THE UNIVERSITY OF HULL

Web Engineering Literature Review Version 0.05

Darren Stephens
Centre for Internet Computing
University of Hull

December 19 2001

“If it looks like a duck, walks like a duck, and quacks like a duck, it’s a duck.”
- *proverb*

Contents

List of Figures

1 Introduction

The discipline of "web engineering" is a relatively new one. The term seems not to have been in (wide) use before 1998 and the 9th ACM Conference on Hypertext and Hypermedia [Bieber1998], although Murugesan claims to have used the term as early as 1997[WebE2001]. This is in contrast to the more general area of hypermedia engineering, which had been defined at least as far back as the early 1990s. A similar term "Web Document Engineering" had been used earlier, in 1996 by Bebo White of SLAC (q.v) in a session at the 5th World Wide Web Conference [1] in 1996 [White1996].

Given the relative immaturity of the area of web engineering, much of the early literature consists mainly of position papers and overviews of the problem domain of engineering quality in web projects. Many of these papers also describe issues of macroscopic structure, the need for component re-use and process engineering and the problems of mapping the limits of web sites. Other early works such as Powell [Powell1998a] acknowledge the rapidly changing nature of web projects and attempt to codify some ground rules for working on such projects.

As an aid to understanding the need for more structured methods for the creation of web projects, it is, perhaps, useful to refer to Powell, who divides the evolution of web applications into a number of generations, illustrated in Figure 1 (shown overleaf).

The generations can be explained in the following way:

1. The first generation predates the development of the Mosaic browser and inline images. Browsers of this nature were text-based systems. Mark-up existed purely to define content, not its presentation. The first example of this type was the initial site at CERN in Switzerland (http://info.cern.ch/), the first public web server.

2. The second generation, like the first, was simple, mainly consisting of static webpages and very basic CGI processes. These initial projects were usually ad hoc and lowly budgeted. Extra functions were generally

[1]This conference was held in May of 1996, in Paris, France. The conference programme is available on-line at: http://www5conf.inria.fr/

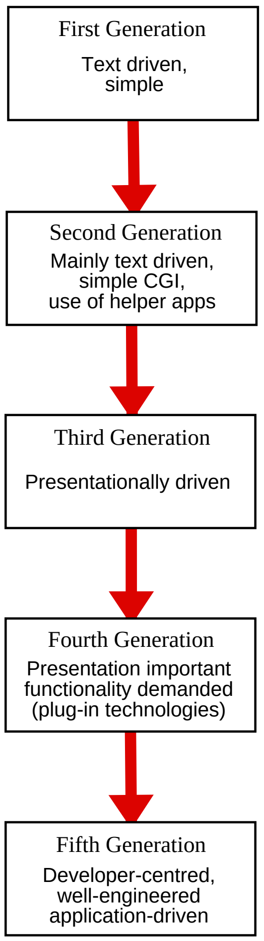
First Generation
Text driven, simple
Second Generation
Mainly text driven, simple CGI, use of helper apps
Third Generation
Presentationally driven
Fourth Generation
Presentation important functionality demanded (plug-in technologies)
Fifth Generation
Developer-centred, well-engineered application-driven

Figure 1: Evolution of Web Projects

provided by the use of helper applications that needed to be configured at the client.

3. In the third generation, begun by the release of Netscape Navigator, presentational issues assumed greater importance. As Powell observes, many third generation sites were driven more by appearance than usability. This was an opinion shared by Jakob Nielsen in his regular surveys of web usability.

4. In the fourth generation, users became used to "pretty" sites and de-

manded more functionality. This was provided by additional content, the use of multimedia and technologies such as Java to provide interactivity and increased functionality. Much of this was (and is) provided by 'plug-in' technologies or the use or scripting and other client side technologies.

5. The web now appears to be entering a fifth generation. This generation is driven, not by user requirements but by those of developers. The growing size and complexity of web systems and the tasks they are being designed to perform is the cause of this evolution.

Powell refers to this most recent generation of web design as **Software Centred Web Site Development**, making direct comparisons with software development while acknowledging differences, most particularly in terms of timescales and lifecycles. Once again, there is a return to using markup purely to define content, separating it from presentation, especially now through the burgeoning use of XML and its applications. Berners-Lee talks about this at some length in his book 'Weaving The Web' [Berners-Lee2000]. This change is as a result of the growing need for interoperability between systems as Internet platforms become more fragmented [2]. The size of sites also forces developers to implement this type of more regular design.

[2]The increasing number of versions of the Windows platform as well as the number of -not always-compatible Unix variants is evidence of this

2 An Introductory Classification of Current Work

Warren et al. [Warren2001a], in a paper presented at WSE2001, have already proposed a basis for understanding and classifying research being done on engineering and metrics for the World Wide Web. According to them, much of the current work in web engineering is based around three major areas, viz.:

1. **Software Metrics**
   The area of software quality measurement is an enormous one. A comprehensive review is well beyond the terms of this document. Readers are advised to refer to the work of both Pressman and Somerville [Somerville1989] for an introduction to the area of software metrics. In his WSE2001 paper, Warren describes the use of systems such as GQM (Goal-Question-Metric) [Berghout1999] to define general principles for creation of metrics. The application of this has been principally in software systems and the possibility exists that at least some of these can be extended to the web. There is an underlying assumption that web systems can be described as 'software-like' in nature. This is by no means obvious.

2. **Hypertext**
   Web systems are generally seen to be specialisms of more general hypertext systems. Current work in this area may have impact on the more specialist one. Of particular interest are applications and metrics derived from the use of methods such as OOHDM [Schwabe1995] and RMM [Isakowitz1995], the most common in use. This is in addition to the original works of Hatzimanikatis et al [Hatz1995]. and Botafogo et al. [Botafogo1992], which attempted to address the problem of defining metrics for idealised hypermedia systems. This area also describes document engineering, which can, with some justification, be described as a subset of hypermedia engineering in general. Latterly, the work of Mendes et al. [Mendes2001] attempts to define and measure metrics by means of case studies in hypermedia development.

