Making Quality a First-Class Citizen in Web Engineering

Paul Grünbacher
Johannes Kepler University
Systems Eng. & Automation
4040 Linz, Austria
Ph: +43 70 2468 8867
pg@sea.uni-linz.ac.at

Rudolf Ramler
Software Competence Center
Hagenberg (SCCH)
4232 Hagenberg, Austria
Ph: +43 7236 3343 872
rudolf.ramler@scch.at

Werner Retschitzegger
Johannes Kepler University
Information Systems
4040 Linz, Austria
Ph: +43 70 2468 8883
werner@ifs.uni-linz.ac.at

Wieland Schwinger
Johannes Kepler University
Telecooperation
4040 Linz, Austria
Ph: +43 70 2468 9260
wieland@schwinger.at

Abstract

Quality is important in any engineering project but particularly critical when developing Web applications. In Web Engineering for example, usability, performance, or security aspects need special attention. In this paper we discuss the specific characteristics of Web applications. We explore quality-related challenges in Web Engineering and discuss a set of quality-related research issues.

1. Introduction

In the last years the Web has changed fundamentally from pure information orientation towards an application medium comprising full-fledged applications like booking systems, e-learning platforms, or online shopping malls. As a consequence, quality concerns such as security, usability, or performance become particularly critical in Web engineering [20, 23]. Compared to their predecessors modern Web applications have to meet sophisticated quality requirements which requires the in-depth consideration of quality aspects [13] and a remarkable shift from functional to quality requirements.

Although software quality is regarded as increasingly important, it is still ill defined. A number of quality models are available for software [32]. For example, in IEEE-STD-729 [1] quality is defined as the "Totality of features of a software product that bears on its ability to satisfy given needs." Another definition in the same standard specifies quality as the "Composite characteristics of software that determine the degree to which the software in use will meet the expectations of the customer." The well-known standard ISO 9126 [16] defines important quality dimensions. All the definitions are intuitive but rather vague and ambiguous and it is hard to apply them to real-world situations. It seems to be easier to actually achieve quality than to measure and define it [31].

For example, the development of a Web application with a conventional quality model in mind may result in accidentally emphasizing improper quality attributes and overlooking critical ones. Quality models specific for Web applications are still evolving. A model based on ISO 9126 has been proposed by Olsina [25]: Based on a quality evaluation of a Web site for museums [24] and the later adaptation of the underlying evaluation method for academic Web sites, the author propose a hierarchy of quality requirements (for academic Web sites) containing more than 120 quality characteristics and a catalog of metrics. Dhyani *et al.* give an overview on Web metrics for quantifying Web page significance, Web page similarity, search and retrieval, usage characterization, and information theoretic properties [8].

It is, however, not the goal of this paper to invent a new quality model for Web applications, not least since particular instances of a quality model will be highly domain specific as demonstrated by examples from information systems (focussing on data quality [4, 22]), real-time Web applications (emphasizing performance related quality attributes), or multimedia-intensive systems (requiring new quality attributes such as "attractiveness"). Rather, in this paper we argue that quality aspects of Web applications must be considered first-class citizens in Web Engineering. This is particularly important as there is still a widespread believe that the Web hasn't necessarily introduced any new quality issues. In addition, the exploration of Web-specific challenges posed on the overall quality of Web applications represents a prerequisite for developing Web-specific quality models.

For this, the paper is organized as follows: In Section 2 we discuss specifics of Web applications. In Section 3 we explore quality challenges in Web Engineering. We conclude with a set of quality-related research issues in Section 4.

2. Are Web Applications Really Different?

Web applications are software systems based on technologies and standards of the World Wide Web Consortium (W3C). They provide Web-specific resources such as content and services through a user interface, the Web browser. With this definition of Web applications in mind, we explore the differences to 'conventional', non-Web applications. We structure our discussion using application-related, usage-related, development-related and evolution-related characteristics of Web applications [17].

