Towards a Contingency Based Approach to Web Engineering

Götz Botterweck and Paul Swatman

Department of Informatics, University of Koblenz-Landau, Germany
botterwe@uni-koblenz.de

Stuttgart Institute of Management and Technology, Germany &
School of Information Systems, Deakin University, Melbourne
swatman@uni-simt.de

Abstract

In this paper, we introduce our preliminary work in developing an analytic framework which allows us to contrast the problem of developing 'web-like applications' (WLAs) against our accumulated understanding of software systems development. The analytic framework will form a basis for the development of a contingent approach to selection of methods, tools and techniques, then integrating these within a suitable methodological process, for WLA development. This approach allows us to make use of our existing understanding of software engineering methodology, while alleviating the danger of relying on the consequences of assumptions within that literature which do not hold, or which hold imperfectly, in the domain of WLA development. We illustrate how our approach could provide structure for the analysis of the characteristics of the potential user base of a WLA vis-à-vis the user base for a conventional software system.

Keywords:

Web Engineering, Contingency Based Approach, Software Engineering, Software Development Methodologies, Web Applications, Distributed Applications, WAP, VoiceXML

1. Introduction

In this paper, we discuss methodological issues surrounding the building of "web-like applications". We consider, in particular, the extent to which the conventional wisdom in the fields of Software Engineering and IS development methodology is applicable to – and to what extent it is misleading in – systems development for the World Wide Web (WWW) and conceptually related technological infrastructures.

In comparison to traditional software, both web sites and – indeed to a greater extent – Web and web applications are immature. Until relatively recently, web sites and web applications were commonly developed in an ad hoc manner. We frequently saw that web sites were 'under construction' and contained dead links leading to HTTP Status 404 pages. In the excitement and enthusiasm of the newly seen possibilities, web creators often focused on speed and novelty at the expense of quality and structure. This can now be seen to have largely changed – necessarily so – users have more alternatives and low switching costs lead to competitive pressure on the providers of web sites and web application.

There is a growing awareness of quality amongst both users and creators of web applications and – in terms of what is actually used or visited – we see the survival of only the fittest.

Consequently, quality now should play an important role during development of web applications.

Simultaneously web creators are gaining experience and discovering solution patterns which have proven themselves in practice. Accordingly, one can anticipate that, similarly to what we have seen in respect of software engineering generally, we shall see a maturing engineering sub-discipline for web-like applications.

The structure of our paper, preliminary work towards the development of a model of contingent selection of methodology for the development of web-like applications, is as follows:

- First, we offer motivation for our study and offer some formal definitions relating to the World Wide Web, similar platforms and the applications supported by such platforms.
- Secondly, we develop a conceptual model of systems development.
- Thirdly, we contrast web-like systems engineering against software engineering leading to a preliminary analytic framework to support contingent methodology selection.
- Finally, we illustrate the use of this analytic framework to structure a discussion of methodological issues associated with the context or environment in which web-like applications are developed, focusing particularly on the potential user base.

2. Web Applications

Over the last decade, a considerable literature has developed in respect of methods for hypermedia development or web-based systems development (ex. Fernández et al. 1998; Isakowitz et al. 1995; Schwabe et al. 1996). The literature is generally descriptive in nature – introducing and illustrating the use of methods for and approaches to the development of systems diversely categorized as, for example 'web applications' or 'hypermedia systems'. It is natural, in such work, that the class of system targeted by the method remains rather loosely defined. The consequence of this is, however, both redundancy and conflict within the terms used to describe classes of target system across the work of the various authors in the field.

In our work, nevertheless, we seek to develop a foundation for an approach to the selection of methods, tools and techniques of systems development which is contingent upon the characteristics of, and the context within which, the system is to be built. Consequently, it is essential that we are able to define precisely the characteristics which will guide us in our methodological choice.

We aim to extend the scope of our study to include systems founded on platforms which are, in some sense, similar to the World Wide Web. We, therefore, extend our terminological work with definitions of the World Wide Web itself and with dimensions of similarity between underlying technological platforms which we consider to be, for our purposes, similar.

We begin by considering systems which are based on the WWW. We may attempt to distinguish between:

- A *web site* which publishes content focusing and primarily on information browsing. The web site itself consists of documents (web pages) and links between them. This structure directly corresponds to the hypermedia model (Halasz and Schwartz 1994).
- A *web application* based on World Wide Web technologies (W3C 2002c) which allows transactions (e.g. database update or sending an email) to be executed.

The category "web application" may, however, be better conceived as a superset which includes the special form "web site" but which additionally allows the possibility of additional functional logic. In the following discussion, we deal with the general class of web applications.

Some authors distinguish between 'hypertext' and 'hypermedia' depending on the type of media, while others use the terms interchangeably. In this paper, we will adopt the latter view. We can define a web site as an example of hypermedia – one which provides access to information organized as a digraph (Bieber 2000; Conklin 1987; Halasz and Schwartz 1994). Then we can say: hypermedia is an abstract or conceptual model, which may be implemented in the form of a web site.

Hypermedia systems, in their general form, possess additional important properties including the ability to manipulate/annotate nodes and bidirectional links. Early design ideas for the World Wide Web incorporated such functionality (Berners-Lee 1990) – some of these ideas were demonstrated in technological prototypes (W3C 2002a; W3C 2002b) and some can be simulated by additional tools (Google 2002). Nonetheless, these approaches never really made it into the Web that we know today. Since these concepts were not implemented in practice, some authors have considered the web not to be a full hypermedia platform.

We can now examine a web application with functionality beyond that of the web site (see the illustration in Figure 1).

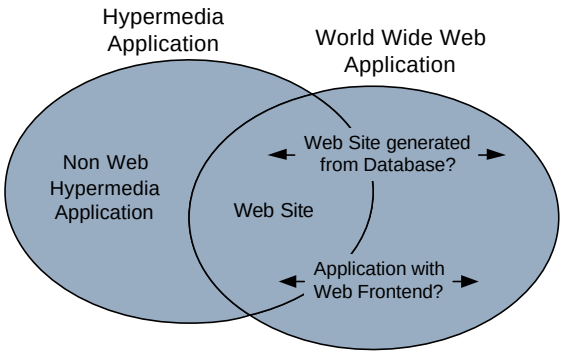

Figure 1 - Hypermedia vs. World Wide Web

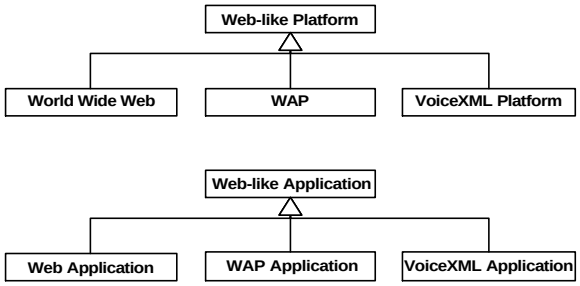

Figure 2 – Web-like Platforms and Web-like Applications

If we have a dynamic web site (where, for example, content is generated from a database), we can still consider this as an instance of hypermedia – provided the essential functionality is restricted to displaying information and allowing the user to traverse links to other pieces of information. A database and other foundations for dynamic web sites, such as scripting technologies, are conceptually null – merely implementation detail.

However, in the case of an application with a web interface (e.g. a web accessible messaging server) we move, conceptually, beyond simple hypermedia – we define such an application as containing a hypermedia component.