3. **Human Computer Interaction**
   A great deal of work has been done in the area of HCI on the web, espe-

cially (although not exclusively) by Jakob Nielsen [Nielsen2000],[Nielsen2001], who has a long history in the investigation of usability engineering. Some may argue that any study of engineering for the web must give great consideration to user interaction because of the highly user-centric nature of any systems developed. This is a valid observation, but not necessarily of direct concern when considering the production of systems in the first instance, for example. Although this field is without doubt of interest it will not be discussed at any further length in this document.

It appears that, at present, web development is being equated directly with software developement, with little thought applied to considering both software and mark-up based systems in a wider context. This lack of a wider view can be seen as recently as 1996 when authors such as Stross [Stross1996] could say with some confidence:

> "A web is a publication, not a piece of software."

This view is one that is quite clearly becoming outdated. Web projects are growing rapidly, both in size and complexity. This can be seen in two ways:

1. The number of pages on the web is rising very rapidly, faster than the number of web users indicated by recent surveys. Google estimates this figure at 1,610,476,000 'pages' as at November 16 2001.[Google2001] This figure has grown by around 60% in less than a year. Nielsen estimated that there were around 100 million web sites in January 2000. This number would rise to 25 million by the end of that year and then further to around 100 million by the end of 2002. Conversely, Internet usage in the UK at least had fallen very slightly in mid 2001, according to a recent survey by the UK Office of Telecommunications (Oftel) [Oftel2001]. It appears that internet usage, in the short term at least appears to be reaching a plateau, but the number of pages on the web is growing at great speed.

2. The number and scope of content development and delivery tools for the web is growing extremely rapidly and significant communities are rising up to exploit them (e.g., PHP, ASP, ColdFusion, SourceForge, Zope etc.)

The process of producing such systems, marrying the disparate technologies they use and for measuring their quality is becoming rapidly ever more complex. Already there is an appreciation that this increase in the size and complexity of web projects is a burgeoning problem. A so-called “web crisis” [Zelnick1998], mirroring the “software crisis” that was seen in software in the 1970s and 1980s. The concerns of practitioners of that time, such as Djikstra, Parnas and Wirth, calling for increased refinement, modularisation and re-use are finding resonance with current web developers. Many of the overview and position documents identify this parallel between current web development and earlier software engineering.

2.1 Overviews of the Web Engineering Problem Domain

As has previously been alluded to, managing the production and maintenance of web systems is a non-trivial task, analogous to the management of software systems. This issue is explored in early work by Murugesan et al. [Murugesan1999], defining the problem domain of web engineering. The theme is continued by Deshpande and Hansen [Deshpande2001]. Both works make a distinction between ‘traditional’ software and web-based systems. These differences amount to a combination of three main factors:

1. The importance of the user interface - more so than ‘usual’ software.
2. High Network Overhead - web systems are network intensive in general.
3. Heterogeneous components - web projects are likely to comprise markup, program code and other types of content, typically multimedia.

Each of these can be seen in other types of software development but a combination of these factors, as well as hugely compressed lifecycles for developing such projects, sets them apart.

These works put the case that web engineering is a discipline of itself and should be treated as such, separately from software engineering. To illustrate how the areas within Web Engineering are connected to other disciplines, a broad illustration of the area is presented ‘ in Figure 2.1.

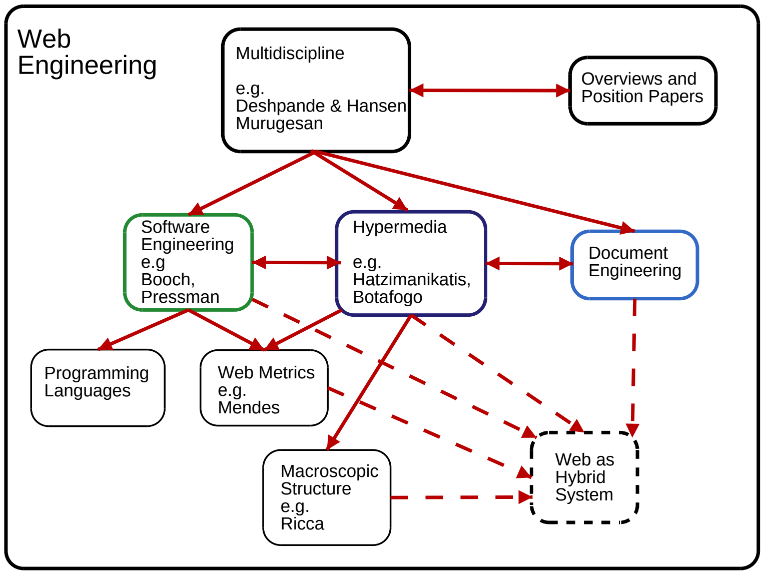

Figure 2: An Overview of Web Engineering Literature

Figure 2.1. shows a brief overview of current and recent work in the web engineering field. There are a number of disparate areas, with between them shown by solid lines. The dotted lines and boxes show connections to this document and to the so far seemingly unexplored area of hybrid web systems. Each of the areas shown will be considered at various points later in this document.

Importantly, Roger Pressman, in an article entitled “What a Tangled Web We Weave” [Pressman2000b] uses the phrase “software-related” to describe what he calls “WebApps” (what this paper describes as ’webware’).

The implication of this article is that he does not see webware systems as *purely* software systems. This is an issue that the Warren WSE paper also addresses, stating that the “software-like” nature of web systems has yet to be established fully. Sensibly, Pressman contends, like others in the area, that a disciplined engineering approach is essential for the creation an maintenance of useful web systems.

To define ‘disciplined’ he cites the following criteria:

1. Problem solving
2. Good Design
3. Thorough Testing
4. Maintenance

He clearly equates these with the well-established criteria applied in software enginering. Pressman is not the only person to make this connection between software and the web, although differences do appear in the degree of the comparisons that are made. Powell also asserts that such approaches are preferable to the use of the more prevalent RAD[3] [Boehm1997] in web environments. Although RAD is seen as useful because of its use of prototyping (seen as especially useful for interface design), as a method it is wasteful and, judging by the current state of many sites, does not result in well-engineered systems [Powell1998b]. He concludes that this is because many web practitioners are not conversant with more structured engineering methods.

