

WEB ENGINEERING FOR INTRANETS: RETHINKING SOFTWARE ENGINEERING

Edgar Weippl¹, Ismail Khalil Ibrahim¹, Wieland Schwinger¹, Werner Winiwarter²

1) *Software Competence Center Hagenberg*
Hauptstr. 99, A-4232 Hagenberg, Austria
{*edgar.weippl, ismail.khalil-ibrahim,*
wieland.schwinger}@*scch.at*

2) *Electronic Commerce Competence Center*
Siebensterngasse 21/3, A-1070 Vienna
werner.winiwarter@ec3.at

KEYWORDS

Security, Web-based training, secure Web access

ABSTRACT

The principles of software engineering are to some extent well established and different methods have been put into context in everyday work. Web engineering, a rather new field of research, is however not yet explored in a very systematic way. Most Web applications are designed in an ad-hoc manner. They are not well documented and very difficult to maintain. In this paper, we present how methods and models of software engineering can be used for Web engineering without too much adaptation. We show that the real challenge in Web engineering is to adapt project plans and management to constant change of the underlying technology. Web engineering should be understood as a framework to provide a continuous service, not a single, well-defined project.

1. INTRODUCTION

During the last decade the Web has evolved into a global environment addressing applications that range from small-scale and short-lived services to large-scale enterprise applications distributed over many sites. Furthermore, enterprises are using the Web to implement business processes for their employees, to communicate with their partners, to connect and integrate their back-end systems, and to perform e-commerce transactions.

Consequently, a new discipline called Web engineering is evolving, which deals with the establishment and use of scientific, engineering, and management approaches to support the development, deployment, and maintenance of distributed Web applications. From a software and data engineering perspective the Web is a new application domain. As it is generally the case with new domains, engineering initially focuses on enabling technologies and tools for ad-hoc development.

However, the tremendous growth of the Web is now leading to large-scale distributed applications, which increasingly contain highly dynamic and interactive components, and handle gradually more sensitive and valuable content. Therefore, research currently concentrates on distributed object technology, component models and architectures for Web applications specifically in e-commerce and similar strategic areas, and on security aspects within Web-based enterprises.

The main difference between Inter-, Intra- and Extranet is its audience. The Internet is an external web, whereas the Intranet is an internal web, which can only be addressed from users inside the company. The Extranet can be described as "a blend of the public Internet and the closed Intranet" [11].

The Internet is a technology for organizations providing means to communicate with the public. For confidential information an Intranet for communicating partners within the company and an Extranet for those outside can be used to allow only authorized people to access it. The visual design of Intranets respectively Extranets, should be similar to the one of the Internet, with slight differences "to

help emphasize the closed and confidential nature of the Extranet" [11].

The growth of the World Wide Web has already had a significant impact on our personal and working lives. A couple of years ago, it all started with static HTML pages. The Web was mainly a form of publishing documents. The way people had to write changed with the possibility to link documents and to provide information as hypertexts. However, since the content was static, organizing Websites was primarily an editorial task and did not really raise engineering issues.

The next steps towards interactivity were forms whereby users could send data back to the Web server and simple "applications" like guest books were possible. The rising demand for interactivity lead to the development of scripting languages (e.g. JavaScript, Visual Basic Script). Later on, Java applets and ActiveX allowed distributing small applications via the Internet. As soon as databases could be accessed via Web interfaces, fully-grown business applications were developed and people discovered the advantages of "net-computing". Everyone expected the rise of networked thin client computers.

Despite the failure of ubiquitous network computers, Intranets indeed offer functionality that used to be provided by full-scale fat client applications, only. One often cited advantage of network computers and also of Web technology is – in our opinion the ultimate rationale for Web engineering – that due to centralized administration, applications can easily be updated.

However, there has been a proliferation of different scripting languages and many Web sites do not work reliably because of their extremely chaotic maintenance. So one may ask project managers whether they have forgotten everything they learned in their software engineering classes. This paper aims to give an indicative overview over software engineering technologies that have proven to be useful for Web engineering. We point to ongoing research on dealing with the improvement of existing techniques to adequately address Web engineering issues.

2. WEB ENGINEERING

Despite the fact that most Web administrators have already become used to the chaotic state of

some Intranets and many Internet Web sites, people have realized it was high time to introduce engineering standards in Web projects. Due to the number and the sophistication of Web-based applications there is reason for legitimate concern about the manner in which they are created and their long-term quality and integrity. Web engineering is concerned with establishing and using sound scientific, engineering and management principles for developing Web-based applications. We follow Conallen's definition of a Web application being "a Web system (Web server, network, HTTP, browser) in which user input (navigation and data input) effects the state of the business" [5] (p63).