2.1 Application-related characteristics

When developing Web applications one has to consider not only functionality but equally address *content*, *hypertext*, and *presentation* aspects.

Content. The origin of the Web is its role as a medium to present information. Beyond the required functionality, Web applications are thus heavily content-driven. Content comprises not only structured data residing in database systems but also unstructured and semi-structured data such as textual descriptions or multi-media information. Complexity arises especially from the fact that content is often highly *dynamic* and continuously updated. Also, users typically demand high *content quality* in terms of topicality, accurateness, consistency, or reliability [29]. Consequently, the development of Web applications is not only a complex engineering task but relies strongly on *authors* responsible for the content.

Hypertext. Web applications advocate the hypertext paradigm [7] as the fundamental paradigm for structuring information. The basic elements of the Web's notion of hypertext are nodes, links and anchors. Typical examples of accessing hypertext information includes browsing (like in online stores' catalogues), querying (like in e-learning applications), or guided tours (like in virtual exhibitions). The essential feature of the hypertext paradigm is its *non-linearity* requiring from both authors and users to address the potential issues of disorientation and cognitive *overload*. This can be achieved for example through specific navigation design (site maps, keyword searches, traversed paths, etc.) and is essential to preserve *quality of access*.

Presentation. In conventional software systems the "look and feel" is often to a large extent determined by standardized user interface elements and style guides. Presentation is a central quality factor for Web applications not least to the high competitive pressure on the Web where *visual appearance* is subject to (ever-changing) fashion, trends, and new technical features. In addition, as application designers cannot expect Web users to consult a user's manual Web applications need to be *self-explanatory* requiring particular attention to visual design and the consistency of the interaction style behaviour.

2.2 Usage-related Characteristics

Unlike in more traditional settings, the users of Web applications often vary in numbers and cultural background, use heterogeneous devices and can freely chose the time and location of accessing the Web application [18]. Developers frequently cannot predict all these potential settings.

Varying usage context. This includes aspects of the *location* and *time of access*, offering the opportunity of new kinds of context-based services, not least due to the advent of mobile computing. In addition, the possibility of *immediate* and *permanent availability* of Web applications requires special quality considerations such as 24/7 availability.

Unpredictable technical infrastructure. Available end-user devices vary in hardware and software capabilities such as display size, computational power, or browser version. Also network connections differ with respect to bandwidth, reliability, stability, and availability, all affecting the *quality of service*. Complexity is increased even further due to the fact that the actual representation of the Web application on the client device is to a large extent outside the control of the developers. For example, users configure their browsers individually and may even disallow certain essential features (e.g., cookies or JavaScript).

Diversity and magnitude of user base. Web application users differ in age, social and cultural backgrounds, goals, intentions, skills, or capabilities. These heterogeneity has to be considered by application developers since the Web entails no obligation and Web applications will only be used if they bring immediate advantage [15]. The way users interact with the Web application can be hardly predicted and users may leave the Web application at any time. Also, the number of users accessing the Web application may vary considerably making *scalability* another crucial quality aspect.

2.3 Development-related Characteristics

Web application developers need to deal with conditions, risks, and uncertainties not always present in traditional software projects.

Development team. Web application development is a *multi-disciplinary* effort comprising a mixture of print publishers, authors, software developers, marketing experts, engineering, or art designers. Such teams are also dominated by significantly *younger team members* which are less willing to adhere to conventions and more inclined towards applying new (and often still immature) technologies. Another important characteristic is the involvement of *open source communities*.

Development environment. The technical infrastructure used for developing a Web application is characterized by a high degree of *volatility and heterogeneity*. Web application development relies on a broad spectrum of different COTS components (e.g., Web server, application server, database system, publishing framework, etc.). Because of the increased time-to-market pressure these components are often immature and fall short in stability, reliability, and desired functionality.

Legacy integration. Web applications often need to integrate legacy systems. The external services provided by these systems are, however, rarely documented and often change without notice, thus negatively affecting the quality of the overall Web application.