We can identify a number of related concepts which, for completeness, we mention here:

- The term *web presence* is used almost synonymously to web site. However, it emphasizes the marketing, corporate identity aspect.
- Some authors use terms such as *web-based information systems*. For example, (Rossi et al. 1999) describe web information systems as "information systems that are constructed using Web technology". (Gnaho 2001) defines "A WIS [Web-based Information System]

is an Information System providing facilities to access complex data and interactive services through the Web."

- In some cases, *web service* is used to describe a web site, which offers some kind of service. Recently, however, this term has come to be used predominantly for a special form of application – those which expose their functionality to other application by using web protocols and languages (W3C 2002d).

2.1 Web-like Platforms (WLP) and Web-like Applications (WLA)

Having grounded our initial discussion in terms of the WWW and hypermedia systems, we now move to extend our scope explicitly to include conceptually "similar" platforms, which we will call Web-like Platforms (WLP). Applications based on these platforms will be named Web-like Applications (WLA). By extending the scope of our work in this way:

- We can make statements with more general applicability. In general, we expect most of our future findings in respect of the applicability of methods, techniques and tools for WLAs to be valid across a range of – possible all – WLPs.
- Alternatively – when we cannot make a general statement across all WLPs – a comparison of the differences can yield further conceptual refinements to our theoretic model which we introduce in Section 3 below. As an example, we may consider the transition of an arbitrary application from a WLP supporting fully-fledged web front ends to one supporting small mobile devices in multi-channel applications. The impact of user interface capability on the entire usage structure of an application may be expected to be significant.

We see the key discriminating characteristics of Web-like platforms to be:

- Adherence to the *client server model* (typically with the aim of fostering a separation of concerns). A client implementation of the user interface of the application accesses a server, which provides information and/or executes transactions parameterized upon this information.
- The communication between client and server is mediated through a *request response protocol* (e.g. HTTP (Fielding et al. 1999)) designed to be transported over wide area networks and deal with the associated characteristics of such networks (e.g. latency).
- Content is logically organized in the form of a *hypertext model*, i.e. structured in digraphical (node/hyperlink) form.
- Documents/Data offered as content or exchanged between separate system components are described in *standardized data description languages*, typically markup languages (ex. HTML (Ragget et al. 1999)).
- *Resources* (nodes) are *identified by a location in an information space* (ex. URI (Berners-Lee et al. 1998)).

Figure 2 gives examples of platforms having these characteristics: the World Wide Web itself, the Wireless Application Protocol – which is definitely not only a protocol, but rather a software platform for mobile devices – (WAP Forum 2002), and VoiceXML (VoiceXML Forum 2002).

2.2 Towards Distributed Applications in General

Although, for the remainder of the paper, we focus on WLPs and WLAs, we take this opportunity to extend our definitional structure and further generalize the scope of our future work.

We can take one more step towards generality if we move the focus from the implementation details of the platform towards the distributed applications which are implemented using this platform – seeing the web (as it is now) or web-like platforms (as defined above) as just *one* way of implementing them. The implications of this generalization are:

- By considering a broader range of application we can either make more general statements (if the features are common across different platforms) or work out the differences. This newly gained knowledge about platforms and their characteristics can help us to further develop our contingency based approach.
- We can include more approaches. An obvious step is the extension towards non-web-like platforms for distributed applications. For example we might consider non-web-like mobile devices as front-ends. In addition, we could try to also address the specialties of multi-channel applications, i.e. applications with several front-ends, some web-like some not.
- The most significant difficulty is that we must consider how the Web will evolve in the future. New application platforms are evolving which includes principles and technologies of both traditional software development approaches (desktop applications) and internet/web technologies (see Figure 3). Examples can be seen in recent developments surrounding Java and Microsoft's .NET initiative. Irrespective of whether we can expect a coalescence of platforms (Web and traditional desktop platforms) or a spectrum of options, to consider the future development of the Web, we must include these approaches and technologies.

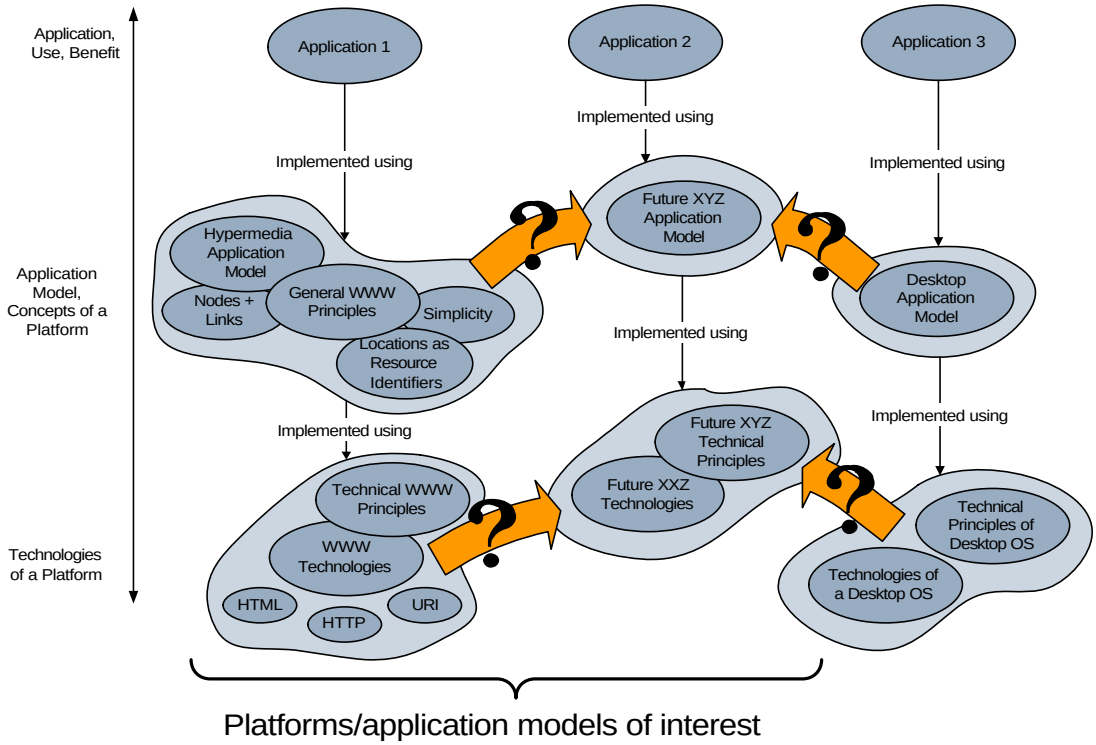

Figure 3 – The future development of application platforms

3. Sketching a Conceptual Model of Software Development

Many authors in the methodological literature have suggested a contingent approach to the selection of systems development method. Our work follows in this tradition and aims to extend its application to include web-like systems. In order to do so effectively, it is clearly necessary to identify relevant contingencies – and ultimately to build an orthogonal and parsimonious but nonetheless complete model to support methodological selection.

We begin from first principles. In Figure 4, we present, in the form of a simple semantic net, an illustrative extract of our model of the characteristics of software systems and the contexts in which they are developed – a model which was initially based in the general software engineering literature, which we acknowledge to be incomplete (indeed we would argue that it is necessarily so) and which will remain under continuous development during the life of our research programme.