Just like traditional software engineering ensures efficient development and maintenance of software applications, Web-based applications have to be "engineered", too. Though, it is not necessary to invent many new process models, notations or programming paradigms. Most Web-based processes can be analyzed, designed, implemented and maintained using existing techniques that have already been developed for object-oriented and component-based software. However, in contrast to software engineering, Web engineering is rather concerned with delivering a service than a product.

2.1. Applications for Web Engineering

Before taking a closer look at software engineering, we have to know which kinds of applications we are likely to encounter in Intranets. We can then compare the required engineering steps to those of the corresponding "traditional" software engineering process.

An Intranet addresses the members of an organization, compared to the organization's Internet site, which is used by a completely different audience.

Furthermore, Web applications tend to be process-oriented. In order to implement, e.g. a paperless office it is necessary to redesign the processes, which create, distribute, and update information within the organization. The Intranet can facilitate communication and access to the required information.

Moreover, an Intra- or Extranet should also support the collaboration of remotely located people on business processes. Not only access to information should improve but also to people

needed to carry out the business processes. The Intranet is meant to be an easily accessible, platform-independent virtual space, integrating people, processes and information. Therefore the Intranet is a tool and a model for an efficient, process-centered organization. This universal, single point of access for all strategic information – often referred to as “corporate information network” – should help to enhance the organization's goals.

2.2. Revisiting Software Engineering

Software engineering covers all the major topics associated with software architecture: what software architecture is, its quality attributes, architectural styles, enabling concepts and techniques, architecture description languages, development of product lines, etc. [1].

Nevertheless, software engineering is more than a collection of tools and techniques. It is a complex organizational problem involving many people with various educational and professional backgrounds. Therefore, it is essential to follow a standardized way of developing software. Various models to evaluate the process of software engineering have been proposed, among them CMM and SPICE.

The CMM (Capability Maturity Model) describes an evolutionary improvement path from an immature process to a mature one. The CMM is appropriate for developing when working with other organizations that are CMM compliant. CMM is also very useful to focus on the management of software development, not the development process itself [15].

SPICE (ISO 15504), on the other hand, provides a framework for assessing software development processes. SPICE might be the best choice if one is involved in maintenance as well as in development. It is particularly appropriate for a small organization that needs to be able to show the results of specific improvement efforts. Because the result of a SPICE assessment is a profile of individual capabilities in many areas, it can make small-scale improvements more visible than a CMM assessment [15].

Frameworks like SPICE are useful for a helicopter view on the software engineering process. On the other hand, there have always been considerable efforts to develop standardized frameworks in a bottom-up way. The SWEBOK [4]

is a fabulous collection of references to current knowledge in software engineering. It is a guide that provides a “consensually-validated characterization of the bounds of the software engineering discipline and [it provides] a topical access to the Body of Knowledge supporting that discipline.” Being a bottom-up approach, it promotes UML as a modeling language that can be used throughout all stages of software engineering, i.e. analysis, design, implementation, testing and maintenance.

Today, UML [3] is a very popular notation that many practitioners use in their every-day work. Despite the fact that UML is used a lot and that this set of notations covers all cycles of software engineering, it is by no means the ‘best’ notation. According to the SWEBOK “there is little empirical evidence to support claims for the superiority of one notation over another” [4] (p30: 2-14-163).

2.3. The Transition from Software Engineering to Web Engineering

In the previous section we have given an indicative overview over references of the state-of-art in software engineering. In this section we will focus on how we can use the existing knowledge in Web engineering. As Web projects often start with a couple of HTML pages and continuously grow to large Web applications this section is also organized in a bottom-up way.

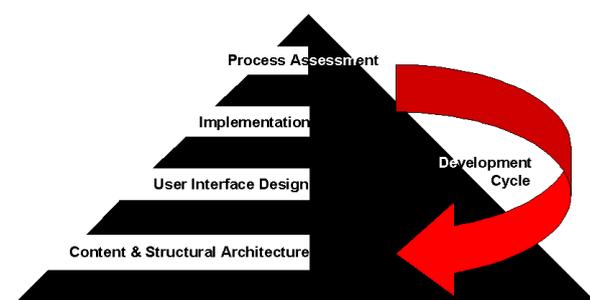


Figure 1: Bottom-Up Approach for Web Engineering

According to Powell [14] (p47), the following few aspects make Web sites different from software:

- Web sites will continue to be document-oriented, having static pages, at least to some degree.