Warren et al. speculate that webware may be only one of a family of more generally engineered artefacts. Engineered items are created using a structured method, such as the waterfall method in software engineering amongst others [Pressman2000c]. Any general method for the engineering of artefacts follows the same basic steps, whatever the artefact may be. These steps are:

1. **Analyse** - Define the problem domain and evaluate current approaches where appropriate.
2. **Design** - Provide one (or more) possible solutions to the problem along with a rationale for choosing the preferred solution.
3. **Build** - Construct the system defined by the preferred solution using the most appropriate methods and tools available.
4. **Test** - Ensure that the construction phase of the project satisfies design criteria and is fit for the purpose of its construction.

[3]A similar idea is also found in the practice of ‘Extreme Programming’

5. **Maintain** - Once the construction has reached the required standard and is deemed complete (or a stable configuration agreed), ensure that it continues to meet the required operational standards set and to repair and enhance it if necessary.

Clearly a web system (like software) is subject to all of these and is clearly an engineered system of some type when constructed in an ordered way. This is extended in software engineering by the Sprial [Boehm1988] and Waterfall models (and its variants) . Much of the work by the authors in this field are merely restatements of this seemingly facile assertion. Best practice in the area , defined by the huge number of HTML Authoring texts, largely conforms to these general principles, confirmed by the work of authors such as Powell. The early chapters of Jakob Nielsen's Designing Web Usability [Nielsen2000] concur with much of this thinking. Nielsen firmly places himself in the camp who believe in the use of engineering principles, although he also stresses the importance of creativity and inspiration. Even within this context however, the value of user-centred design is emphasised and the need for sound principles of design is established.

2.2 Early Work

As has been established, much of the earliest work in the field only goes as far as calling for a methodical approach in the development of web projects. Work in the OOHDM and RMM systems have tried to establish more regular processes to describe the production process in general hypermedia systems. Such an approach is, however, complicated by the fact that web projects may not, as discussed earlier, be like traditional software. There are special problems to be faced because web systems are heavily loaded toward end-user functionality, use elements that cannot easily (if at all) be described in a way satisfactory to traditional software engineering and are subject to lifecycles generally dissimilar from more conventional software projects.

The area of hypermedia has been extant for a great number of years. It is widely acknowledged that the defining work is that of Vannevar Bush, whose article "As We May Think", published in 1945 describes a prototypical hypermedia system. Nelson, who is believed to have first used the term 'hypertext' [Nelson1965] and Englebart, who demonstrated a working hypermedia system (known as ‘NLS’ - oNLine System) in the 1960s as part of the ‘Augment’ project [Englebart1963], also made huge contributions to the development of hypermedia. Conklin [Conkin1987] provides an introduction to the area of hypermedia a little before the first papers published by Botafogo, Rivlin and Sheiderman.

Much work however has been done on the organisation of hyperdocuments. Much of this later work sprang from two major sources; the work of Botafogo, Rivlin and Shneiderman [Botafogo1992] and work of Hatzimanikatis, Tsalidis and Christodoulakis [Hatz1995]. The thrust of both sets of work was not concentrated on Web documents because, at this time of the writing of the earlier document the web had only been a public entity for about a year. Take-up of web services, although rapid, had not reached sufficient levels to warrant special treatment.

Apart from the consideration of the web as a hypermedia system, others, such as Bray [Bray1996] and Pitkow [Pitkow1995], attempted to consider engineering problems concerned with the web in other ways. Pitkow was, for example, far more concerned about underlying network performance and how to optimise internet systems in this way. Bray’s concerns seemed to be more about mapping the extent and scope of web sites, a handy precursor to the work of those like Warren (who is referenced elsewhere in this work).

3 The Evolution and Form of Web Sites

The introduction to this document describes a linear development of web systems, in which there is some considerable overlap between generations, mainly effected by browser development. This went hand in hand with the evolution of the HTML mark-up 'language' itself [W3C2001] and related technologies such as XML. The progression is from static document-based systems to dynamic systems of much more complexity, including elements of document systems, software and other, less easily classified components.

Much current work on the engineering of web systems concentrates upon navigational issues (e.g Ricca and Tonella). This is largely in line with the often cited Botafogo and Hatzimanikatis papers that also concentrate on 'global' structures and metrics, giving little if any consideration to the node level measures that both papers do identify as an issue.

Booch [Booch2000], Hassan [Hassan2001a] and Hassan & Holt [Hassan2001b] also discuss the macroscopic structure of web systems. In Hassan's case this is mainly out of a concern to be able to reverse engineer web system architecture. The representations they provide for the modelling of such systems are shown in figures 3 and 3

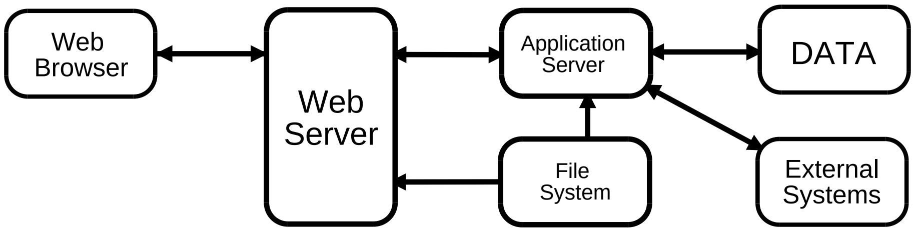

Figure 3: Booch - Diagrammatic Representation of a Web System

In both cases, the architecture is essentially three-tiered, comprising a browser (end user), Web Server (middleware system) and Application Server.