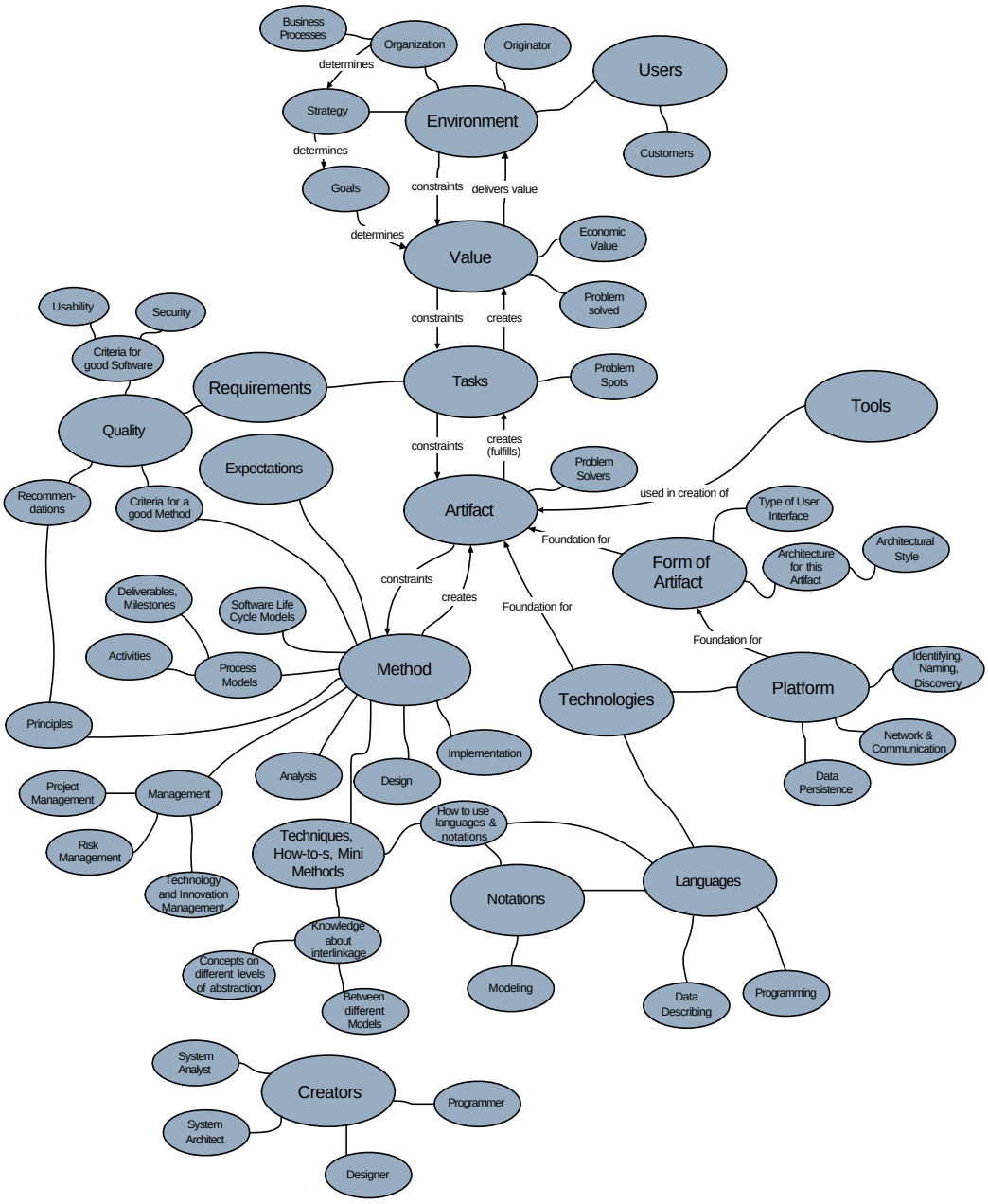

Figure 4 - Aspects of Software Development (illustrative)

Although the model in Figure 4 is merely an illustrative extract, it is clear that it is already very complex – and, of course, by virtue of the semantic net formalism it is both flexible – allowing us to argue and reason freely about the issues modelled – and also unstructured – thus forming a rather unsatisfactory foundation for future conceptual development. There are of course many possible ways to add structure and simplify such a model. Our creative conceptual work has suggested the approach to simplification and structuring through a clustering strategy which is illustrated in Figure 5. Descriptions of the clusters, and some examples of their practical application, follow in Table 1.

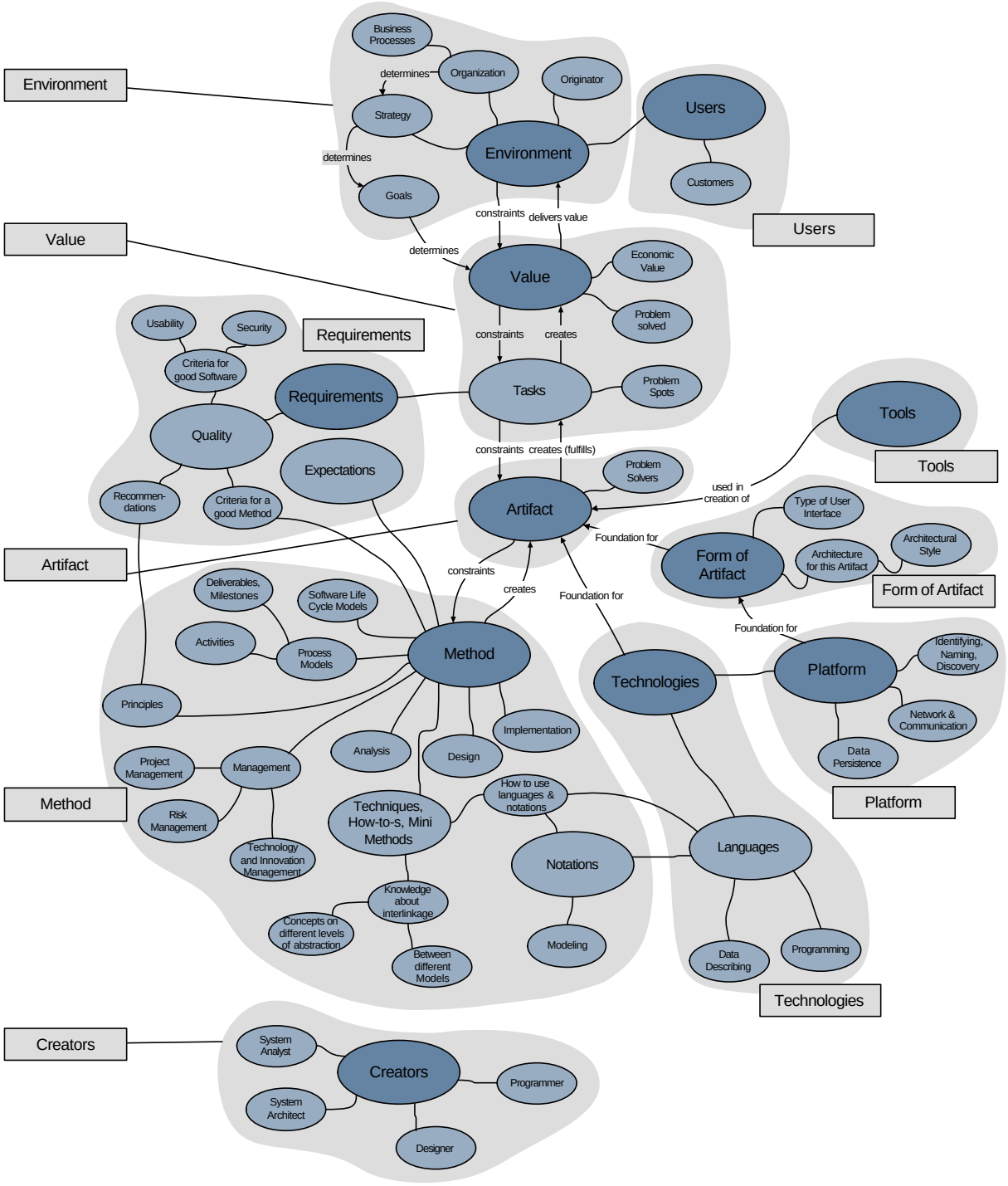

Figure 5 - Aspects of Software Development (clustered)