Process. Web application development processes are

characterized by frequent changes and adjustments, which are necessary due to rapid technological developments, fast changing trends, volatile requirements, and rigid schedules. This calls for highly iterative, flexible, and prototype-oriented development methods [3, 5].

2.4 Evolution-related Characteristics

Web applications are subject to frequent changes and permanent evolution: Their development is driven by rapidly changing technology and the volatility of Web users leads to a highly competitive situation where immediate Web presence and short time to market are considered crucial: "Unlike conventional application software that evolves over a series of planned, chronologically spaced releases, Web applications evolve continuously." [26].

3. Quality Challenges in Web Engineering

The discussion in the previous section has revealed some distinct characteristics of Web applications often not present as such in conventional applications. These characteristics are tightly intertwined with all other bits and pieces of a Web application and have a tremendous impact on the overall quality: "… from the evaluator's point of view, it is not possible to separate the quality of data from other aspects like for example the quality of the provided navigation functions and their usability." [2]. These characteristics lead to specific quality Web Engineering challenges we'll discuss by using the following categories:

(1) *Elicitation* challenges, i.e., how to acquire and negotiate quality-related needs from system stakeholders.
(2) *Engineering* challenges, i.e., how to actually build the system with the desired quality level.
(3) *Operational* challenges, i.e., how to maintain the desired level of quality during runtime.

3.1 Elicitation Challenges

As Web applications and their related concepts and metaphors are still new to many users, they have difficulties to develop realistic expectations and to express their needs (IKIWISI [5, 20] is a typical phenomenon). The failures in the early days of e-business [14] are an impressive witness of the mismatch between end-user expectations and provided solutions. The lack of experience, tradition, and settled general knowledge has frequently led to misunderstandings and false expectations regarding basic quality attributes and their reasonable extent among stakeholders – customers, end-users, designers, and developers alike. Simply put, the perceived quality, defined in accordance to ANSI/IEEE Std. 729-1983 ("… the degree to which the software in use will meet the expectations of the customer") was low, although the solutions per se were often state-of-the-art at the time of development. In recent years, experience has grown and expectations became considerably more realistic. Nevertheless, the rapidly changing technology and the exploration of new fields requires constant recalibration of expectations to (intermediate) development results to avoid mismatches between expectations and technical feasibility.

Interdependencies and trade-offs. Elicitation is even more challenged by complex interdependencies and trade-offs among quality requirements. Quality requirements may positively or negatively affect each other, e.g., efficiency positively correlates with flexibility but negatively with portability [9]. For example, increasing the level of service of the generally conflicting quality attributes usability and performance at the same time may not be possible. To complicate things even more, the situation often looks differently when sub-attributes are considered: The experience from designing the hypertext structure of the Web application for the Olympic Games 1996 and 1998 [21] illustrates that a positive interdependency can even exist between sub-attributes of usually conflicting quality attributes. While the deep navigation structure in the first version of the Olympic Games' Web application resulted in a high workload for the Web server as all visitors accessed multiple pages to find the required information, improving the usability through a better navigation design reduced the number of pages that had to be accessed and, thus, resulted in higher overall performance. Then again, interdependencies between quality attributes may dynamically change as their level of fulfilment varies. Depending on to what degree a quality attribute is "satisfied", correlation often changes from positive over indifferent to negative, making the search for optimal trade-offs an almost impossible mission.

Constant evolution. Iterative development, frequent releases, and stakeholder involvement have been proposed to deal with highly volatile requirements and uncertainty in Web development. Prototyping-based RAD approaches have been proposed to leverage the involvement of end-users. However, these approaches require constant involvement of end-users in reviewing the evolving Web applications and in providing feedback for the developers. Also, Web sites are often perceived as being constantly "under construction" leading to confusion and frustration.

Unknown end-users. Due to usage-related characteristics of Web applications, conventional approaches to elicit requirements and to agree on a realistic baseline are difficult to apply in Web projects. It can be difficult or impossible to identify stakeholder representatives because of the potentially high number of end-users (to be partitioned in representative groups), their geographical distribution (possibly all over the world), or their "anonymity" as a result of the lack of direct interaction.