Fine detail within these systems differ but Booch alone *directly* illustrates the filesystem level as an important component. Booch's analysis goes a little further by classifying the right-hand side of his diagram (Fig. 3) as more in keeping with 'traditional' client-server systems, while the left-hand side

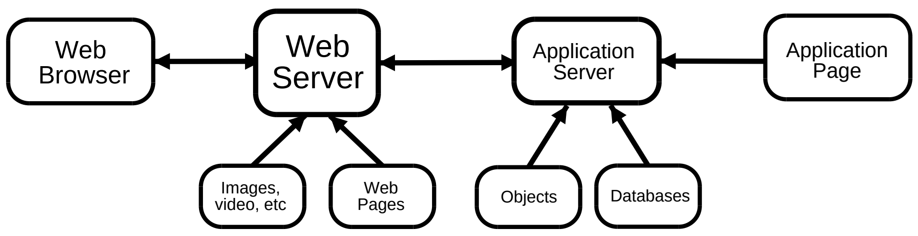

Figure 4: Hassan - Diagrammatic Representation of a Web System

details the more newer web-based extensions to the process, an evolution from more traditional software. The elements of the user interface and network are added at this point, concurring with other analyses of the properties of web projects. Hassan's system is rather more abstract, in that it uses the application server to acess obejcts or databases of some description. The mechanism for doing so is not specified.

3.1 "Web Engineering is Not Software Engineering"

Warren [Warren2001b], suggests that web sites obey Lehman's Laws of Software Evolution but also that web systems require one type of maintenance that is not extant in standard software, *speculative maintenance.*

Powell, once again, spends a great deal of time considering the web in terms of software engineering [Powell1998c]. Part of this consideration is a list of properties of a well-engineered web site:

1. **Correctness** - The site performs its specified functions and does so as if error free. Correctness is a difficult quality to specifiy entriely: sites that may *appear* to be correct may contain incorrect (invalid or badly formed) components..
2. **Testablity** - The site can have functionality and usabilirty tested as thoroughly as possible, preferably with test data and test scenarios for sites containing interactive components.
3. **Maintainability** - Making changes to the site should be as easy as possible, including the ability to make small changes or even to add or

delete entire sections easily.

4. **Scalablity** - The site is able to support increases in the number of users. The site should also be able to be ported to other servers easily, for the purpose of mirroring or clustering, for example.

5. **Reusability** - The site should be copmponent-based where possible, allowing developers to re-use similar code in other projects, or to adapt code from other sources.

6. **Robustness** - The site should be able to be used by users with the confidence that services will be available and properly functioning. This applies not only to interface issues, but also those of bandwidth, server uptime and properly functioning back-end components.

7. **Readability** - Source files and resources used to build site components should be perspicuous for developers, once again aiding maintainability. Many of the issues involved, such as commenting and formatting of source code are those that apply in 'traditional' programming.

8. **Well-Documented** - Well-documented sites aid maintainability, in that the project history is there for future maintainers to see and consult when needed. The importance of on-line help for users as sites expand should also not be underestimated.

9. **Appropriately Presented** - The function and target audience of the site should be key considerations for developers when developing interface components..

These qualities are more specific restatements of Pressman's application of sound engineering principles. The first six items certainly describe the quality of the product in engineering terms. Two of the final three items also define the quality of the systems for its developers, alluding the more inclusive concept of *habitability* (q.v.) of a system for **all** of its users. The very last items refers to user interface creation and branding within commercial projects.

Powell does however state, quite categorically, that web and software engineering are different. This is mainly as a result of the more document and content-focused nature of many sites and also because of the culture of the developer base. With a large emphasis upon user interface, the aesthetics of the system take a much more prominent role.

4 Webware, Software, Documents and Hypermedia

As has been discussed earlier, most of the earlier literature (and much of the later lierature too) identifies clear parallels between web and software systems. Even considering the work of people like Booch, Pressman et al, there is little at present to connect the two areas in any significant way. Little progress has seemingly been made since the first publications in the late 1990s.

4.1 Webware as Software

The programmes for the major software evolution conference ISPSE / IWPSE [4] have contained nothing specifically related to web evolution in the recent past, while the ICSM conference the has the WSE workshop on web evolution co-located with it. Generally, the software evolution community appears to be paying liittle attrention to the specific problem of web evolution.

The work of Boldyreff and at the Research Institute in Software Engineering (RISE) in Durham in the UK, amongst others, does try to place Web engineering within a wider software engineering context. RISE has, in the past, and is continuing today, to do a great deal of research in the area of software metrics, but most particularly in the field of software systems evolution and in the field of web maintenance, evolution and metrics (viz. WWWMaint, the WWW Maintenance project and WebSEM-Web Site Evaluation Metrics, in conjunction with CIC Scarborough) [RISE2001]. Much of this work has connections with the work of Lehman and his laws of software evolution ([Lehman1985],[Lehman1999] inter alia), which is a starting point for some of Warren's work [Warren2001b], [Warren1999]. There are also weaker connections to more general software evolution, apparent in the work of Glover [Glover2001], also working at Durham, who describes the improvement of software systems to allow easier maintenance. His work describes the problems of adapting software systems when its environment changes (environment being anything with which the software interacts). He spec-

[4] the International Symposium / Workshop on Principles of Software Evolution

ifies an agent-based system to help devise self adapting systems to react o environment changes.

To define the evolution of web systems in terms of software, only manages to describe web systems *purely* in software terms. This, given some of the emerging properties of document systems, and the way in which they (both documents and their properties) are being considered by those who create them, is not likely to provide satisfactory answers to the problem of the nature of the web.

To some degree Warren's work shares some of these concerns although his paper presented at WSE2001[Warren2001a] does present the possibility that web and software systems may be members of a higher level family of engineered artefacts. This is likely to be a rich area of exploration in the near future.

4.2 Webware as Software-Like Systems

Many software engineers, such as Pressman [Pressman2000a] (q.v.), have found themselves commenting on the production of web-based systems, There seems to be a belief in many quarters that web engineering is merely a specialised case of engineering a software system.

Pressman acknowledges that major differences exist between "traditional" software engineering projects and web engineering [Pressman2000b]. Most of these are connected with software lifecycle and the software process, where many theories about software evolution can be greatly twisted. Lehman's laws of software eevolution are applicable but the lifecycle of a web project is likely to be highly compressed and discontinuous. This can lead to differences in engineering approach in web systems.