Name	Description	Example
Environment	Circumstances under which the artifact is used	A consulting company and its context (business processes, social norms, market in which it operates …)
Users	The persons directly using the artifact	The staff of the consulting company
Value	Acknowledged benefit provided by the artifact	The advantage to an individual or organization of an ability to create and publish reports
Requirements	Desired, expected and required features of the artifact	Expectations about the performance and the functionality of text processing software
Artifact	Software (or software intensive system) being created for a practical purpose	Microsoft Word XP
Method	Systematic procedure used by the creators for designing and building the artifact	The OPEN method (Henderson-Sellers 1997)
Technologies	Manners of accomplishing a task especially using technical processes, methods, or knowledge	Enterprise Java Beans (EJB) (Sun Microsystems 2002)
Platform	Set of principles and technologies providing a base for creating and using software artifacts	Microsoft Windows XP including the underlying technologies like COM (Microsoft Corporation 2002a)
Form of Artifact	A form of implementation of software (or software intensive systems)	'Desktop GUI application' or 'Web Application'
Tools	Tools (especially software programs) used during creation of the artifact	Microsoft Visual Studio
Creator	Person/organization designing and producing the artifact, applying the method	The designers and developers of Word XP

Table 1 – Aspects for Software Development

We may now take these 11 clusters and look to simplify still further. We see a strong conceptual association between 'Users' and 'Environment'; between 'Requirements' and 'Value'; 'Form of Artifact' and 'Platform' and between 'Tools' and 'Method' consequently we can consolidate our analysis by focusing on five clusters, nominally:

- Environment (Users)
- Value (Requirements)
- Artifact
- Methods (Tools) + Technologies + Platforms (Form of Artifact)
- Creators

Referring back to our original semantic net model, we can add some additional semantic structure – seeing that, four of these clusters (excluding 'Creators') can be understood as forming a layered model where each step represents, in some sense, a step on the 'Abstract' to 'Concrete' continuum. Figure 6 shows a graphical representation of all five clusters and, in particular of the four level layering.

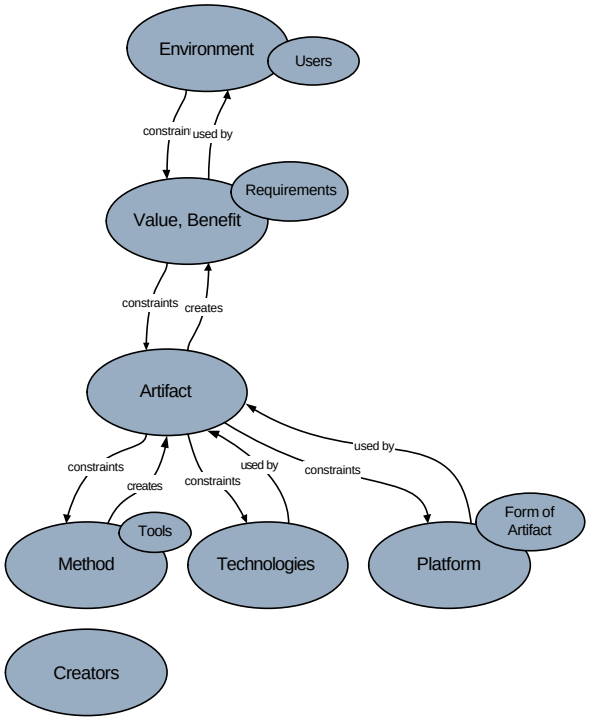

Figure 6 - Aspects of Software Development

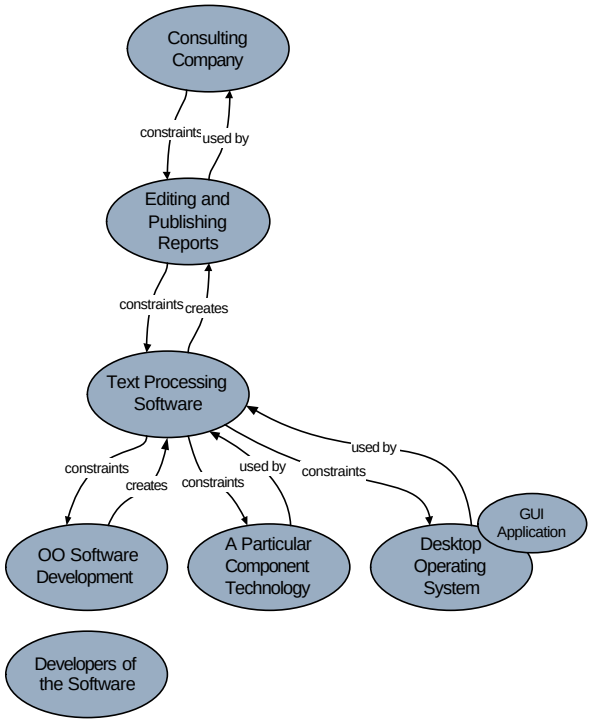

Figure 7 - Aspects of Software Development (Example)

It is interesting to observe that a layer may be seen to constrain and, in turn, be satisfied by, the immediately underlying layer. Figure 7 offers an application of this model to a consulting company (Environment) which needs the facility to edit and publish reports (Value). This facility is provided by a text processing application (Artifact) being created as a GUI Application (Form of Artifact) running on top of a desktop operating system (Platform). Conversely, the services of the operating system are used by the text processing software, which in turn provides the facility of editing and publishing reports. This facility again is of use to the consulting company.

The constraints-relationships can, however, be seen to be bidirectional: while a desired artifact (planned text processing software) constrains choices for the underlying form-of-artifact (GUI application suitable, web application unsuitable) and the related platform (desktop operating system suitable, web unsuitable); a given platform constrains the possible forms-of-application which can run and these, in turn, constrain the applications which can be implemented.

4. Sketching a Conceptual Model of Web Engineering

In the preceding section, we structured the area under discussion by identifying some aspects of software and the relationships between them. We now extend the model by explicitly contrasting software in general and web-like applications – illustrated by the model shown in Figure 8.