- Web sites focus on “look and feel”; this is especially true for Internet sites, which are often driven by marketing goals.
- Web engineering is more content-driven than software engineering. The development of the content is often included in Web projects, whereas this is not normally part of software engineering projects.
- The Web is generally more unpredictable and not as well understood as software.

2.3.1 Content & Structural Architecture

As UML is a very common notation, it is obviously desirable to model the content of a Web project with UML, too. Conallen [5] has published a paper on how UML can be used to model Web application architectures. Stereotypes should be used to define a semantic meaning for a modelling element. Constraints can be added so that “the well-formedness of a model” [5] (p66) can be defined. Using these features, Conallen proposes to create two separate models: one reflecting the client’s perception of the Web application (see Figure 2), i.e. a set of hyperlinked HTML pages, and another model representing the server-side view (see Figure 3).

This approach is useful as both the user’s view and the technical details of how the content’s structure can be modeled are outlined.

Figure 2: Client view of a Web

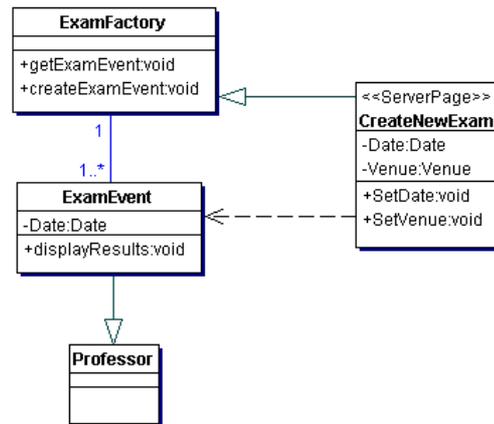
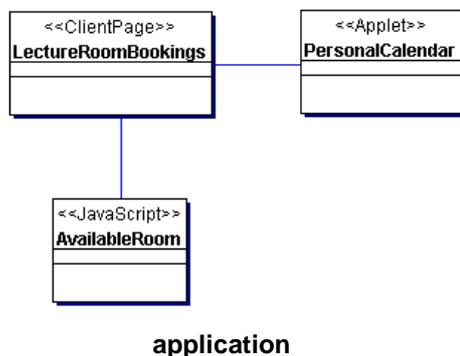


Figure 3: Server-side view of a Web application

2.3.2 User Interface Design

Designing the UI is a significant part of most applications [10] but it is *the essential aspect* in Web engineering. One major weakness in almost all Web projects is the fact that modeling the UI is very difficult. The specification of UML [12] is by no means clear on how a UI should be modeled. Pinheiro [13] presents results of a case study showing how to model UIs with UML. The main problem is that the UML based notations of UI design do not scale well and quickly become very confusing. According to Pinheiro, modifying the ‘presentation framework’ design pattern described by Gamma et al. [8] seems a promising approach to deal with this shortcoming.

2.3.3 Implementation

Once the UI has been modelled and the content’s architecture has been structured, there are several technical issues to be resolved [4].

All distributed applications and thus obviously Web applications, need some form of concurrency control (overview in Meyer [9]) to avoid inconsistent states and deadlocks. Similar to traditional software engineering, control- and data-flows have to be designed and events (overview in Bass [1]) and exceptions (Meyer [9]) handled. The difference however is, that the state information concerning individual users is difficult to maintain reliably. Users may not allow cookies to be set,

their real IP address is often hidden behind a proxy or pages are retrieved from a cache. Furthermore, the number of concurrent users may vary enormously and is often hard to predict in the long run. Therefore performance predictions tend to be imprecise, thus endangering high availability of Web-based services.

The Web has also brought a new understanding of platform independence (overview in Dsouza[6]) as applications do not have to be recompiled because Java object code can be executed on any Virtual Machine. Web Applications tend to be built even more modular and the goal of separating UI classes from functionality can often be achieved.

Taking all that together one can say that most of the problems encountered in Web engineering have been dealt with in a similar way in software engineering. The challenge is to adopt these well-established techniques to fit to Web applications that are highly distributed, much more modular and frequently modified.

2.3.4 Process Models

Web applications cannot be planned even two years ahead due to rapid changes in technology. This fact has to be considered when choosing a model for the whole process of Web engineering. We have found that the spiral model [2], a meta model, is suited best because it allows to use different models at various stages.