The work of people such as Hassan and Holt as well as commentaries by Booch amongst others point to an interesting middle ground: WebApps (as Pressman calls them) are like software systems but are also heterogeneous, containing different types of components and need to be largely platform independent. Pressman also identifies some ccharacteristicsof WebApps that make them distinct from so-called "traditional " software.

- Firstly, development lifecycles for WebApps tend to be shorter [5].
- Secondly, security is a major consideration when using a web based application.
- Thirdly, the user interface and aesthetics are given a much higher ppriority largely because of their greater role. Unlike software projects (with the possible exception of hypermedia projects), web projects are largely much more content driven.

4.3 Webware, Software and the Use of Software Metrics

As has already been mentioned, web projects appear to be undergoing some type of development crisis. There are distinct parallels with software projects of the late 1960s and early 1970s. The discipline of Software Engineering was defined to help address some of these development issues, signposted by such papers as Djikstra's deprecation of the GOTO statement in languages of the time [Djikstra1968] and the work of Parnas [Parnas1972] and Wirth [Wirth1971] in calling for more structure, modularisation and re-use. These are issues that are rising up again in the creation of web systems

Traditional software metrics are broadly comprised of size metrics, complexity metrics and density of comment. Each of these can be measured in a variety of ways. Object-oriented metrics are primarily applied to classes, message passing, coupling, cohesion and inheritance. Many class based metrics measure the complexity of a class by considering its constituent methods. Traditional metrics can be applied in these cases. Little of importance appears to have been done to produce a mapping basis for metrics between object-oriented systems and node-based hypermedia systems, although object-oriented development methods do exist within hypermedia.

Webware can justifiably be described as a subset of a more generic class of hypermedia applications, being distributed across a network,scalable but closed in the sense that the embedded link system is generally not extensible (at least not in HTML: XML and XLink change this) . We must, therefore,

[5]Discontinuity notwithstanding

also consider whether any current developments in the field of hypermedia are applicable to webware. As Hall and Lowe comment, however, metrics in the area of hypermedia are not common[Lowe1999a], Mendes, Mosley and Counsell's paper, published in IEEE Multimedia in Summer 2001 (q.v) attempts to address this situation by attempting to use some existing metrics to provide a starting point for the production of meaningful metrics. This is partially successful but does not attempt to consider the role of programming in the engineering process.

The use of the term 'programming language' when applied to HTML/XML etc. seems not to be well- defined in literature surrounding the subject. Hall and Lowe do appear to link mark-ups and programming languages together when they discuss the 'Publishing Model' for hypermedia development [Lowe1999b]. The possibility of being able to apply some software metrics to mark-up documents appears to be a real one if this relationship can be established more firmly.

Metrics applicable to the web can be placed into three broad categories, each containing some overlap with the others:

- **Development Metrics** are those concerned with the creation of HTML and other mark-ups, the topology of the hypermedia navigation space and the creation of other components, whether they are programmatic like Java applets or servlets, script based like Javascript or PHP or of other types, such as SVG, Flash, RealMedia etc. These are the already discussed node and global metrics.

- **Deployment Metrics** are concerned with how the engineered items are placed into the server environment and how they interact with other services, such as databases and mail transport, for instance. This is, in part, addressed in the work of those like Pitkow, Bray (both mentioned earlier.) and similar. This is a major part of engineering a web project, much more so than many in 'traditional' software enginering, given the non-deterministic nature of internet performance. Some may class these as 'operational issues' and not of relevance to the engineering of the initial system. These may also be classified as global metrics.

- **Delivery Metrics** are those measures which are applicable to the final delivery of the artefact mainly characterised by usability and end-user prperformanceetrics. The principal work in this field has been done by Nielsen and associates amongst others and will not be dealt with further.

It is a misapprehension to believe that any of these fields are totally distinct from one another. Measures taken to address issues in one area may have significant impacts in either or both of the others. This may go some way to refuting the argument that deployment metrics are not relevant.

Much current work in hypermedia engineering is focused on the application of the major design methods, Schwabe and Rossi's OOHDM [Schwabe1995] and Isakowitz, Stohr and Balasubramanian's RMM [Isakowitz1995]. Broadly speaking these two methods are similar in nature although it can be argued that RMM is slightly more user-centric due to its explicit definition of construction and run-time testing stages. As mentioned earlier, the object-oriented nature of these systems suggests that they would be a fertile ground for the application of some types of object-oriented software metrics. Puzzlingly, this seems not to have happened to any great degree. Gaedke and Gellersen have spent a great deal of time applying Object-Oriented methods to first hypermedia in general, then more recently, to web systems

4.4 Webware and Document Engineering

The use of the term "document engineering" as applied to the World Wide Web is also a relatively new one, stemming from the inital work of Bebo White, discussed earlier, and explored in much of the work defining web engineering as multi-disciplinary that has already been dicussed. The need to create ordered and well-made documents seems to have been a key issue when the first mark-ups were defined in the mid-1970s. An important part of the development of mark-up languages was a recognition that many different types of data may need to be represented (in some cases) or referenced, where representation was not possible [6]. Indeed, Rick Jelliffe [Jelliffe1998] identifies this feature as a highly important one in any mark-up language. Earlier browsers used the helper application to access the data referenced by HTML documents in the usual way.

[6]The SGML standard makes provision for utilising data that the author does not wish to be, or cannot be, marked up

The rise in the use of XML and also in the use of technologies such as XSLT has resulted in improved engineering of systems because the construction of such items has required them to be done in a more regular, modular way. The recognition of the need for engineering principles to be applied to web documents was noted in a session at WWW6 in 1995 by Bebo White of SLAC (Stanford Linear Accelerator) [7] In this session at the conference he defines a 'Web Document' as a collection of linked 'Web Pages'. He also goes on to define the term 'Web Document Engineering' a precursor to the more generic term 'Web Engineering' coined around two to three years later.