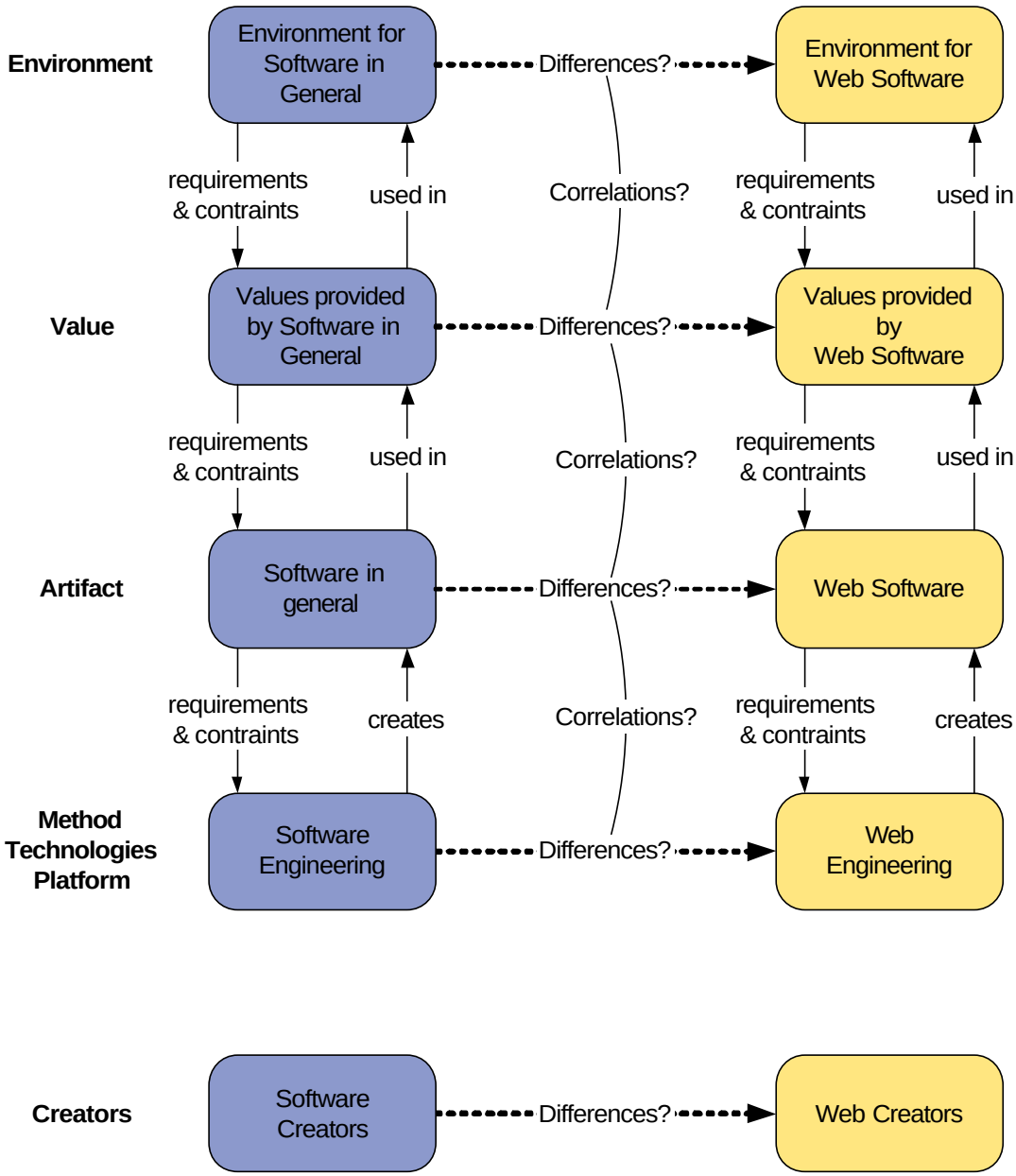

Figure 8 – Software Engineering vs. Web Engineering

Figure 8 suggests an approach to analysis in which we might first explore the differences at each level individually and then consider the differences on the various levels in conjunction with each other. It might be interesting to take into account that differences at one level can lead inexorably to differences at another level. For instance, the distinguishing features of web software may require special development methods.

Figure 8 presents the contrast between "general software" and "web-like software" as a peer-to-peer relationship – but that is, in fact slightly misleading or, at the least, limiting. In fact, web-like applications form merely one example within a range of possible kinds of software, each of which can be thought of as a sub-class of general software. Rather than being peer-to-peer, then, the relationship is a generalization/specialization relationship. Although our interest is restricted to web-like applications, we have developed the model which supports our analysis in a way which will also support researchers concerned about other classes of software. At each layer of our model (and in respect of the Creator cluster) we seek to describe a useful classification scheme. These ideas are captured in Figure 9.