3.2 Engineering Challenges

Making quality aspects first-class citizens in Web En-

gineering is often neglected in typical engineering activities where the focus usually lies on functional requirements.

Understanding and managing component quality. The first step towards improving the quality of a Web application is to assess the quality of the different parts incorporated namely, third-party components. However, we lack means to measure quality properties and models supporting to aggregate individual component properties in order to allow simulation and prediction of system quality.

Component configuration. The more general and flexible the used components are, the more effort has to be put into their configuration, and, thus, the more errors are possibly lurking in application settings, configuration files, and registry entries. The configuration usually takes place after the component's quality has been assessed and before the component is integrated into the system. However, no appropriate test assures the "correct configuration" at this level. Configuration testing is therefore usually a by-product of integration testing, which assumes correctly working components and focuses on errors in their composition. Moreover, not only testing but also other basic quality assurance measures, e.g., version control, are often neglected for configuration data. In order to achieve quality of component-based systems, component configurations have to be treated with the same care as "real" source code.

Web testing. At present, we experience a similar situation as 25 years ago in software testing. In 1979 Goodenough replies to the question on what to test: "An obvious answer is: Test whether the product functions correctly. But … correctness alone is not sufficient. Before discussing correctness, four other behavioural properties will be considered, namely, utility, reliability, robustness, and performance." [27]. The lack of common experience and understanding of quality attributes has led to a revival of the discussion about quality attributes in testing for Web applications. Books on Web testing describe methods and tools for testing various quality attributes such as performance, usability, reliability, or compatibility. However, they often do not define the term quality or even the (possible conflicting) relations between quality attributes. As a result, rational measures for quality levels are usually neglected. In the rush to test at least something, the questions about adequate criteria to stop testing and appropriate progress metrics for Web testing are often completely ignored. Although quality models exist, at least for software products, they do not offer a clear mapping of quality factors into test methods. Furthermore, the absence of stable requirements in (evolutionary) Web development is another reason why convenient measures known from (conventional) requirements-based testing can hardly be applied.

3.3 Operational Challenges

Many of the characteristics and parts of a Web application that influence quality are outside the scope of control at the time of engineering. Challenges during operation time include the operation environment, accessed third-party services, and content/data quality in decentralised environments, to name but a few. How can quality be assured and maintained despite of such unstable and unforeseeable conditions?

Many times, simplistic measures must be applied to avoid, e.g., low content quality. For example, a popular Austrian tourism information system automatically removes a skiing resort's snow report if it has not been updated for more than three days. More sophisticated approaches are implemented by adaptable and self-adapting applications [28]. They range from load balancing to customisable Web interfaces.

However, many problems caused by future changes cannot be anticipated and resolved in advance. A typical example are new generations of Web browsers. Over the recent years, new Web browsers have been released that were incompatible to previous versions. Thus compatibility testing had to be repeated with each new version and fixes had to be applied, although the Web application itself had not changed at all.

Further challenges stem from the fact, that Web applications represent a example of a multi-stakeholder distributed system (MSDS) [12], "... a distributed system in which subsets of the nodes are designed, owned, or operated by distinct stakeholders." These nodes are often designed or operated in ignorance of one another or with different, possibly conflicting goals. A popular example is a network of Web services. In an MSDS the requirements placed by diverse stakeholders are often ephemeral and conflicting since details about the elements of such a dynamic system are largely unknown to single stakeholders and outside their sphere of control [11].

4. Research Issues

Obviously, this paper cannot succeed in deriving a complete list of research issues from the challenge areas discussed in the previous section due to space limitations. We will thus present one selected research issue from each challenge area.

Trade-off analyses in requirements elicitation. In his recent keynote talk at the 2003 Euromicro Conference Voas presented a set of Grand Software Engineering challenges. One of the challenges presented was entitled: "Designing-in" the "ilities" [31]. Voas argues that ignoring non-functional attributes is not an option and we should make an attempt to discuss them with the client even if quantification is not possible. Identifying and negotiating (feasible) combinations of quality attributes with stakeholders is one of the challenges we're interested in.