Although little published work seems to exist in this specific area at present there is a definite increase in interest, resulting in the creation of a new forum, the ACM Symposium on Document Engineering [ACMSDE2001], to be held in Atlanta, USA in November 2001. Unfortunately, the proceedings of this conference were not available at the time of writing although they are expected in the very near future. The call for papers to this conference acknowledges that, 'documents' are no longer static, physical entities'. It continues, to provide this instructive definition of a document from a more contemporary viewpoint:

> "A document is a representation of information that is designed to be read or played back by a person. It may be presented on paper, on a screen, or played through a speaker and its underlying representation may be in any form and include data from any medium. A document may be stored in final presentation form or it may be generated on-the-fly, undergoing substantial transformations in the process. A document may include extensive hyperlinks and be part of a large web of information. Furthermore, apparently independent documents may be composed, so that a web of information may itself be considered a document."

This may seem to illustrate the major difference between software and webware in a traditional sense; documents are designed to be read by humans, software is to be read by machines. Software source, however, *is* meant to be

[7]SLAC ran the first non-European web server, going on-line on December 12 1991, according to http://www.w3c.org/History.html

read by humans, those developing and maintaining the code. In this sense there can be no doubting that, in the source state, software projects are also document systems. This, once again, reinforces the idea that web and software systems may be part of a more general family of artefacts.

4.5 Webware and Hypermedia Systems

Lowe and Hall, during the course of a survey of web engineering practice, found little, if any, evidence of well-defined metrics extant in the more general area of hypermedia. Given the amount of work done in the hypermedia field, this is somewhat surprising. It is, however an area of burgeoning interest in the software engineering sphere as even more traditional texts such as Pressman's [Pressman2000] now make reference to engineering quality in webware, even though it may only be at a superficial level.

Little attention seems to be being paid to the uncomfortable (for software engineers) issue of how non-software components are engineered and their roles within wider systems. For a web project this is a major issue, both in terms of product and process. This is an issue that the Web-SEM project was created to address [Boldyreff2001].

Even some of the most recent work (such as that of Ricca and Tonella) seems to concentrate on the macroscopic structures within web hypermedia. Little seems to be being written about the previously mentioned relationship between mark-up languages and what Wirth describes as, “formal notations” [Wirth1977] rather than programming languages per se. It appears that little progress has been made from previously mentioned papers published in 1992 by Botafogo et al. and in 1995 by Hatzimanikatis et al. where idealised hypermedia systems are modelled, mainly to analyse navigation and movement through them.

This emphasis upon large scale structure influences work on usability issues [Nielsen2001],[Krug2000]. Few of them sufficiently describe the problem of the creation and maintenance of web systems from the point of view of the developer. Nakayama [Nakayama2000] has approached this problem, describing not only macroscopic design issues but also those of page layout and conceptual relevance. The technique presented still concentrates on improving visual layout for the end user's benefit, placing less emphasis upon the developer's needs.

Work in large scale hypermedia appears to be like macroeconomic theory: it may help to explain large scale phenomena like inflation or interest rates but does not, for instance, seem to explain the operation of smaller systems such as the markets for certain products or localised behaviours. In a similar sense, explanations of global navigation do not describe behaviour inside nodes within that system. As Deshpande and Hansen discuss (q.v), this may be an issue that is uppermost in the engineering of web systems.

5 Is Mark-up a Programming Paradigm?

The architect Christopher Alexander describes the concept of "inhabitability" in an architectural system. He asserts that those who finally use the structure are not, in fact, its *only* users. Even those who build and maintain such systems also need to use it in some way. Good architecture also takes account of the needs of these users. Thus, the engineering quality of a web system affects not only its end users but also those who 'inhabit' the code while writing, testing and maintaining it. In building such systems the needs of the developer must also be given appropriate weight.

This is a subject spoken about by the programmer Richard Gabriel in his book 'Patterns of Software', [Gabriel1996a]. The importance of building code that is easy for developers to 'inhabit' is an established one for normal software systems. Software metrics attempt to quantify the factors that developers feel to be important indicators of code's complexity compared to its ease of use.

Although not widely accepted and controversial, there are a very few sources who classify HTML (and markups generally) as VHLLs (Very High Level Languages) in much the same way that a language such as SQL might be [Frostburg2000]. Even Lowe and Hall, as discussed previously, make at least some connection. Open as this point is to argument, it has not prevented some from attempting to provide more conventional measurements for HTML. Capers Johnes [Johnes2001] provides figures for function points within HTML, classifying it as a Very High Level Language in a similar grouping to languages like Perl and Python.

Such languages are more naturalistic for programmers (and non-programmers) and are generally more declarative in nature. The number of volumes that introduce users to "programming in HTML" are too numerous and written by those with sufficient knowledge of the subject for them all to have confused terminology. This suggests that many do perceive HTML as a programming language of sorts but the lack of literature that states it explicitly is a highly puzzling state of affairs.

More contentiously, they may also be more suited to visual programming methods. They are also generally highly limited in scope and are not useful for general purpose activity, as are many lower level languages. Previous internal discussion papers [Stephens2001a], [Stephens2001b] have proposed similar ideas to explain why some types of software metric may be appropriate for measuring quality in web hypertext systems.

6 Conclusion

Most of the work so far in the field of Web Engineering has focused on only a few specific areas. This is perhaps not so surprising, given the highly disparate nature of web systems and the inherent difficulties in trying to tie them together cogently. In the web's infancy, the major research concerns centred around the topology of the web and delivery of services. The engineering quality of the sites themselves were not considered formally. Early webmasters developed support networks, used ad hoc engineering methods and passed them on as best practice. Thoughts about the engineering quality of web systems only started to become a pressing issue as the size of projects started to balloon.

The time when such ad hoc methods could be relied upon as a method of ensuring quality has now passed: the size of web projects necessitates group working and production of such systems on a large scale.

Now that the engineered quality of web systems is being more actively considered the work divides into several relatively distinct areas, each flawed in some small way as part of a larger picture.

One of these areas is 'Idealised Hypermedia', as seen in the work of Hatzimanikatis et al, Shneiderman et al and latterly with Ricca & Tonella. Each of them concentrates mainly on hypermedia systems at the global level, only focussing on those metrics. Node-based measures are hardly considered at all.