Despite its flexibility, we think that the spiral model is quite dangerous for project managers who have little experience with Web projects. Problems may arise if the various stages are not exactly defined and documented, as the whole development process will tend to end up in a complete chaos.

In order to avoid this, it is extremely important to evaluate these processes. We have found SPICE to be the best tool to assess Web engineering processes because the results of improvement efforts can easily be seen. As previously mentioned, the SPICE framework is especially suitable for small organizations. Since Web applications are developed mostly either in small (startup) companies or by special project teams that are exempted from a big organization's daily routine, SPICE could prove very effective for assessing Web engineering processes.

2.3.5 Quality Assurance

At the end of every Web engineering process there should be a high-quality Web application. As already pointed out, there is actually no real end because maintenance and enhancements are so essential that they should rather be seen as a part of the development. Nevertheless, Powell proposes to evaluate a Web application's quality in eight dimensions:

- Correctness (functionally and cosmetically error free)
- Testability (against Specification)
- Maintainability (for Web projects, maintenance costs exceed development costs by far)
- Portability and scalability
- Reusability
- Robustness and reliability
- Efficiency
- Documentation

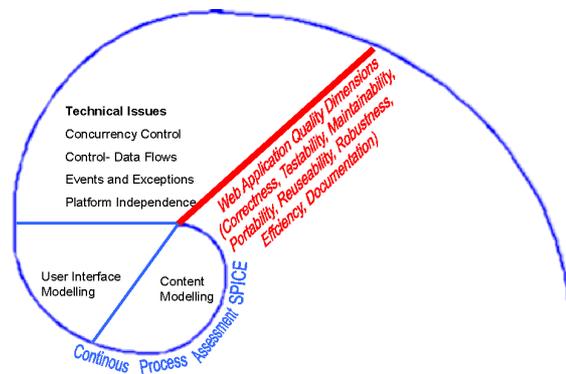


Figure 4: A Cycle in the Spiral Model for Web Engineering

3. CONCLUSION

Despite the fact that today's Internet and Intranet sites are planned, implemented and maintained in an ad-hoc way, we are convinced that sound Web engineering is possible with the tools and techniques already available. Apart from taking up the ideas proposed by researchers cited in this paper, Web designers have to change their mindsets. The days of (only) creative chaos are gone and it is the soundness of engineering principles that is here to stay.

4. REFERENCES

- [1] L. Bass, P. Clements and R. Kazman, Software architecture in practice. Addison-Wesley, Bonn, Paris, Reading, MA, 1999
- [2] B.W. Boehm, "A Spiral Model of Software Development and Enhancement", IEEE Computer, May 1988, 61-72
- [3] G. Booch, J. Rumbaugh and I. Jacobson, The Unified Modeling Language User Guide. Addison-Wesley, Reading, MA, 1999
- [4] P. Bourque and J.W. Moore, SweBok: Guide to the software engineering Body of Knowledge - A Stoneman Version 0.7. Joint IEEE Computer Society - ACM committee, <http://www.swebok.org/>, April 2000
- [5] J. Conallen, "Modeling Web application architectures with UML", Communications of the ACM, 42:63-70, 1999
- [6] D.F. D'Souza and A.C. Wills, Objects, Components, and Frameworks with UML - The Catalysis Approach. Addison-Wesley, Bonn, Paris, Reading, MA, 2nd Edition, 1998
- [7] P. Fraternali, "Tools and approaches for developing data-intensive Web applications: A survey", ACM Computing Surveys, 31:227-263, 1999
- [8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object Oriented Software, Addison-Wesley, Reading, MA, 1995
- [9] B. Meyer, Object-Oriented Software Construction. Prentice Hall, Upper Saddle River, NJ, 2nd Edition, 1997
- [10] B. Myers and M. Rosson, "Survey on user interface programming", Proc. CHI'92, p192-202, 1992
- [11] J. Nielsen, The Difference Between Intranet and Internet Design. www.useit.com/alertbox/9709b.html, 1997
- [12] Object Management Group. OMG Unified Modeling Language Specification, Version 1.3, June 1999
- [13] P. Pinheiro da Silva and N.W. Paton, "User Interface Modeling with UML", Proc. of the 10th European-Japanese Conference on Information Modeling and Knowledge Bases, p208-222, May 2000
- [14] T.A. Powell, D.L. Jones and D.C. Cutts: Web Site Engineering: Beyond Web Page Design. Prentice Hall, 1998
- [15] The Spire Project Team, The SPIRE Handbook: Better, Faster, Cheaper Software Development in Small Organisations. The European Community, 1998