in *<<UML>>2000: The Third International Conference on The Unified Modeling Language*. York, U.K.: IEEE.