Others, like Boldyreff, for example, seem to consider web systems simply as adjuncts to software systems. This attitude persists because, seemingly, only the software components of such systems are being considered and not the mark-ups contained at the node level. The role and nature of markup itself seems to have been largely forgotten or relegated to a level of little importance in the wider scheme of web systems development, possibly beacause the production mark-up it is not considered as a type of software development.

Document engineers, if the evidence of recent activity is to be believed, are beginning to re-evaluate the function and role of documents. There is a growing appreciation that documents are now no longer static entities. There is seemingly no acknowledgement, however, that the construction of a document system has some visible similarities to the construction of software systems.

There appears to be little or no work of a more ‘holistic’ nature with regard to web engineering. Such a hybrid approach, mixing programming paradigms with the creative process of content creation, would seem to be a necessity given the web’s rapid evolution and its need to be a truly heterogeneous cross-platform environment. Once the relationship between software and document system is more clear, the application of appropriate metrics can be considered afresh.

References

[ACMSDE2001]	Munson EV, *ACM Symposium on Document Engineering*, Available on-line at http://www.documentengineering.com/ Accessed: October 12 2001
[Berghout1999]	Berghout E, van Solingen R, *The Goal / Question / Metric Method*, McGraw-Hill, 1999
[Berners-Lee2000]	Berners-Lee T, *Weaving The Web*, TEXERE, 2000, pp38-46
[Bieber1998]	Bieber M, *Hypertext and Web Engineering*, In Proc. 9th ACM Conference on Hypertext and Hypermedia: Links,Objects, Time and Space-Structure in Hypermedia Systems, 1998, pp277-278
[Boehm1988]	Boehm BW, *A Spiral Model of Software Development and Enhancement*, Computer 21(5), May 1988, pp61-72
[Boehm1997]	Boehm BW, Devnani-Chulani S, Egyhed A eds, *Knowledge Summary: Focused Workshop on Rapid Application Development*, USC, Los Angeles CA, USA, 23-27 June 1997.
[Boldyreff2001]	, Boldyreff C, Warren P, Gaskell C, Marshall A, *Establishing a Measurement Programme for The World Wide Web*, In Proc. 2001 Symposium on Applications and the Internet-Workshops, (SAINT 2001 Workshops), IEEE, 2001
[Booch2000]	Booch G, *The Architecture of Web Applications*, Available on-line at IBM Partnerworld for Developers, http://www.developer.ibm.com/library/articles/-booch_web.html, Accessed: October 12 2001
[Botafogo1992]	Botafogo RA, Rivlin E, Shneiderman B, *Structural Analysis of Hypertexts: Identfying Hierarchies and Useful Metrics*, ACM Transactions on Information Systems 10(2), April 1992, pp142-180

[Bray1996] Bray T, *Measuring the Web* In Proc. 5th WWW Conference, Paris, France, 1996

[Bush1945] Bush V, *As We May Think*, The Atlantic Monthly, July 1945

[Conkin1987] Conklin, J *Hypertext: An Introduction and Survey*, IEEE Computer, 20(9), September, 1987, pp17-41

[Dalton1996] Dalton S, *A Workbench to Support Development and Maintenance of World-Wide Web Documents*, Available Online at :http://www.dur.ac.uk/ dcs0cb/workbench/, Accessed November 10 2001, Department of Computer Science, University of Durham, UK, 1996

[Deshpande2001] Deshpande Y, Hansen S, *Web Engineering: Creating a Discipline among Disciplines*, IEEE Multimedia 8(2) April-June 2001, pp82-87

[Djikstra1968] Dijkstra EW, *Go To Statement Considered Harmful*, CACM, 11(3), March 1968, pp147-148

[Englebart1963] Englebart DC, *A Conceptual Framework for the Augmentation of Man's Intellect*, In Vistas in Information Handling,Spartan Books, pp1-29, 1963.

[Englebart1995] Englebart, DC, *Boosting Our Collective IQ - Selected Readings*, Bootstrap Institute/BLT Press, 1995

[Frostburg2000] *Survey of Programming Languages*,
Available on-line at:
http://www.frostburg.edu/dept/cosc/htracy/cosc120/
MODULES120/NetPL/PL_Net.htm,
Accessed November 2, 2001,
Last alteration date: December 13 2000, Department of Computer Science, Frostburg State University MD USA, 2000

[Gabriel1996a] Gabriel RP, *Patterns of Software*, Oxford University Press, 1996

[Gaedke1999] Gaedke M, Gellersen HW, Schmidt A, Stegem´uller U, Kurr W) *Object-oriented Web Engineering for Large-scale Web Service Management*, Thirty-Second Annual Hawaii International Conference On System Sciences (HICSS-32) Maui, USA, January 5 - 8, 1999.

[Gellersen1998] Gellersen HW, Gaedke M, *An Object-Oriented Model (not only) for Hypertext in the Web*, HTF5: The Fifth International Workshop on Engineering Hypertext Functionality into Future Information Systems, 20th International Conference on Software Engineering, Kyoto, Japan, April 1998.

[Glover2001] Glover JJ, *Inherently Flexible Software*, PhD Thesis, University of Durham UK, 2001

[Google2001] , Google.com, Available online at: http://www/google.com, Accessed November 16 2001

[Hassan2001a] Hassan AE, Architecture Recovery of Web Applications, MSc Thesis, University of Waterloo, Ontario, Canada, 2001

[Hassan2001b] Hassan AE & Holt RC, Towards a Better Understanding of Web Applications, In Proc. Third International Workshop on Web Site Evolution WSE 2001, Florence, Italy, November 2001,

[Hatz1995] Hatzimanikatis AE, Christodoulakis D, Tsalidis CT, *Measuring the Readability and Maintainability of Hyperdocuments*, Software Maintenance: Reasearch and Practice. Vol 7, pp77-90

[Isakowitz1995] Isakowitz T, Stohr EA, Balasubramanian P *RMM: A Methodology for Structured Hypermedia Design*, CACM 38(8) August 1995, pp34-44

[Jelliffe1998] Jelliffe R, *The XML and SGML Cookbook: Recipes For Structured Information*, Prentice Hall 1998