Unlike in more conventional software engineering domains, where years of experience and research have led to more mature quality models or, at least, common sense, Web Engineering lacks a similar profound background so far. Nonetheless, even though we are currently not able to (completely) define the interdependencies among quality attributes, they result in various trade-offs between quality attributes [30]. For example, it should be explored how such quality trade-off analyses could complement existing requirements negotiation approaches [6]. Negotiation is a promising approach as different stakeholders may perceive quality differently.

Aggregating component quality. Web applications typically employ a large number of off-the-shelf components and infrastructure. Their quality strongly influences overall system quality. Approaches to assure quality of third-party components are therefore essential, for open-source as well as for commercial software, but hard to accomplish. Even if we know the quality of individual components or subsystems, we lack a suitable model to aggregate the results. Summing up all values for performance surely is not an option, nor is calculating the average. Overall performance may be determined by the "weakest link in the chain" of all underlying components. But how do other quality attributes, from the various other components of the system, influence the aggregated value [30]? This research could, e.g., build on existing work on modelling the dependencies between actors, components, and system properties as described in [10].

Maintaining quality in operating Web applications. In addition to new technologies, methods, and tools, new business partners, competitors, legal issues, as well as hypes emerge, forcing organizations to swiftly adapt their business strategies and systems. Consequently, the operations phase of Web applications tends to focus on adaptive maintenance (in contrast to corrective maintenance in conventional operation) [28], and ongoing evolutionary development (in favour of operating a "finished" product). David Lowe concludes: "The evolution of Web applications is analogous to a garden changing as a natural part of its cycle of growth." [19]. Quality is therefore a constant issue in operating Web applications.

In this paper we argued that quality aspects have not received sufficient attention in Web Engineering research and practice so far and that they should be considered first-class citizens. To stimulate further discussion, we outlined some research issues that reflect the characteristics and challenges in today's development of Web applications.

References

[1] IEEE Standard Glossary of Software Engineering Terminology. 1983, The Institute of Electrical and Electronics Engineers, Inc.: New York, NY.

[2] Atzeni, P., Merialdo, P., and Sindoni, G. *Web Site Evaluation: Methodology and Case Study.* In, *Proc. of the Int. WS on Data Semantics in Web Information Systems (DASWIS'01) at the 20th Int. Conf. on Conceptual Modeling*. 2001. Yokohama, Japan.

[3] Beck, K., *Extreme Programming Explained: Embrace Change*. 1999: Addison-Wesley.

[4] Bobrowski, M., Marré, M., and Yankelevich, D., A Software Engineering View of Data Quality.

[5] Boehm, B.W., Requirements that Handle IKIWISI, COTS, and Rapid Change. *IEEE Computer*, 2000. **33**(7): pp. 99-102.

[6] Boehm, B.W., Grünbacher, P., and Briggs, R.O., Developing Groupware for Requirements Negotiation: Lessons Learned. *IEEE Software*, 2001. **18**(3): pp. 46-55.

[7] Conklin, J., Hypertext: An Introduction and Survey. *IEEE Computer*, 1987. **20**(9): pp. 17-41.

[8] Dhyani, D., NG, W.K., and Bhowmick, S.S., A Survey of Web Metrics. *ACM Computing Surveys*, 2002. **34**(4).

[9] Egyed, A., and Grünbacher, P. *Towards Understanding Implications of Trace Dependencies among Quality Requirements*. In, *2nd International Workshop on Traceability in Emerging Forms of Software Engineering*. 2003. Montreal.

[10] Franch, X., and Maiden, N.A.M. *Modeling Component Dependencies to Inform their Selection*. In, *2nd International Conference on COTS-Based Software Systems*. 2003: Springer.