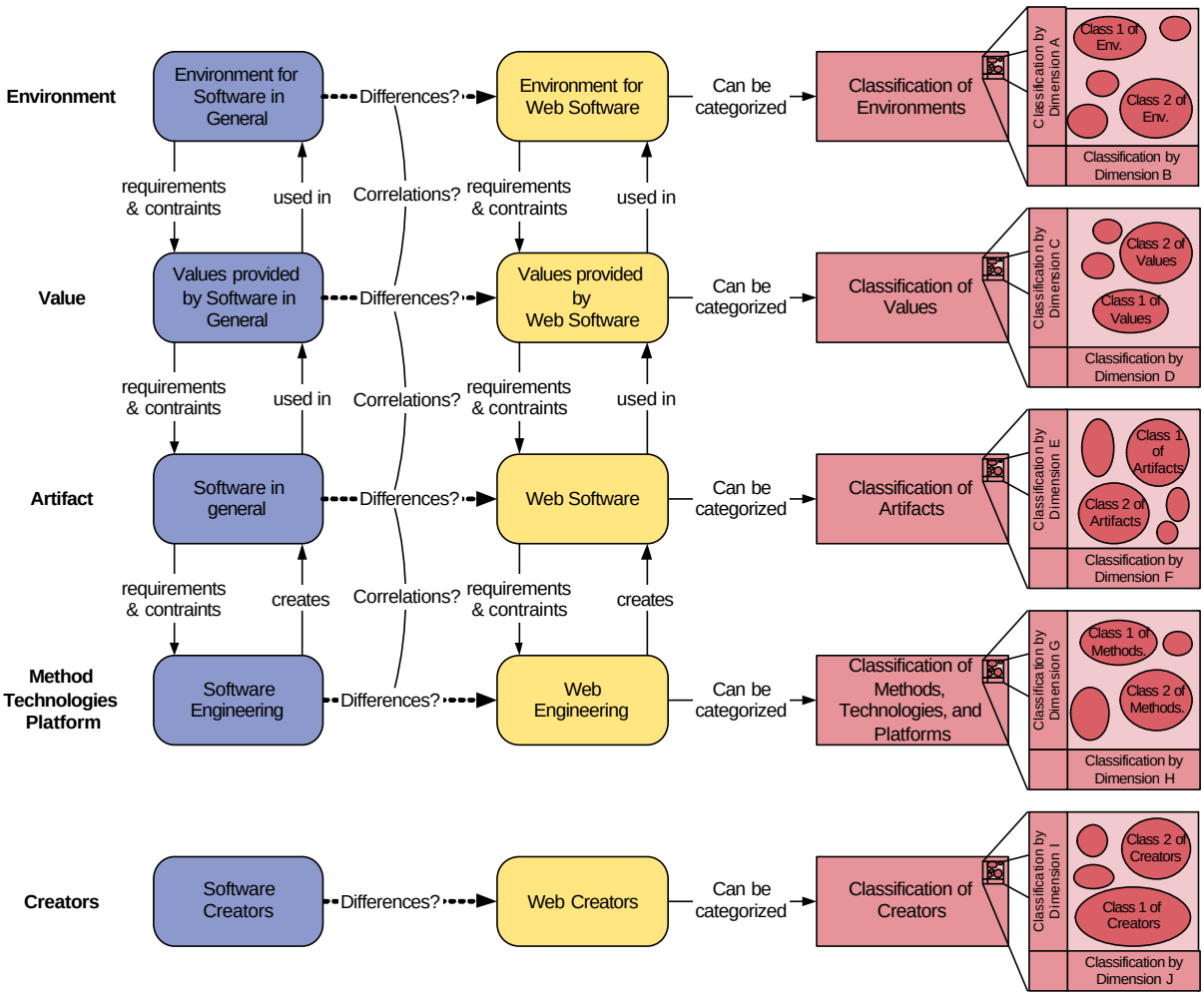

Figure 9 – Software Engineering vs. Web Engineering + Categorization

We may now extend our analytic model, to further support investigating the relationship between layers through a structure for reasoning about the relationships between the classification structures from layer to layer. We could, for example, consider that a certain environment category (consulting company) indicates that certain values are required (preparing and publishing reports) or that a certain value (ability to process images) requires a certain type of software (image editing software), which in turn constraints the choice of software platforms (desktop operating system suitable, web unsuitable).

This knowledge about the correlations between the various levels could be valuable since it, in principle, opens up the chance to formulate recommendations. The existence of a particular occurrence at the environment and/or value level can then lead to a suggestion of the form of an artifact or the selection of an appropriate software development approach from our set of available Methods, Technologies and Platforms (Figure 10).

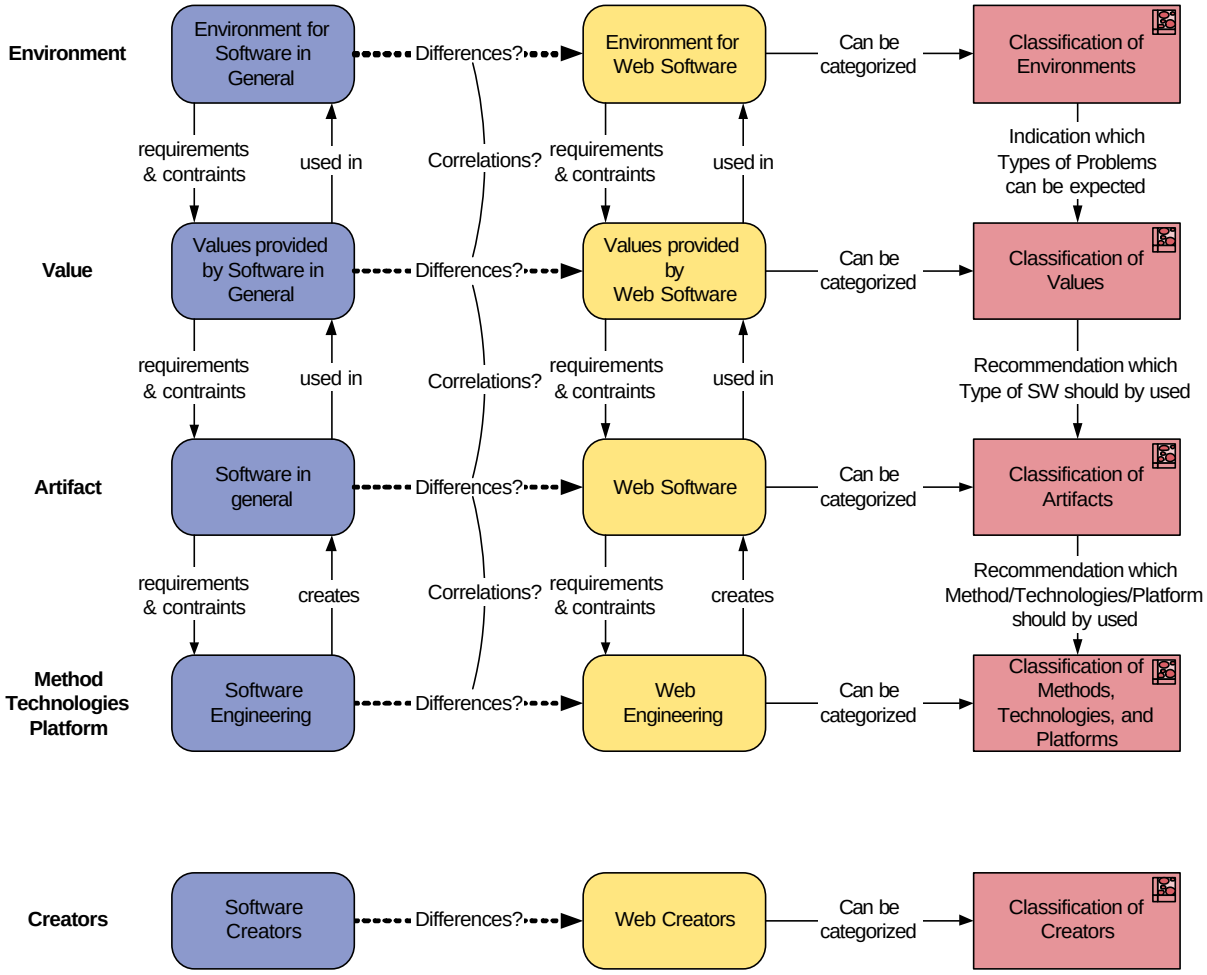

Figure 10 – Software Engineering vs. Web Engineering + Categorization + Recommendations

Finally, it is necessary to consider the cluster "Creators" which clearly does not fall into the concept of a layered model as we have discussed it in this section of the paper. It is perhaps easiest to see the characteristics of the Creator of the software as orthogonal to the layered model and as suggesting a chain of logic impacting on the methodological choice, largely independent of the relationship which we have suggested exists linking the problem-in-context and the solution strategy. The extent to which one might expect to find the existence of a level of homogeneity within the class of creators of a specific class of software and a distinction between the classes of creators of differing classes of software remains unclear and is currently the subject of further research.

5. A Preliminary Application of the Conceptual Model

We are now in a position to use our analytic model to structure an examination of the relationship between web-like applications and conventional software. Appropriately for a requirements engineering workshop, we focus our attention, in this paper, primarily on the environmental layer and the interface between the environmental and value layers of our model (see Figure 11). Notwithstanding this primary focus, however, some of the issues we discuss have impact beyond these layers and we do sketch out a pursuit of these issues.

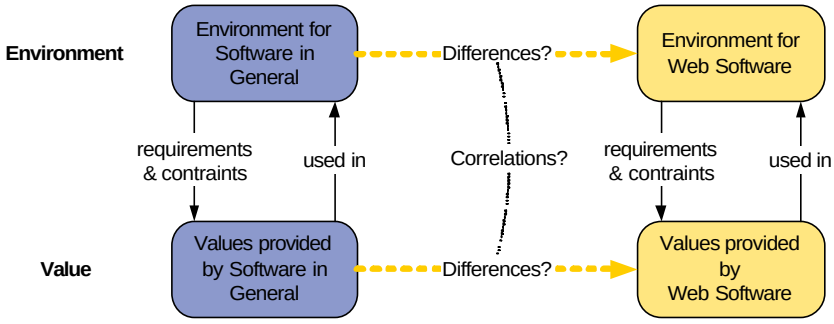

Figure 11 – Detail of the conceptual model presented in the preceding section

5.1 Exploring the Context

The first issue on which we focus relates to the potential user base for the application – at an individual and collective level – and to the relationship between the "participants" in the application. The relationships between the participants in conventional software are typically well defined. Consider the following basic categories of conventional software:

The client for an intra-organisational software development is typically considered to be "the organization". End users of such software relate both to each other and to the organization as a whole in a way which is generally consistent with the organizational relationships which exist beyond the scope of the system. End users are, in general, identifiable, relate to the system categorically (rather than individually), and are subject to political pressure to conform to organizational norms and expectations.