[Johnes2001]	Johnes C, *Sizing Up software*, Scientific American, xx(x), Dec 1998. Available online at: http://www.sciam.com/1998/1298issue/1298jones.html Accessed October 15 2001
[Krug2000]	Krug S, *Don't Make Me Think*, Que, 2000
[Lehman1985]	Lehman MM, *Program Evolution*, Academic Press, 1985
[Lehman1999]	Lehman MM, *Software System Maintenance and Evolution in an Era of Reuse, COTS and Component Based Systems*, IEEE International Conference on Software Maintenance, Oxford, England, August 30 - September 3, 1999,
[Lowe1999a]	Lowe D, Hall W, *Hypermedia And The Web: An Engineering Approach*, Wiley, 1999, p200
[Lowe1999b]	*ibid*, pp509,510
[Mendes2001]	Mendes E, Mosley N, Counsell S, *Web Metrics-Estimating Design and Authoring Effort*, IEEE Multimedia 8(1) Jan-Mar 2001, pp50-57
[Murugesan1999]	Murugesan S et al., *Web Engineering: A New Discipline for Development of Web-based Systems*, Online Proc. 1st International Conference on Software Engineering Workshop on Web Engineering, http://fistserv.macarthur.uws.edu.au/san/icse99-WebE/ICSE99-WebE-Proc/San.doc
[Nakayama2000]	Nakayama T, Kato H, Yamane Y, *Discovering the Gap Between Web Site Designers' Expectations and Users Behavior*, In Proc. 9th International World Wide Web Conference (WWW9), Amsterdam May 15-19 2000, Available online at: http://www9.org/w9cdrom/start.html
[Nelson1965]	Nelson T, *A File Structure for the Complex, the Changing, and the Indeterminate*, 20th National Conference, ACM, 1965

[Nielsen2000] Nielsen J, *Designing Web Usability: The Practice of Simplicity*, New Riders Publishing, 2000, pp10-15

[Nielsen2001] Nielsen J, Available online: http://www.useit.com/ Accessed October 1 2001

[Oftel2001] The Office of Telecommunications Oftel), *Consumers? use of Internet Oftel residential survey Q6 August 2001*, Available Online at: http://www.oftel.gov.uk/publications/research/2001/ q6intr1101.htm, Published november 4 2001, Accessed November 18 2001

[Parnas1972] Parnas, DL, *On the Criteria To Be Used in Decomposing Systems into Modules*, CACM 15(12), December 1972 pp1053 - 1058

[Pitkow1995] Pitkow JE, *Summary of WWW Characterizations*, In Proc. 7th WWW Conference, 1998

[Powell1998a] Powell TA, et al, *Web Site Engineering: Beyond Web Page Design*, Prentice Hall, 1998

[Powell1998b] *ibid*, pp16,17

[Powell1998c] *ibid*, pp25-49

[Pressman2000] Pressman, RS, *Software Engineering: A Practitioner's Approach (European Adaptation)*, (Fifth Edition), McGraw-Hill, 2000

[Pressman2000a] *ibid*, pp813-842

[Pressman2000b] Pressman, RS, *What a Tangled Web We Weave*, IEEE Software, 17(1), Jan 2000, pp18-21

[Pressman2000c] Pressman, RS, *Software Engineering: A Practitioner's Approach (European Adaptation)*, (Fifth Edition), McGraw-Hill, 2000, pp813-842

[Ricca2000a] Ricca F, Tonella P, *Web Site Analysis: Structure and Evolution*, In Proc. 2nd International Workshop on Web Site Evolution, WSE 2000, Zürich, Switzerland, 2000

[Ricca2000b] Ricca F, Tonella P, *Visualisation of Web Site History*, In Proc. 2nd International Workshop on Web Site Evolution, WSE 2000, Zürich, Switzerland, 2000

[RISE2001] University of Durham, Research Institute in Software Evolution, Available online at: http://www.dur.ac.uk/CSM/, Accessed November 23 2001

[Schwabe1995] Schwabe D, Rossi G, *The Object Oriented Hypermedia Design Model*, CACM 38(8) August 1995, pp45-46

[Somerville1989] Somerville I, *Software Engineering - Third Edition*, Addison-Wesley 1989

[Stephens2001a] Stephens D, *What Is Software?*, Unpublished Paper University of Hull, 2001

[Stephens2001b] Stephens D, *What Is A Document?*, Unpublished Paper, University of Hull, 2001

[Stross1996] Stross C, *The Web Architect's Handbook*, Addison-Wesley, Harlow UK, 1996, p180

[W3C2001] World Wide Web Consortium, Online at http://www.w3c.org, Accessed October 30 2001

[Warren1999] Warren P, Boldyreff C, Munro M, *The Evolution of Websites*, in Proc. International Workshop on Program Comprehension IWPC99, Pittsburgh PA, USA, IEEE Computer Press 1999 Submitted November 2001

[Warren2001a] Warren P, Gaskell C, Boldyreff C *Preparing the Ground for Website Metrics Research*,
In Proc. Third International Workshop on Web Site Evolution WSE 2001,
Florence, Italy, November, 2001

[Warren2001b] Warren P, *The Evolution of Websites*, PhD Thesis, University of Durham, Submitted November 2001

[WebE2001] Murugesan S ed., *WebE Home Page*, Available Online at: http://fistserv.macarthur.uws.edu.au/san/WebEhome/ Accessed November 16 2001

[White1996] White B, *Web Document Engineering*, Available Online at: http://www5conf.inria.fr/fich_html/slides/tutorials/T14/all.htm, Accessed November 10 2001

[Wirth1971] Wirth, N, *Program Development by Stepwise Refinement*, CACM, 14(4), April 1971, pp221-227

[Wirth1977] *What Can We Do About the Unnecessary Diversity of Notation for Syntactic Definitions*, CACM 20(11) Nov 1977, pp822-823.

[Zelnick1998] Zelnick N, *Nifty Technology and Nonconformance. The Web in Crisis*, Computer, 31(10) October 1998, pp115-116,119