[11] Grünbacher, P., Stallinger, F., Maiden, N.A.M., and Franch, X. *A Negotiation-based Framework for Requirements Engineering in Multi-stakeholder Distributed Systems*. In, *Workshop on "Requirements Engineering and Open Systems (REOS)" at RE'03*. 2003. Monterey, CA: http://www.cs.uoregon.edu/~fickas/REOS/.

[12] Hall, R.J. *Open modeling in multi-stakeholder distributed systems: requirements engineering for the 21st Century*. In, *Proc. First Workshop on the State of the Art in Automated Software Engineering*. 2002. U.C. Irvine, Institute for Software Research.

[13] Hendrickson, E., and Fowler, M., The Software Engineering of Internet Software. *IEEE Software*, 2002. **19**(2): pp. 23-24.

[14] Ho, J., Evaluating the World Wide Web - A Global Study of Commercial Sites. *Journal of Computer-Mediated Communication*, 1997. **3**(1).

[15] Holck, J., and Clemmensen, T., What Makes Web Development Different? 2002, Copenhagen Business School.

[16] ISO/IEC-9126, Information technology - Software Product Evaluation - Quality characteristics and guidelines for their use. 1991.

[17] Kappel, G., Pröll, B., Retschitzegger, W., and Reich, S., *Web Engineering: Systematische Entwicklung von Webanwendungen*. 2003: dpunkt.

[18] Kappel, G., Retschitzegger, W., and Schwinger, W. *Modeling Customizable Web Applications -- A Requirement's Perspective*. In, *Proc. of the Int. Conf. on Digital Libraries: Research and Practice (ICDL)*. 2000. Kyoto, Japan.

[19] Lowe, D., *A Framework for Defining Acceptance Criteria for Web Development Projects*, In, *Web Engineering - Managing Diversity and Complexity of Web Application Development*, Murugesan, S. and Deshpande, Y., Editors. 2001, Springer. pp. 279-294.

[20] Lowe, D., Web System Requirements: An Overview. *Requirements Engineering Journal*, 2003. **8**(2).

[21] Murugesan, S., and Deshpande, Y., *Web Engineering - Managing Diversity and Complexity of Web Application Development*. LNCS. Vol. 2016. 2001: Springer.

[22] Naumann, F. *From Databases to Information Systems - Information Quality Makes the Difference*. In, *Proceedings of the International Conference on Information Quality (IQ)*. 2001.

[23] Offutt, J. *Web Software Applications Quality Attributes*. In, *Quality Engineering in Software Technology (CONQUEST 2002)*. 2002. Nuremberg, Germany.

[24] Olsina, L. *Web-site Quality Evaluation Method: a Case Study on Museums*. In, *2nd Workshop on Software Engineering over the Internet at ICSE'99*. 1999.

[25] Olsina, L., Godoy, D., Lafuente, G.J., and Rossi, G., Specifying Quality Characteristics and Attributes for Websites. *LNCS*, 2001. **2016**.

[26] Pressman, R.S., *Software Engineering: A Practitioner's Approach*. 5 ed. 2001: McGraw-Hill.

[27] Schach, S., *Testing - Principles and Practice*, In, *The Computer Science and Engineering Handbook*, Tucker, A.B., Editor. 1996, CRC Press. pp. 2378-2398.

[28] Scharl, A., *Evolutionary Web Development*. 2000: Springer.

[29] Strong, D.M., Lee, Y.W., and Wang, R.Y., Data Quality in Context. *Communications of the ACM*, 1997. **40**(5): pp. 103-110.

[30] Voas, J. *Trusted Computing's Holy Grail*. In, *1st Workshop on Software Quality at ICSE'02*. 2002. Orlando, FL.

[31] Voas, J., A Baker's Dozen: 13 Grand Software Engineering Challenges (Keynote at the Euromicro 2003 conference in Belek, Turkey). 2003.

[32] Wallace, D., and Reeker, L., *Software Quality*, In, *Guide to the Software Engineering Body of Knowledge: SWEBOK*. 2001, IEEE CS.