The clients for inter-organisational software (such as systems supporting supply chain management, logistics, collaborative engineering) development may, by simple extension, be considered as the collaborating organizations. We anticipate the relationships between organizations mediated by the system to be relatively predictable. Within each organization, we might reasonably expect users to share the characteristics of the users of intra-organisational systems – that is to behave in a relatively predictable manner commensurate with their organizational behaviour more generally.

Conventional packaged software systems share similar user-base characteristics. We can, perhaps most easily see this when we consider the organizational change management task associated with the installation of an ERP package such as SAP R3. Irrespective of any "tuning" of the software undertaken at the organizational level, it always remains necessary to "coerce" end users of the system to behave in respect of the system in a specific organized and predictable way.

Packaged software for the individual (e.g. Microsoft Word) forms a final category of conventional software. Certainly, organizational standardization policy can be a basis for an organizational decision to coerce conformant intra-organisational user behaviour. Nonetheless, the potential user base extends beyond any individual organization (indeed, it extends to non-organisational use).

One characteristic shared in installations of all these categories of software are relatively high switching costs. In the first three cases these costs are clear. In the fourth case, it is interesting to contrast use of a traditional software application against that of a web application:

- In the first case you must identify, procure (including payment and delivery), install, configure and finally run and use the software.

- In the second case you must identify and possibly pay for a web application – immediately after that it can be used.

For example, compare signing in for an online messaging service like Yahoo! Mail to the purchase, installation and usage of a desktop messaging application like Microsoft Outlook.

The setup costs (or the costs for switching between alternatives) are, in general, lower for web sites and web applications than for traditional desktop applications (here we ignore issues associated with purchase price). Reasons for the cost reduction include:

- Web applications are already installed, simply waiting for additional users.
- Web applications deliberately restrict their style in ways which are consequent on the decision to use web technologies (e.g. only user interfaces which can be described by HTML are supported).
- The use of open standards based mechanisms (HTML, HTTP, URI, …) enhances the substitutability of systems.

Reduced switching costs significantly impact the relationship between the user of software and the developer. Insignificant switching costs and high substitutability form a basis for an alternative software business model – suggesting a transition from software purchase towards pay-per-use (Cisco Systems 2002).

5.2 User Involvement in Informal/Unstructured Distributed Applications

In some cases, WLAs may simply replace traditional software applications – the user interface may be built using differing technologies, but the relationship between the application and its Environment remains traditional. In such a case, the client, the developers and the users might be members of the same or directly cooperating organizations. While such applications are subject to analysis through the lens of our model, they are uninteresting to discriminate at the analytic level which forms the focus of this paper. Rather, we concentrate on those WLAs which, due to the platform, form an extended opportunity for software support.

There is an opportunity for WLAs to have a more distributed, global nature than do traditional software applications. Developers and users are organizationally independent and geographically distributed. As examples, consider public, global messaging services as they are offered by Yahoo! or Microsoft (Microsoft Corporation 2002b; Yahoo! 2002). Technically, the client platform is unknown and uncontrollable. It is intrinsic to the web that differing users have differing technical setups (browser software, plug-ins). Assumptions about available resources (display size, fonts) are not possible. Perhaps, more significant, users themselves are unknown and their behaviour unpredictable. Users are in charge of navigation. They may leave a web site (or exit a web application) whenever they wish. As we have discussed, switching costs are low and alternatives just a click away. User loyalty consequently becomes an important challenge for providers of web applications (Nielsen 1997). As a result of the organizational and geographical distance, collateral support measures (introductory training, user education, help desk) are also harder to implement.

The challenges of organizational detachment and global distribution point to the importance of user involvement to builders of WLAs:

- It is widely accepted, one might even say it is unarguably the case, that user participation is beneficial in systems development generally.

- Analyzing user behaviour, not just in theory (through e.g. interviews) but in practice can help specification of better, more acceptable – even user-seductive – system. Flaws in, for example, the user interface can be found and rectified and positive features can be recognized. This allows learning for the future, for example by identifying successful design patterns for forthcoming systems.

After preliminary requirements have been gathered, provisional design decisions are made and the first versions of the system built, praxis-based analytical methods become vital. We must differentiate between analysis during build time and during usage time.

- *Analysis during build time* – In conventional software development, prototypes can be tested by a sample of users throughout the various cycles of development prior to completion of the final system. A similar strategy can be adopted in WLA development, but there are significant additional difficulties in sampling the potential user-base satisfactorily – these difficulties are both practical (due to geographic diffusion) and theoretic (often it not possible to identify what constitutes the potential user-base).
- *Analysis during usage time* – In comparison to conventional software it is relatively easy to analyze the real life usage of web applications after they have been deployed – irrespective of global dispersion.

It has been convincingly argued that it is dangerous to rely on the expressed opinions and memories of the users, empirical analysis of their *actions* is necessary (Nielsen 2001). However, empirical usage data is automatically recorded by all major web servers. Data collected includes information about the client (IP address, browser software) and all requested resources (e.g. pages, images…). Simple usage data (hits, page views) can be derived directly. Through such additional information as the referrer field in the web server log and session identification mechanisms (e.g. cookies, session IDs) one can then isolate information such as click streams and user sessions. These data can then be analyzed for more advanced usage patterns by employing the methods of data mining. Related research activities can be summarized under the label "Web Usage Mining" (Srivasta et al. 2000). An additional option is the analysis in combination with user/customer profiles.

For the future one can expect that technological developments will continue to have influence in the requirements and usability area. For example, a growing spectrum of front-ends for a system such as mobile devices and voice interfaces should be considered. The challenge of dynamic user expectations becomes even greater when considers offering not just a WLA, but rather an application offered (appropriately in each case) over multiple WLPs (multi-channel WLA) (Botterweck 2000).

There are, of course, numerous activities during the development process in which one hopes to involve end users. In the early stages this predominantly means the gathering and validation of requirements (Fuccella et al. 1998). Other researchers are studying environments where designers and users together can interactively draw up the future system (Klemmer et al. 2001). The developer of a distributed WLA for a potentially global user-base faces a range of difficulties which the selected development method must address – one specific issue is that many tools (such as interviews, focus groups) require what we may term "broadband personal communication". It is clear, therefore, that the constraints on effective communications between the developer and the user group at both individual and group level form an important characteristic on which the choice of an effective development methodology is dependent.

5.3 Technical Considerations

We said earlier that our discussion would necessarily extend beyond the top two layers of our analytic model. It is clear that the fundamental character of the underlying WLP constrains, in a range of ways, the possibilities for satisfying a potential user base. In contrast to desktop applications, a web application must deal with an unknown (certainly imperfectly known) user base and the characteristics of global wide area networks – characteristics which include:

- latency (signal dispersion plus delays in networking components)
- risk of disconnection or packet loss
- the trade-off between bandwidth and costs
- uncontrollably heterogeneous bandwidth across the system
- a public and therefore insecure infrastructure

Consequently, we must pay greater attention to certain quality criteria we expect from software (ISO 1991) such as reliability, robustness and security.

These challenges have to be dealt with through the design of the application platform and of the application itself. One example of such mechanisms is the request-response-style communication between a web server and a web client via HTTP (Fielding et al. 1999) intrinsic to the world wide web and which is suitable for the global communication between loosely coupled components of an hypermedia system.

6. Conclusion

In this paper, we have introduced, from first principles, our preliminary work in developing an analytic framework which allows us to contrast the problem of developing web-like applications (WLAs) against our accumulated understanding of software systems development. The analytic framework will form a basis for the development of a contingent approach to selection of methods, tools and techniques, then integrating these within a suitable methodological process, for WLA development. This approach allows us to make use of our existing understanding of software engineering methodology, while alleviating the danger of relying on the consequences of assumptions within that literature which do not hold, or which hold imperfectly, in the domain of WLA development.

We proceeded to illustrate how our approach could provide structure for the analysis of the characteristics of the potential user base of a WLA vis-à-vis the user base for a conventional software system. This analysis suggests additional and differently weighted criteria for the selection of tools and methods of requirements engineering and other user interaction for WLAs. Work continues to define the model in more detail – and, in particular, to further develop the dimensions of classification and demonstrate the logical inter-layer linkages.

References

Berners-Lee, T. (1990): *Original Design Issues*, W3C. http://www.w3.org/DesignIssues/

Berners-Lee, T., et al. (1998): *RFC2396 -- Uniform Resource Identifiers (URI) -- Generic Syntax*, The Internet Society. http://www.ietf.org/rfc/rfc2396.txt

Bieber, M. (2000): Hypertext & Hypermedia: Definition, *Encyclopedia of Computer Science (4th Edition)*, pp.799-805, Hampshire, UK, Nature Publishing Group. http://www.cs.njit.edu/~bieber/pub/cs-encyclopedia/hypertext.html

Botterweck, G. (2000): *Einsatz von XML und WAP zur Realisierung strukturierter, ubiquitärer Informationsdienste*, Koblenz, Universität Koblenz. http://www.uni-koblenz.de/~botterwe/publications/Botterweck2001_SUInfodienste_PrintQualityA4.pdf

Cisco Systems (2002): *Application Hosting Services: Opportunities for Service Providers (Business Case)*. http://www.cisco.com/warp/public/cc/so/cuso/sp/webhost/aphs_bc.htm

Conklin, J. (1987): Hypertext: An introduction and survey, *IEEE Computer*, Vol. 20, No. 9, pp.17-41.

Fernández, M., et al. (1998): Catching the boat with Strudel: experiences with a Web-site management system, *International Conference on Management of Data and Symposium on Principles of Database Systems*, pp.414 - 425, Seattle, Washington, United States. http://doi.acm.org/10.1145/276304.276341

Fielding, R., et al. (1999): *RFC2616: Hypertext Transfer Protocol -- HTTP/1.1*, The Internet Society. http://www.ietf.org/rfc/rfc2616.txt

Fuccella, J., et al. (1998): Web Site User Centered Design: Techniques for Gathering Requirements and Tasks, *ITG Newsletter*. http://internettg.org/newsletter/june98/user_requirements.html

Gnaho, C. (2001): Web-Based Information Systems Development - A User Centered Engineering Approach, *Web Engineering - Managing Diversity and Complexity of Web Application Development (LNCS2016)*, Berlin, Germany, Springer. http://link.springer.de/link/service/series/0558/bibs/2016/20160105.htm

Google (2002): *Google Toolbar Features*. http://toolbar.google.com/button_help.html

Halasz, F. and M. Schwartz (1994): The Dexter hypertext reference model, *Communications of the ACM*, Vol. 37, No. 2, pp.30-39. http://doi.acm.org/10.1145/175235.175237

Henderson-Sellers, B. (1997): OPEN: Object-oriented Process, Environment and Notation - The first full lifecycle, third generation OO method, *Handbook of Object Technology*, Boca Raton, Florida, United States, CRC Press. http://www.open.org.au/Publications/Documents/crcchap.pdf

Isakowitz, T., et al. (1995): RMM: A Methodology for Structured Hypermedia Design, *Communications of the ACM*, Vol. 38, No. 8, pp.34-44. http://doi.acm.org/10.1145/208344.208346

ISO (1991): *ISO/IEC 9126 Information technology -- Software product evaluation -- Quality characteristics and guidelines for their use*, Geneva, Switzerland, International Organization for Standardization, International Electrotechnical Commission.

Klemmer, S. R., et al. (2001): The Designers' Outpost: A Tangible Interface for Collaborative Web Site Design, *CHI Letters, The 14th Annual ACM Symposium on User Interface Software and Technology: UIST 2001*, Vol. 3, No. 2, pp.1-10.

Microsoft Corporation (2002a): COM: Delivering on the Promises of Component Technology. http://www.microsoft.com/com/

Microsoft Corporation (2002b): *MSN Hotmail*. http://www.hotmail.com/

Nielsen, J. (1997): Loyalty on the Web, *Jacob Nielsen's Alert Box*. http://www.useit.com/alertbox/9708a.html

Nielsen, J. (2001): First Rule of Usability? Don't Listen to Users, *Jacob Nielsen's Alert Box*. http://www.useit.com/alertbox/20010805.html

Ragget, D., et al. (1999): *HTML 4.01 Specification*, World Wide Web Consortium. http://www.w3.org/TR/html401/

Rossi, G., et al. (1999): Improving Web Information Systems with Navigational Patterns, *Proceedings of the 8th International Conference on the World Wide Web*, pp.589-600, Toronto, Canada. http://www8.org/w8-papers/5b-hypertext-media/improving/improving.html

Schwabe, D., et al. (1996): Systematic hypermedia design with OOHDM, *Proceedings of the seventh ACM conference on Hypertext*, pp.116-128, Bethesda, Maryland, United States. http://doi.acm.org/10.1145/234828.234840

Srivasta, J., et al. (2000): Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data, *SIGKDD Explorations*, Vol. 1, No. 2, pp.12-23. http://viror.wiwi.uni-karlsruhe.de/webmining/bib/pdf/Srivastava2000.pdf

Sun Microsystems (2002): *Enterprise JavaBeans 2.1*. http://jcp.org/jsr/detail/153.jsp

VoiceXML Forum (2002): *Get the Spec - VoiceXML Specifications*. http://www.voicexml.org/spec.html

W3C (2002a): *Amaya - W3C's Editor/Browser*. http://www.w3.org/Amaya/

W3C (2002b): *The Annotea Project*. http://www.w3.org/2001/Annotea/

W3C (2002c): *W3C Technical Reports and Publications*, W3C. http://www.w3.org/TR/

W3C (2002d): *W3C Web Services Activity*. http://www.w3.org/2002/ws/

WAP Forum (2002): *WAP Forum Specifications*. http://www.wapforum.org/what/technical.htm

Yahoo! (2002): *Mail service offered by Yahoo!* http://mail.yahoo.com/